UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

Análise para inclusão da Disciplina de Distribuição no ipPROCESS

TRABALHO DE GRADUAÇÃO

Aluna: Daniele Patrícia Santos (dps@cin.ufpe.br)
Orientadora: Edna Natividade da Silva Barros (ensb@cin.ufpe.br)

Recife, Junho de 2008

Assinatura

Esta monografia é resultado do Trabalho de Graduação da aluna Daniele Patrícia Santos, sob a orientação da professora Edna Natividade da Silva Barros, sob o título de Análise para inclusão da Disciplina de Distribuição no ipPROCESS. Todos abaixo estão de acordo com o conteúdo deste documento e os resultados deste Trabalho de Graduação.

_____

Edna Natividade da Silva Barros (Orientadora)

_____

Daniele Patrícia Santos (Aluna)

Dedicatória

Dedico este trabalho aos meus pais e àquelas que me acolheram em seu lar durante todos esses anos de graduação, compartilhando as alegrias e dificuldades: Dona Fátima, Brenda e Rebeca. A vocês todo meu amor e gratidão.

Agradecimentos

Agradeço a Deus, por ter me dado força e coragem de lutar pelos meus objetivos. À minha mãe, por todo amor, paciência e por nunca permitir que eu desistisse do meu sonho. Ao meu pai, pelo imenso amor e dedicação que sempre me contemplou, apesar de não estar mais fisicamente presente, estará eternamente em meu coração e pensamento.

À Dona Fátima, Brenda, Rebeca e Fred, pela amizade, apoio, confiança e por terem me mostrado o verdadeiro significado da palavra felicidade. Amo muito vocês.

À Aninha, Gustavo, Bruno e Marcela, pela longa e verdadeira amizade.

A todos da turma de Engenharia da Computação 2003.2, especialmente a Luciano, Rodrigo (Rodrigo Peixoto) e Galeguinho (Renato Bibiano), pela amizade e companheirismo.

A minha orientadora, Edna Barros, por todos os ensinamentos e apoio.

Um agradecimento especial à Fran (Francielle), pela imensurável ajuda em todas as etapas deste trabalho e a André Aziz. Muito obrigada pela amizade, apoio, dedicação e ensinamentos que vocês sempre me proporcionaram.

Aos meus grandes amigos: Rebeka Gomes e George Inácio, pessoas que eu tenho uma imensa admiração e sei que posso contar com elas tanto nas alegrias quanto nas lamentações, e quantas lamentações já escutaram de mim, não foi? (risos).

À Millena, por toda ajuda com o padrão SPIRIT e a André Feitoza, pela ajuda com o estudo de caso.

E àquele que apesar de brigarmos tanto e de não levar a sério meus conselhos, é um dos meus melhores amigos: Williams Thiago.

Resumo

O aumento da complexidade dos projetos de circuitos integrados associado a um *time-to-market* cada vez mais curto fez surgir um novo conceito de desenvolvimento, o *System-On-Chip* (SoC), onde módulos de propriedade intelectual (*Intelectual Property* - IP-core) pré-verificados e com diferentes funcionalidades são integrados em um único chip. Diante dessa nova abordagem, o sucesso no desenvolvimento de circuitos integrados depende, dentre outros aspectos, da qualidade dos IP-cores utilizados e do seu correto reuso. Para uma boa escolha do IP-core a ser utilizado e de sua correta integração, faz-se necessário o fornecimento, por parte dos seus desenvolvedores, de informações completas e concisas relativas a certas características do mesmo, como sua funcionalidade, arquitetura e restrições, por exemplo. Diante deste novo panorama surgiu o ipPROCESS, um processo de desenvolvimento de Soft IP-cores baseado no *Rational Unified Process* (RUP) para prototipação em *Field Programmable Gate Array* (FPGA). Apesar de propor tarefas e responsabilidades para a equipe desenvolvedora com o objetivo de produzir IP-cores de qualidade, o ipPROCESS não define passos para produzir e fornecer aos usuários informações necessárias acerca do IP-core que promovam o sucesso de sua integração em um SoC. Em vista disto, este trabalho tem como objetivo elaborar a disciplina de Distribuição (*Deployment*) a ser implantada no ipPROCESS. Esta disciplina consistirá em um fluxo de atividades e um conjunto de artefatos cujo intuito é a geração de informações necessárias a ser distribuída juntamente com o IP-core para a correta escolha e reuso do mesmo. Os artefatos definidos serão aplicados em um estudo de caso, para mostrar que parte das informações necessárias à elaboração dos mesmos são geradas durante a execução das demais disciplinas do ipPROCESS.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Em Abril de 1965, Gordon Moore, fundador da Intel, fez uma afirmação que ficou conhecida como a Lei de Moore, na qual ele dizia que o número de transistores de um circuito integrado dobraria a cada dois anos enquanto os custos permaneceriam constantes [1]. Atualmente tem-se um cenário parecido com a perspectiva de Moore, de acordo com [2], anualmente há um aumento de 58% na complexidade dos projetos de circuitos integrados enquanto o aumento na produtividade dos desenvolvedores é menor que 21%. Associado a isso se tem um mercado de produtos eletrônicos cada vez mais competitivo, com *time-to-market* decrescente, onde a sobrevivência das empresas depende da garantia da qualidade dos produtos desenvolvidos e do cumprimento de prazos e custos. Diante disso, percebeu-se que se novas metodologias não fossem adotadas, no futuro ficaria impossível finalizar com sucesso projetos na área. Surgiu então uma nova abordagem de desenvolvimento de circuitos integrados, a abordagem *System-On-Chip* (SoC), que consiste na utilização de módulos de propriedade intelectual (IP-cores) previamente projetados e verificados em um único chip. Na Figura 1.1 temos uma ilustração desta abordagem. A parte superior da Figura 1.1 apresenta o IP-core ou VC (*Virtual Core* ou *Virtual Component*), o qual apresenta uma funcionalidade específica e pode ser agregado a um circuito integrado. O SoC, parte inferior, é composto por uma série de componentes (IP-cores), combinados em um chip.

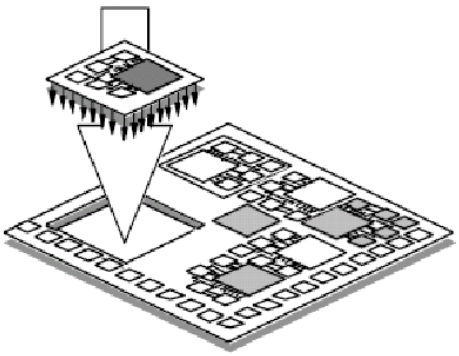

Figura 1.1: Abordagem de SoC e IP-core [3].

O uso de SoCs fez com que os projetistas se concentrassem no sistema completo sem ter que se preocupar com o comportamento exato ou com o desempenho de componentes individuais, porém a qualidade dos IP-cores utilizados e o sucesso da integração destes são peças-chaves neste tipo de projeto. Diante disto, é clara a necessidade por parte dos produtores de SoCs, os *IP-Integrators*, de uma criteriosa seleção dos VCs a serem utilizados, e por parte dos desenvolvedores de IP-cores, os *IP-Providers*, do desenvolvimento de IP-cores reusáveis e de qualidade.

Reuso não é um conceito novo. Na área de software, reuso de código é utilizado há décadas. Reuso de IP-cores difere significativamente do reuso em

software, pois neste último a funcionalidade do componente reusado é evidente quando o programa é compilado e rodado, enquanto o teste real para hardware é a sua fabricação, uma decisão que custa entre 500 mil a um milhão de dólares e que cujo esse valor pode ser perdido em um piscar de olhos devido a uma linha de código RTL (*Register Transfer Level*) com erro. Há então a necessidade de um processo de desenvolvimento sistemático para IP-cores que auxilie na sua qualidade e reusabilidade [4].

Há várias iniciativas tanto do mercado quanto acadêmicas de definição de metodologias e padrões que auxiliem o reuso e integração dos IP-cores, um exemplo disto é o *Reuse Methodology Manual* (RMM) [5] que apresenta um conjunto de práticas para o desenvolvimento de IP-cores reusáveis para uso em projetos de SoCs. Em se tratando de IP-cores, o segredo da reusabilidade não é apenas a transferência de código reusável, mas também a transferência do conhecimento para uma boa utilização de suas funcionalidades. A importância da qualidade da documentação não pode ser subestimada e deve ser uma das características chaves de um *IP-Provider* [6], ou seja, além de IP-cores reusáveis e de qualidade, é necessária a aquisição de informações destes sobre sua funcionalidade, arquitetura, restrições, entre outros aspectos, por parte dos *IP-Integrators* para uma correta escolha e reuso dos VCs.

Diante do panorama de desenvolvimento de projetos de circuitos integrados apresentado anteriormente, surgiu o ipPROCESS [7]. O ipPROCESS é um processo de desenvolvimento de Soft IP-cores baseado no *Rational Unified Process* (RUP) [8] com prototipação em *Field Programable Gate Array* (FPGA). Entende-se por Soft IP-cores o código fonte comportamental do sistema digital, ele é independente de tecnologia, podendo ser modificado e ser sintetizado usando diferentes tecnologias [9]. Apesar do ipPROCESS provê uma abordagem disciplinada para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento de componentes de hardware, ele não define atividades que forneçam informações a serem distribuídas juntamente com o IP-core para escolha e correto reuso do mesmo.

1.1. OBJETIVOS DO TRABALHO

Em face da deficiência do ipPROCESS citada anteriormente, este trabalho propõe a definição da disciplina Distribuição (*Deployment*) e sua implantação no já citado processo.

O objetivo deste trabalho é definir um fluxo de atividades e um conjunto de artefatos que possam fornecer aos *IP-Integrators* as informações necessárias e suficientes à integração do IP-core em um SoC, informações estas que foram previamente identificadas através do Trabalho de Graduação intitulado ipQuality-Identificação e Análise de Atributos de Qualidade de soft IP-Cores [10].

Para tanto, serão analisados padrões de distribuição e metodologias de desenvolvimento tanto da área de software quanto de hardware bastante utilizados no mercado como o VSIA (*Virtual Socket Interface Alliance*) [11], o RMM (*Reuse Methodology Manual*), o IEEE [12], o SPIRIT [13] e o RUP (*Rational Unified Process*).

1.2. *ESTRUTURA DO TRABALHO*

Este trabalho encontra-se organizado da seguinte forma:

- Capítulo 2 - apresenta o estado da arte na área de pesquisa;
- Capítulo 3 – descreve o processo de desenvolvimento ipPROCESS e o ipQUALITY;
- Capítulo 4 – apresenta a proposta deste trabalho, a disciplina *Deployment*, com seu fluxo de atividades e artefatos;
- Capítulo 5 - demonstra a geração, em um estudo de caso, dos artefatos da disciplina proposta;
- Capítulo 6 - apresenta as conclusões obtidas com o desenvolvimento deste trabalho assim como propostas de trabalhos futuros.

2. Estado da Arte

Com o intuito de atender as demandas de seus mercados relacionadas à qualidade do produto e cumprimento de prazos e custos, vem surgindo novas abordagens de desenvolvimento tanto de softwares quanto de hardwares. Este capítulo apresentará algumas destas abordagens, dando ênfase à metodologia de desenvolvimento de software RUP e em relação à hardwares, a ênfase é dada a padrões de distribuição de IP-cores assim como ao RMM.

2.1. *RATIONAL UNIFIED PROCESS – RUP*

O *Rational Unified Process* (RUP) é um processo de engenharia de software desenvolvido pela *Rational Software Corporation* [14], atualmente uma divisão da IBM [15]. Ele oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Seu objetivo é garantir a produção de software de alta qualidade que atenda as necessidades dos usuários dentro de um cronograma e de um orçamento previsíveis.

A figura abaixo apresenta a arquitetura geral do RUP.

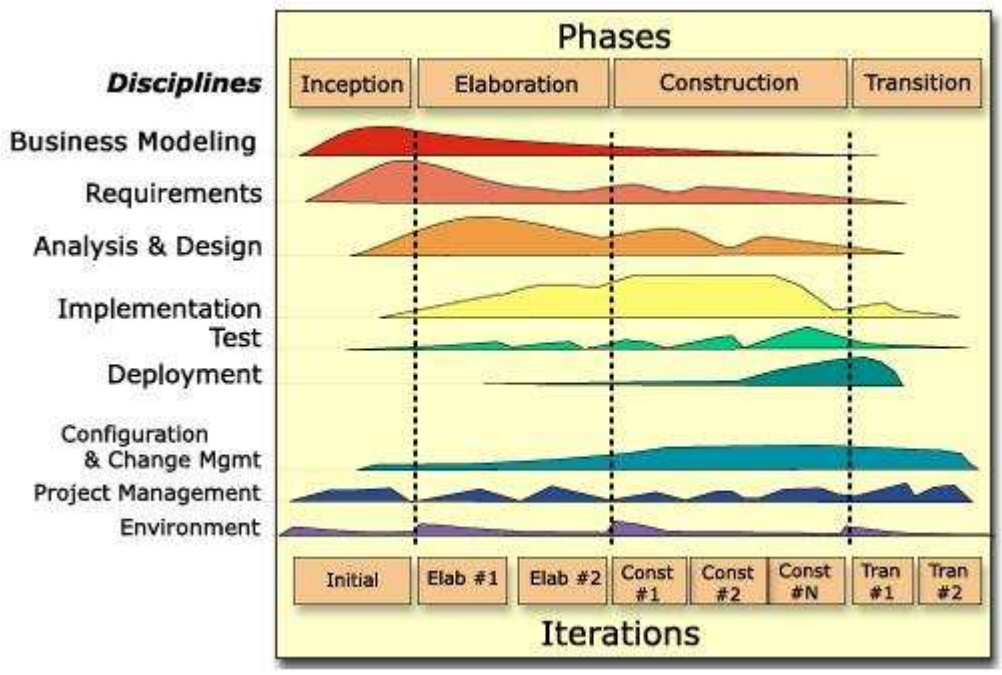

Figura 2.1: Arquitetura do RUP [16].

O RUP pode ser descrito em duas dimensões:

- Eixo horizontal: representa o tempo e mostra o aspecto dinâmico do processo, é expresso em termos de ciclos, fases, iterações e marcos.

- Eixo vertical: representa o aspecto estático do processo, ou seja, como ele é descrito em termos de componentes, disciplinas, atividades, fluxos de trabalho, artefatos e papéis do processo.

No *Rational Unified Process* (RUP), o desenvolvimento do software é dividido em ciclos, cada ciclo trabalhando em cima de uma nova geração do produto e sendo cada um deles dividido em quatro fases consecutivas, são elas:

- Concepção: A meta desta fase é estabelecer os objetivos do ciclo de vida do projeto e delimitar seu escopo. Esta fase tem muita importância para o desenvolvimento de sistemas novos, onde há muitos riscos de negócio e de requisitos que precisam ser cuidadosamente tratados a fim de permitir o prosseguimento do projeto. Os objetivos principais desta fase são estabeler o escopo do software do projeto e suas condições limites, discriminar os casos de uso críticos do sistema, estimar o custo, risco e programação do projeto, assim como preparar seu ambiente de suporte.
- Elaboração: Tem como propósito analisar o domínio do problema, desenvolver uma arquitetura para o sistema em questão, desenvolver o plano de projeto e eliminar os elementos que forneçam altos riscos ao projeto.
- Construção: Possui como meta o esclarecimento dos requisitos restantes e a conclusão do desenvolvimento do sistema com base na arquitetura definida na fase anterior. A ênfase desta fase dá-se no gerenciamento de recursos e controle de operações que otimizem custos, programação e qualidade.
- Transição: Tem como foco assegurar que o software esteja disponível aos seus usuários finais. Esta fase pode atravessar várias iterações e inclui atividades de teste do produto no ambiente do usuário a fim de obter melhorias de acordo com o *feedback* recebido.

Nove disciplinas compõem o RUP, sendo estas executadas com diferentes ênfases nas fases anteriormente descritas, são elas:

- Modelagem do Negócio: Tem como finalidade entender a estrutura e dinâmica da organização onde o sistema será implantado.
- Requisitos: Possui como finalidade o estabelecimento e concordância das funcionalidades do sistema, assim como sua delimitação.
- Análise e Projeto: A meta desta disciplina é a transformação dos requisitos em um *design* do sistema a ser criado, desenvolvendo uma arquitetura para o sistema.
- Implementação: Tem como objetivo principal o desenvolvimento do código fonte do sistema.
- Teste: Seu objetivo é verificar se a implementação está de acordo com o que foi definido com o cliente.
- Implantação: Descreve as atividades necessárias à disponibilização do produto aos seus usuários finais.

- Gerência de Configuração e Mudanças: Apresenta um conjunto de atividades responsáveis pelo controle e gerenciamento das mudanças ocorridas nos artefatos do projeto.
- Gerenciamento de Projeto: Contém as atividades cujo objetivo é dar suporte ao gerente de projetos na organização e gerenciamento do desenvolvimento do projeto.
- Ambiente: Tem como meta o fornecimento à organização desenvolvedora, do ambiente de desenvolvimento de software, formado por processos e ferramentas, que dará suporte à equipe de desenvolvimento.

Neste trabalho será dada atenção especial à fase Transição e à disciplina Implantação do RUP, identificando entre suas características, atividades e artefatos os que podem ser aplicados na distribuição de IP-cores.

2.2. *MANUAL DE REUSO – RMM*

O *Reuse Methodology Manual* (RMM) apresenta um conjunto de boas práticas a serem adotadas no desenvolvimento de IP-cores com o objetivo de torná-los reusáveis no contexto de uma metodologia de SoCs.

Cinco fases são sugeridas para serem seguidas durante o desenvolvimento de um IP-core, essas fases são apresentas na figura abaixo:

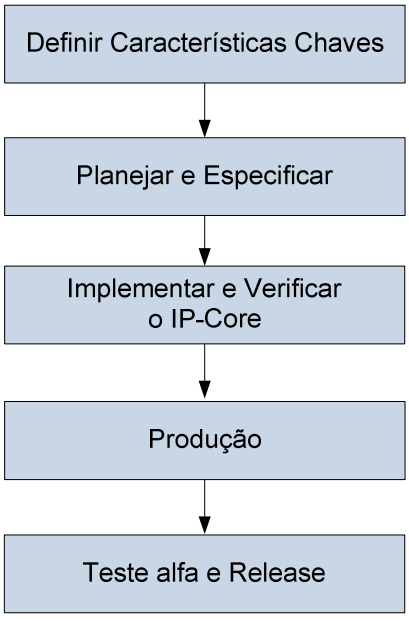

Figura 2.2: Fases de um projeto de IP-core definidas pelo RMM.

1. Definir Características Chaves: Define as principais características do IP-core. Essas características incluem funcionalidade básica e parâmetros de configuração.
2. Planejar e Especificar: O objetivo desta segunda fase é desenvolver uma especificação funcional detalhada do IP-core, assim como uma especificação refinada para a verificação do mesmo, além de um plano para o resto do projeto.
3. Implementar e Verificar o IP-core: Com especificações e plano definidos, os desenvolvedores implementam e verificam o IP-core.
4. Produção: O objetivo desta fase é o desenvolvimento de um pacote contendo os artefatos que serão distribuídos para os usuários finais.
5. Teste Alfa e Release: Nesta etapa, o pacote produzido na fase anterior é testado com o intuito de verificar sua completude e se está pronto para ser utilizado pelos usuários do IP-core.

Como as fases Produção e Teste Alfa e Release têm seus objetivos relacionados com a distribuição de IP-cores a seus usuários finais, estas serão analisadas para o desenvolvimento deste trabalho.

2.3. VSIA

O VSIA (*Virtual Socket Interface Alliance*) é uma aliança criada em 1996, formada por líderes mundiais em semicondutores. Ele tem como objetivo a definição de um padrão para documentação, formato de dados, interfaces de comunicação e abordagem de verificação que auxilie na concepção de IP-cores reusáveis, proporcionando com isso um menor esforço na integração destes em um SoC.

São três os pilares de atuação do VSIA:

- Pilar de Qualidade do IP: Estabelece métricas de qualidade para IP-cores, também para sua verificação e desenvolvimento.
- Pilar de Proteção do IP: Desenvolve soluções para o balanceamento da segurança da propriedade intelectual.
- Pilar de Transferência do IP: Provê padrões que facilitem a troca de IP-cores entre seus desenvolvedores e usuários, tendo como principais objetivos automação do fluxo de projeto e fornecimento incremental do IP-core.

Como resultado de seu trabalho, vários documentos têm sido publicados pelo VSIA, eles podem ser divididos em três categorias:

- Especificações: Especificam um conjunto de requisitos ou dados que devem ser disponibilizados, a fim de realizar transferência de informações dos desenvolvedores do IP-core, os *IP-Providers*, para os seus usuários, os *IP-Integrators*. São disponibilizados padrões que devem ser seguidos para o fornecimento das informações citadas.

- Padrões: Apresentam estrutura, protocolo ou forma como as informações dos *IP-Providers* devem ser apresentadas aos *IP-Integrators*.
- Documentos Técnicos: Formados por definições, artigos, taxonomias e *surveys*.

O VSIA utilizou-se de padrões já definidos pelas corporações que a formam para construir seu padrão, um exemplo disto é a disponibilização por parte da Motorola [17] do seu padrão de desenvolvimento de semicondutores, o *Semiconductor Reuse Standard* (SRS).

2.3.1. SRS

Tendo parte de suas especificações disponibilizadas na web em 1999 [18] e fornecidas integralmente ao VSIA no início de 2000 [19], o SRS, como dito anteriormente, consiste no padrão de desenvolvimento de IP-cores da então área de semicondutores da Motorola, que atualmente é a companhia Freescale [20]. O SRS especifica artefatos a serem entregues aos usuários do IP-core e estabelece requisitos para o desenvolvimento de VCs reusáveis e o compartilhamento e troca de informações entre o *IP-Creator* e o *IP-Integrator* [21]. Os seguintes artefatos foram disponibilizados para a comunidade em geral:

- IP/VC Block Deliverables: Define um conjunto de dados que devem ser desenvolvidos pelo *IP-Creator* e distribuídos ao *IP-Integrator*. Provê também definições de metadados, estrutura dos diretórios de armazenamento e convenções de nomes.
- IP Interface (IPI): Provê um completo conjunto de definições de interfaces padrões que promove rápida integração no SoC.
- Verilog HDL Coding: Estabelece um padrão de codificação para implementações na linguagem Verilog HDL.
- Documentation: Determina o tipo, formato e conteúdo de uma série de documentos relacionados à IP-cores. Esses documentos estão relacionados à criação, uso, integração e testes do IP-core.

Tanto o VSIA como o SRS apresentam padrões de interfaces e metodologias de verificação a serem seguidos, assim como estabelecem um conjunto de informações e dados que devem ser disponibilizados aos usuários do IP-core, não apresentando porém, um fluxo de atividades que possa produzir esses artefatos. Sendo assim, este trabalho não usará diretamente os padrões VSIA e SRS, seu uso se dará através do ipQUALITY, que será explicado mais adiante.

2.4. SPIRIT

O consórcio SPIRIT (*Structure for Packaging, Integrating and Re-using IP within Tool-flows*) utiliza a longa experiência de empresas da área de desenvolvimento de IP-cores, construção de SoCs e de ferramentas EDA (*Electronic Design Automation*) para elaborar um padrão para distribuição de IP-cores, com o intuito de auxiliar ferramentas de integração destes.

O padrão desenvolvido pelo SPIRIT é o IP-XACT, ele atua em duas abordagens:

- Descrição de metadados do IP-core: A partir de esquemas XML cria uma linguagem comum para a descrição de IP-cores que seja compatível com técnicas de integração automática e que proporcionem o uso de IP-cores de diferentes fontes pelas ferramentas que adotem o IP-XACT.
- API para ferramentas de integração: Provê um mecanismo para a troca de dados sobre a arquitetura do projeto entre ferramentas, auxiliando a integração do sistema.

Como pode ser visto, o SPIRIT define um padrão para ferramentas de integração de IP-cores e para o empacotamento destes, não apresentando um fluxo de atividades para a distribuição do IP-core aos seus usuários. Sendo assim, o padrão SPIRIT será utilizado como um guia recomendado para empacotamento do IP-core, não estando a disciplina descrita neste trabalho "amarrada" a este padrão.

2.5. IEEE

O IEEE (*Institute of Electrical and Electronics Engineers*) é uma organização técnico-profissional internacional, dedicada ao avanço da teoria e prática da engenharia nos campos da eletricidade, eletrônica e computação [22].

Dentre as atividades do IEEE está o estabelecimento de padrões. Para este trabalho será analisado o padrão IEEE 1063 – Padrão para documentação do usuário de um determinado software, com o objetivo de estabelecer características relacionadas à estrutura que podem ser aplicadas à documentação do usuário de um IP-core.

2.6. ISO

A ISO (*International Organization for Standardization)* [23] é uma rede de organismos padronizadores de 157 países, sendo o maior desenvolvedor e publicador de padrões internacionais do mundo. Dentre os padrões desenvolvidos pela ISO, encontra-se o NBR ISO/IEC 12119: Tecnologia da Informação – Pacotes de

software – Teste e requisitos de qualidade. Este padrão estabelece os requisitos de qualidade para pacotes de software e instruções de como testá-los com relação aos requisitos estabelecidos.

A contribuição dessa norma para esse trabalho será através da análise dos requisitos de qualidade para pacotes de software que podem ser aplicados em IP-cores, visto que dependo da maneira que seja feita a distribuição se um soft IP-core, esta pode ter semelhanças com a distribuição de softwares realizada através de pacotes, os conhecidos softwares de prateleira.

2.7. O IPPROCESS

O ipPROCESS é um processo de desenvolvimento de soft IP-cores com prototipação em FPGA. Este processo, desenvolvido no contexto do BRAZIL-IP [24], provê uma abordagem disciplinada para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento de componentes de hardware, com o objetivo de desenvolver IP-cores de alta qualidade e que atendam as necessidades dos seus usuários.

A relação deste trabalho com o ipPROCESS se dará com a inclusão no mesmo de um fluxo de atividades e um conjunto de artefatos que forneçam as informações necessárias à escolha e reuso do IP-core. Tais atividades e artefatos irão compor a disciplina de Distribuição (*Deployment*).

2.8. CONCLUSÃO

Como podemos observar, devido à necessidade de reuso dos IP-cores, várias empresas atuantes na área reuniram-se em consórcios e alianças, a fim de estabelecer padrões de documentação e artefatos a serem fornecidos aos usuários dos VCs, com o intuito de diminuir o esforço de integração dos mesmos. Porém, há poucas iniciativas que além do estabelecimento de um conjunto de dados e de artefatos a serem entregues, definem um fluxo de atividades que seja incluído no processo de desenvolvimento de IP-cores com a finalidade de produzir esses artefatos.

Diferentemente, na área de software, além de padrões para a documentação do usuário, há também metodologias, como o RUP, que definem um fluxo de atividades a ser realizado durante o desenvolvimento do software com o objetivo da disponibilização deste aos seus usuários.

Sendo assim, podemos perceber a importância de uma disciplina que, além de produzir um conjunto de artefatos a serem entregues aos usuários de um IP-core, possua um conjunto de atividades distribuídas ao longo do desenvolvimento do mesmo de forma a suportar a produção destes artefatos.

3. Iniciativas de Metodologia de Desenvolvimento e de Análise da Qualidade de IP-cores

Seguindo a tendência das grandes companhias desenvolvedoras de IP-cores em pesquisar novas abordagens para o desenvolvimento destes que promovam seu reuso e o aumento de sua qualidade, foi desenvolvido no contexto do BRAZIL-IP, como já foi dito, uma metodologia de desenvolvimento de Soft IP-cores, o ipPROCESS, além disso, está sendo elaborada uma iniciativa relacionada à análise da qualidade dos IP-cores, o ipQUALITY.

Este capítulo apresentará essas iniciativas, mostrando seus papéis fundamentais na elaboração deste trabalho, uma vez que a disciplina proposta neste documento, a disciplina *Deployment*, será inserida ao ipPROCESS e sua documentação deve estar de acordo com os atributos de qualidade relacionados à documentação do usuário definidos pelo ipQUALITY.

3.1. O IPPROCESS

Como já foi dito anteriormente neste documento, o ipPROCESS é um processo de desenvolvimento de Soft IP-cores com prototipação em FPGA baseado no RUP. Podemos observar na Figura 3.1 a arquitetura geral deste processo.

Assim como o RUP, o ipPROCESS é um processo iterativo e incremental, o desenvolvimento de um soft IP-core no ipPROCESS é realizado através de várias iterações, sendo que no final de cada iteração parte do produto está disponível.

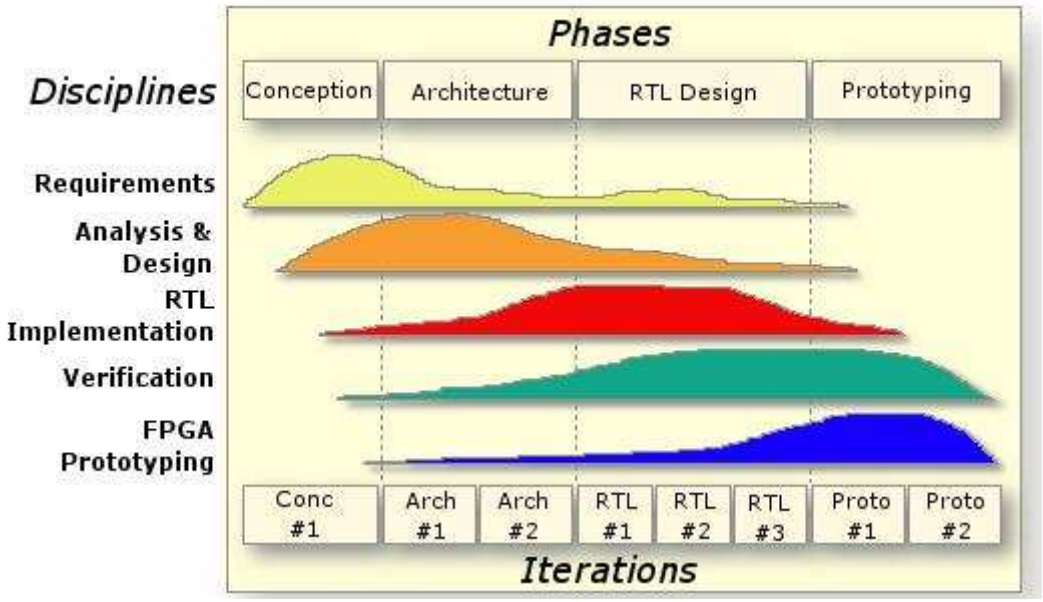

Figura 3.1: Arquitetura Geral do ipPROCESS [7].

3.1.1. Fases

A partir de uma perspectiva de gerenciamento, o ciclo de vida do desenvolvimento de um Soft IP-core no ipPROCESS é composto por quatro fases, sendo estas seguidas seqüencialmente e cuja conclusão de cada uma é definida por um *milestone*, ou seja, um marco principal. Pode-se dizer que cada fase é basicamente um intervalo de tempo entre dois marcos principais. A figura abaixo apresenta as quatro fases do ipPROCESS assim como seus *milestones*.

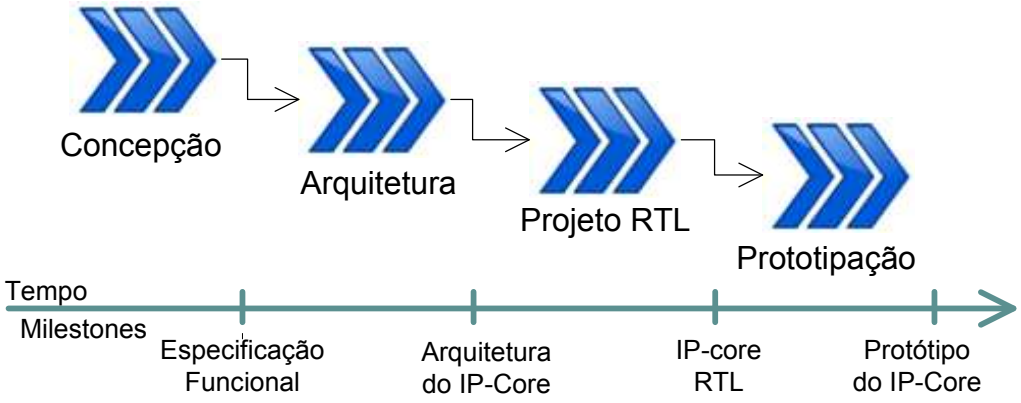

Figura 3.2: Fases do ipPROCESS.

- Concepção: Essa fase tem como objetivos elicitar os requisitos do projeto e definir seu escopo. O marco resultante desta fase é a Especificação Funcional, que como o próprio nome diz, é a definição das funcionalidades que o IP-core deverá apresentar.
- Arquitetura: A fase de arquitetura provê a definição de uma arquitetura estável para o IP-core. A conclusão dessa fase é definida com a elaboração da arquitetura do IP-core.
- Projeto RTL: A meta desta fase é desenvolver um modelo RTL de simulação sintetizável baseado na arquitetura definida na fase anterior. Nesta fase o IP-core é implementado, sendo também desenvolvido um ambiente para a verificação do mesmo de acordo com um plano de verificação definido. O marco que conclui esta fase é a implementação do IP-core no nível de abstração RTL, vale ressaltar que esta implementação deve ter sido verificada.
- Prototipação: Nesta fase é criado um protótipo físico com o intuito de garantir que o IP-core possa ser distribuído para seus usuários finais. Para isso, os módulos implementados e verificados na fase anterior são prototipados em FPGA de acordo com um plano definido. Após sua prototipação, estes módulos precisam ser testados a fim de garantir que nenhum erro foi incorporado durante essa atividade. O marco resultante desta fase é o protótipo do IP-core.

3.1.2. Disciplinas

O ipPROCESS é composto por cinco disciplinas, são elas: *Requirements*, *Analysis & Design*, *RTL Implementation*, *Verification* e *FPGA Prototyping*.

A disciplina ***Requirements*** tem a finalidade de estabelecer o escopo do projeto, definindo seus requisitos funcionais e não-funcionais, assim como as limitações do IP-core a ser desenvolvido. Para isto, as atividades e artefatos mostrados na figura abaixo são realizados e produzidos respectivamente por esta disciplina. O responsável pelas atividades desta disciplina é o *Requirements Analyst*.

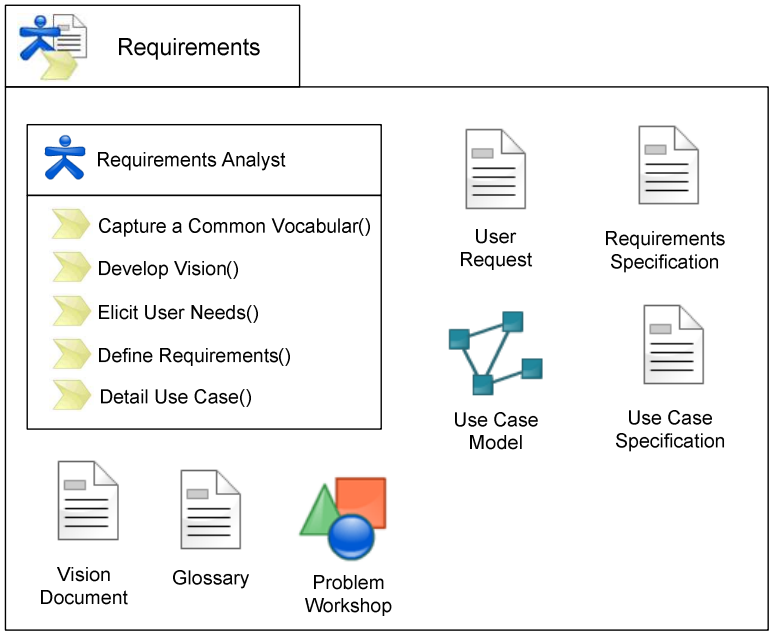

Figura 3.3: Visão geral da disciplina *Requirements*.

As atividades da disciplina *Requirements* são as seguintes:

- *Capture a Common Vocabulary*: Seu objetivo é capturar os termos e expressões utilizados pelos *stakeholders* do projeto.
- *Develop Vision*: Tem como objetivo obter uma visão inicial do escopo do projeto, assim como a visão de todos os envolvidos no projeto e suas necessidades.
- *Elicit User Needs*: Sua meta é elicitar as necessidades dos usuários do IP-core a ser desenvolvido.
- *Define Requirements*: Define e prioriza os requisitos do IP-core.
- *Detail Use Case*: Identifica e detalha os casos de uso do IP-core.

Como resultados das atividades descritas acima têm-se os seguintes artefatos:

- *User Request*: Este artefato contém qualquer tipo de solicitação dos principais envolvidos em relação ao IP-core que será desenvolvido, como

usuários, cliente e etc. Podendo também conter referências a qualquer tipo de fonte externa com a qual o IP-core deve estar de acordo.

- *Glossary*: Apresenta um conjunto consistente de definições dos termos específicos do projeto.
- *Vision Document*: Dá uma visão inicial do escopo do projeto, identificando os envolvidos no projeto e suas necessidades.
- *Problem Workshop*: Tem como objetivo auxiliar no entendimento de questões relativas ao projeto ou ao contexto do IP-core.
- *Requirements Specification:* Lista e prioriza os requisitos funcionais e não-funcionais do IP-core, ou seja, apresenta a funcionalidade que o IP-core deve apresentar assim como as restrições que este deve atender.
- *Use Case Model:* É o modelo das funcionalidades pretendidas pelo IP-core e seu ambiente.
- *Use Case Specification:* Apresenta o detalhamento dos casos de uso identificados do IP-core.

A disciplina ***Analysis & Design*** transforma os requisitos identificados na disciplina anterior em uma arquitetura para o IP-core, sendo definidos os componentes que o compõem, suas interfaces e protocolos de comunicação. O projeto realizado nesta disciplina deve levar em consideração a tecnologia e ambiente de implementação, além dos padrões de projetos aplicáveis. O *IP Architect* e o *Designer* são os responsáveis pela execução das atividades e elaboração dos artefatos desta disciplina.

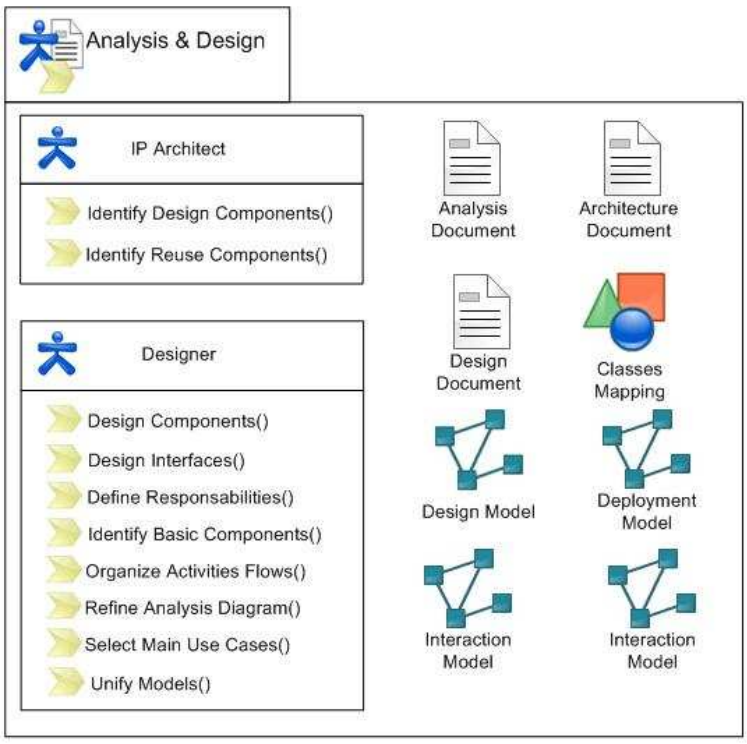

Figura 3.4: Visão Geral da disciplina *Analysis & Design*.

A disciplina de *Analysis & Design* possui as seguintes atividades:

- *Select Main Use Cases:* Esta atividade tem como objetivo selecionar dentre os casos de uso elicitados, aqueles que serão analisados.
- *Identify Basic Components:* Seu objetivo é a identificação, no fluxo do caso de uso, de elementos que auxiliem a definição de um modelo que o implemente.
- *Organize Activities Flows:* Distribui as ações do caso de uso entre os componentes identificados na atividade anterior.
- *Define Responsabilities:* Formaliza a distribuição das responsabilidades entre os elementos identificados.
- *Refine Analysis Diagram*: Sua meta é a finalização do modelo de análise.
- *Unify Models:* O objetivo desta atividade é unificar os modelos de análise encontrados para cada caso de uso.
- *Identify Design Components*: Identifica elementos para o modelo do projeto do IP-core a partir do seu modelo de análise.
- *Identify Reuse Components*: Possui como meta a identificação de componentes que possam ser reusados no modelo de projeto.
- *Design Interfaces:* Seu objetivo é projetar protocolos de comunicação entre os módulos internos e externos ao IP-core, modelando todo seu comportamento.
- *Design Components:* Projeta toda estrutura interna do IP-core, como processos, portas e interfaces, por exemplo.

Os artefatos da disciplina *Analysis & Design* são:

- *Analysis Model*: Este modelo apresenta o resultado da realização dos casos de uso.
- *Analysis Document:* Descreve a realização dos casos de uso, suas iterações e o modelo de análise.
- *Architecture Document:* Apresenta a descrição da arquitetura do IP-core, apresentando os módulos que o compõem e identificando padrões de projetos utilizados.
- *Classes Mapping*: Apresenta um mapeamento entre os elementos identificados na análise com os elementos que constituem o projeto do IP-core.
- *Design Model*: Apresenta o resultado final do projeto do IP-core, apresentando os módulos que o compõem, seus relacionamentos, portas e protocolos.
- *Design Document:* Apresenta um detalhamento de toda estrutura projetada para o IP-core, mostrando seus componentes, portas e protocolos de comunicação.
- *Deployment Model:* Este modelo apresenta a distribuição dos componentes do IP-core em diferentes plataformas.
- *Interaction Model*: Apresenta a interação entre as classes identificadas.

A disciplina ***RTL Implementation*** tem o objetivo de implementar, utilizando alguma linguagem de descrição de hardware, também conhecida como linguagem HDL (*Hardware Description Level*), os componentes identificados na disciplina anterior. Além da implementação, testes unitários são realizados a fim de garantir que os módulos produzidos tenham sua funcionalidade verificada. Os papéis responsáveis pelas atividades e artefatos desta disciplina são o *RTL Designer*, o *RTL Implementer* e o *RTL Integrator*. A figura abaixo apresenta uma visão geral desta disciplina.

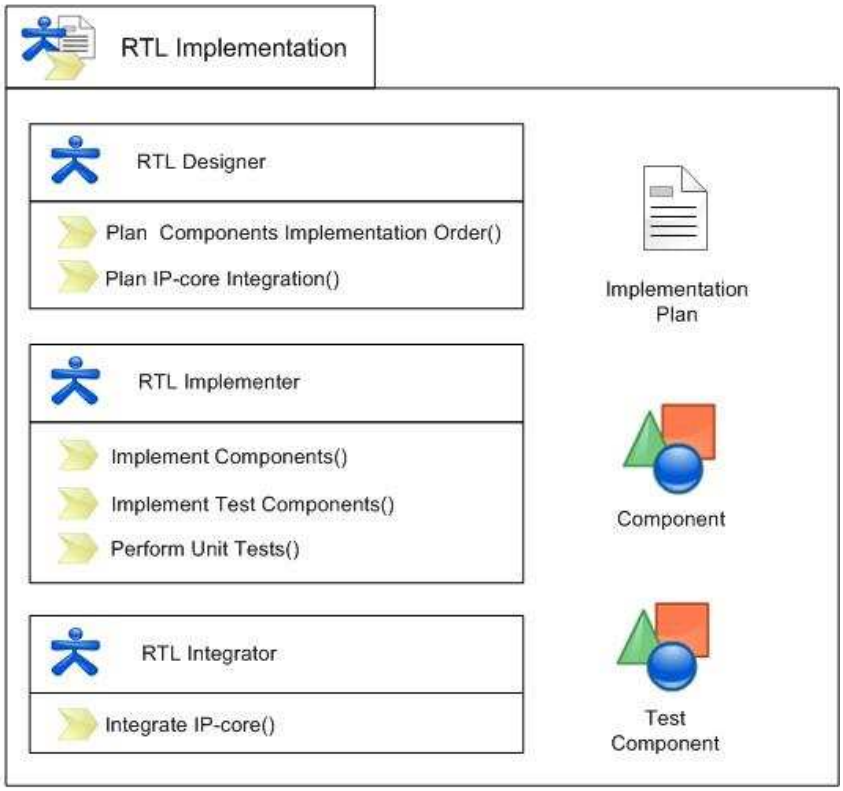

Figura 3.5: Visão geral da disciplina *RTL Implementation*.

As atividades da disciplina *RTL Implementation* são as seguintes:

- *Plan Components Implementation Order*: O objetivo desta atividade é planejar a ordem com que os componentes identificados na disciplina anterior serão implementados.
- *Plan IP-core Integration*: Esta atividade tem como meta planejar a ordem com que os módulos implementados serão integrados.
- *Implement Components*: Tem como objetivo a implementação dos componentes identificados na disciplina *Analysis & Design*, seguindo a ordem definida na atividade *Plan Components Implementation Order*.
- *Implement Test Components:* O objetivo desta atividade é implementar componentes que auxiliem a realização da verificação dos componentes implementados na atividade anterior.
- *Perform Unit Tests:* Uma vez terminada a implementação de um componente, deve-se realizar testes neste, utilizando os componentes implementados na

atividade *Implement Test Components*, com o propósito de verificar sua funcionalidade.

- *Integrate IP-core:* Após a implementação e verificação dos componentes que compõem o IP-core, estes devem ser integrados de acordo com a ordem definida na atividade *Plan IP-core Integration*.

Como resultados das atividades descritas acima, temos os seguintes artefatos:

- *Implementation Plan*: Apresenta o planejamento da implementação dos módulos que compõem o IP-core, definido a ordem com que os módulos serão implementados assim como a ordem em que serão integrados.
- *Components*: São os componentes resultantes da implementação, como códigos-fonte, arquivos binários, arquivos de dados e documentação, por exemplo.
- *Test Components*: Abrangem elementos que produzem o comportamento necessário à execução de testes dos componentes implementados.

Após a implementação do IP-core, é necessária a verificação do mesmo para garantir que sua funcionalidade está de acordo com os requisitos estabelecidos. A disciplina ***Verification*** apresenta um conjunto de atividades e artefatos cujo objetivo é verificar se os componentes foram implementados de acordo com o que foi projetado e se atendem aos requisitos elicitados. Os papéis responsáveis pela execução das atividades da disciplina *Verification* são o *Test Designer*, *Test Implementer* e o *Tester*. Temos na Figura 3.6 uma visão geral da disciplina *Verification*, com suas atividades, artefatos e papéis.

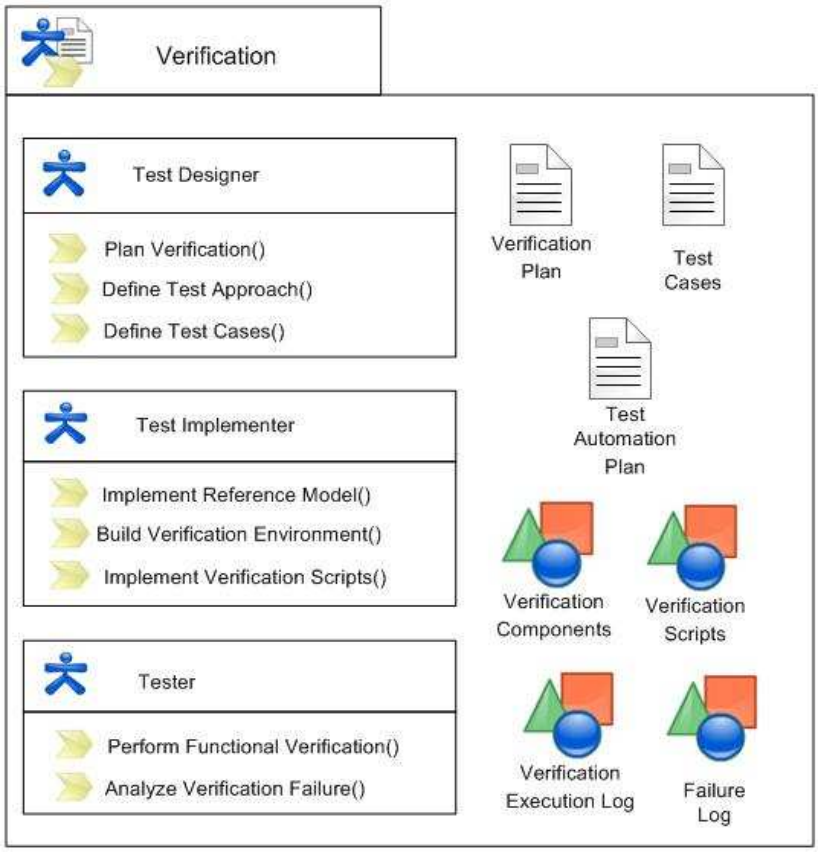

Figura 3.6: Visão geral da disciplina *Verification*.

As atividades da disciplina *Verification* são:

- *Plan Verification*: Esta atividade é responsável pelo planejamento da verificação do IP-core, definido as metas e objetivos que devem ser alcançados com a verificação, assim como a identificação dos itens que serão verificados e os recursos necessários.
- *Define Test Approach*: O objetivo desta atividade é definir a abordagem que será utilizada na verificação do IP-core, identificando as técnicas que deverão ser usadas assim como a abrangência da verificação.
- *Define Test Cases*: Esta atividade define e especifica o conjunto de casos de testes que serão usados na verificação funcional do IP-core.
- *Implement Reference Model*: Seu objetivo é a implementação de um modelo de referência do IP-core. Entende-se por modelo de referência uma implementação em alto nível de abstração da especificação do sistema [25].
- *Build Verification Environment*: A meta desta atividade é a implementação dos componentes necessários à verificação do IP-core. A implementação realizada nesta atividade deve ser baseada nos casos de testes já definidos e especificados.
- *Implement Verification Scripts*: Aqui são implementados *scripts* que serão utilizados na automação dos testes aplicados no IP-core.

- *Perform Functional Verification*: Nesta atividade é realizada a verificação funcional do IP-core, de acordo com o que foi planejado na atividade *Plan Verification* e utilizando os componentes implementados e os casos de testes definidos nas demais atividades dessa disciplina.
- *Analyze Verification Failure:* O objetivo desta atividade é analisar os resultados da verificação funcional do IP-core, definindo os componentes que contém erros.

Como artefatos da disciplina *Verification* têm-se:

- *Verification Plan*: Apresenta um plano para a verificação funcional do IP-core, definindo a abordagem de verificação que será utilizada, além das atividades e recursos necessários.
- *Test Cases*: Especifica o conjunto de casos de testes que serão utilizados na verificação funcional do IP-core.
- *Test Automation Plan*: Apresenta um plano para automação da verificação.
- *Verification Components*: Compreendem os componentes implementados para a realização da verificação funcional do IP-core.
- *Verification Scripts*: São os scripts utilizados na execução automática dos testes do IP-core.
- *Verification Execution Log*: Compreende as saídas capturadas com a execução da verificação.
- *Failure Log*: Coleção das falhas encontradas após a execução dos testes.

A disciplina de ***FPGA Prototyping*** tem o objetivo de prototipar em FPGA os componentes implementados, para tanto, as atividades, artefatos e papéis apresentados na figura abaixo a compõem.

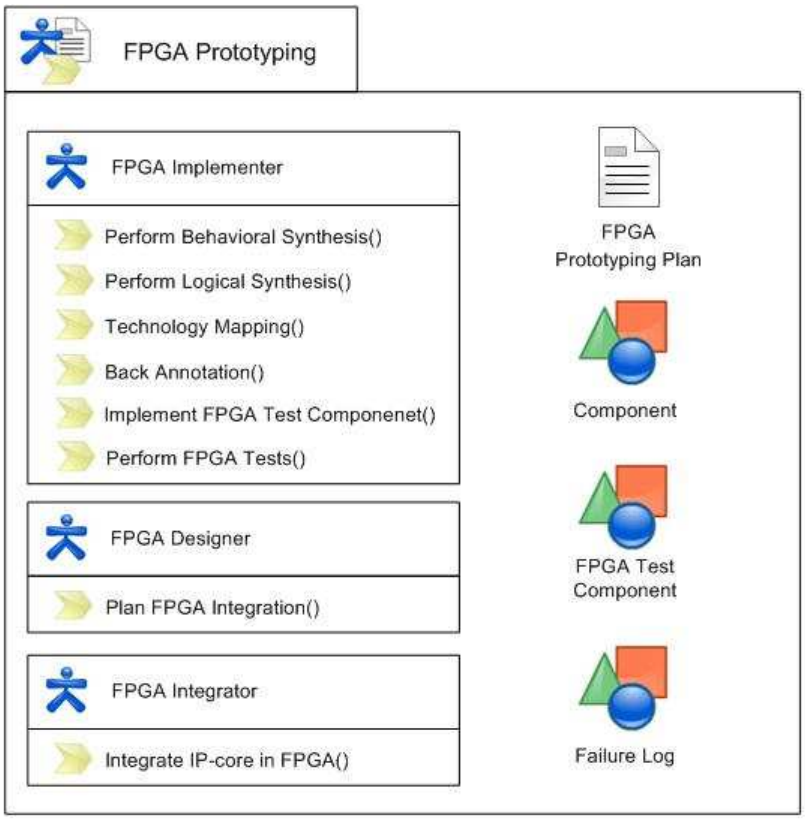

Figura 3.7: Visão geral da disciplina *FPGA Prototyping*.

A disciplina *FPGA Prototyping* possui as seguintes atividades:

- *Perform Behavioral Synthesis*: O objetivo desta atividade é refinar o código implementado do IP-core de um nível comportamental, nível no qual foi implementado na disciplina *Implementation*, para o nível de registradores, também conhecido como RTL.
- *Perform Logical Synthesis*: Nesta etapa, o código do IP-core resultante da atividade anterior será refinado para o nível de *gates*.
- *Technology Mapping*: A meta desta atividade é transformar o componente implementado do IP-core, a nível de *gates*, em um componente que pode ser prototipado em FPGA.
- *Back Annotation*: Esta atividade tem como objetivo fazer a cossimulação do IP-core, ou seja, simular os componentes a nível de *gates* com os componentes de verificação utilizados na verificação funcional.
- *Implement FPGA Test Component*: Nesta atividade são implementadas estruturas que auxiliem a execução de testes dos componentes prototipados em FPGA.
- *Perform FPGA Tests*: Aqui, o objetivo é a realização de testes que verifiquem se algum erro foi inserido durante a prototipação dos componentes no FPGA.
- *Plan FPGA Integration*: Esta atividade tem como meta o planejamento da integração dos componentes prototipados do IP-core, definindo sua ordem de integração.

- *Integrate IP-core in FPGA*: Durante esta atividade, os componentes do IP-core prototipados no FPGA são integrados de acordo com o que foi estabelecido na atividade anterior.

Como resultados das atividades descritas acima, temos os seguintes artefatos:

- *FPGA Prototyping Plan*: Apresenta o plano da prototipação em FPGA do IP-core. Define a ordem com que os componentes serão prototipados em FPGA e a ordem em que serão integrados.
- *Component*: Diz respeito aos componentes prototipados em FPGA.
- *FPGA Test Component*: Abrangem os componentes implementados com o intuito de provê o ambiente e comportamento necessário para a verificação dos componentes prototipados.
- *Failure Log*: Compreende a descrição das falhas encontradas com a execução dos testes nos componentes prototipados.

3.2. O IPQUALITY

Como já mencionado, a qualidade dos IP-cores utilizados é essencial para o sucesso do desenvolvimento de SoCs. Visando determinar atributos que mensurem a qualidade de um IP-core surgiu o ipQUALITY.

Atualmente o ipQUALITY define atributos de qualidade relacionados à funcionalidade de um IP-core e sua documentação do usuário (documentação a ser entregue ao integrador do IP-core), atributos estes identificados pelo trabalho de graduação intitulado ipQUALITY: Identificação e Análise de Atributos de Qualidade de Soft IP-Cores.

Os seguintes atributos são definidos pelo ipQUALITY em relação à documentação e suporte ao usuário:

Documentação do Usuário: *Capacidade do IP-core de prover artefatos de qualidade, de acordo com o propósito pretendido pelo IP-core e necessidades dos usuários. Constituindo assim a documentação completa e suficiente para a integração do módulo de propriedade intelectual em um projeto.*

- **Acurácia:** *O IP-core deve apresentar artefatos com conteúdo confiável e correto, garantindo o atendimento às necessidades dos usuários.*
- **Completude:** *Capacidade do IP-core de apresentar documentação incluindo instruções e informações de referência, com conceitos auto-contidos, facilitando o entendimento dos usuários.*
- **Analisabilidade:** *Disponibilização de informações necessárias para a categorização e avaliação do IP-core, seja por usuários, ou mesmo por terceiros.*
- **Compatibilidade:** *Capacidade do IP-core de identificar as ferramentas EDA utilizadas durante o seu fluxo de desenvolvimento, ou mesmo compatíveis e ainda os padrões de fluxos e metodologias utilizados para desenvolvimento do IP-core.*
- **Conformidade Relacionada á Documentação***: O IP-core deve aderir a padrões*

de mercado aplicáveis e recomendados.

Suporte ao Usuário: *Devem existir mecanismo e infra-estrutura capazes de reduzir a curva de aprendizado do usuário sobre o IP-core e sua utilização, de acordo com as restrições especificadas na documentação deste.*

Além dos atributos de qualidade, o ipQUALITY define um conjunto de artefatos que devem ser entregues ao usuário do IP-core. Este conjunto foi estabelecido a partir da realização de uma comparação dos artefatos propostos por organismos padronizadores da área, o VSIA, o RMM e o QIP [26], para serem entregues aos usuários do IP-core. A Tabela 3.2 apresenta um resumo desses artefatos juntamente com o seu grau de importância na especificação do ipQUALITY, apresentado na última coluna da tabela. Esse grau de importância de cada artefato foi calculado de acordo com a quantidade de ocorrências de obrigatoriedade, recomendação ou opcionalidade indicadas pelas colunas VSIA, QIP, RMM e ipPROCESS, como mostrado na Tabela 3.1.

Na Tabela 3.2, o símbolo ☺☺☺ significa que o artefato é obrigatório de acordo com o respectivo padrão ou processo. A representação ☺☺ indica que o artefato é recomendado segundo o respectivo padrão ou processo, por sua vez, o símbolo ☺ refere-se á opcionalidade do artefato.

Anexo à este documento encontra-se uma tabela com o conjunto completo dos artefatos propostos pelo ipQUALITY para serem entregues aos usuários do IP-core.

Tabela 3.1: Distribuição do grau de importância dos artefatos da especificação ipQUALITY em função das indicações do VSIA, QIP, RMM e ipPROCESS.

Somatórios de ☺s nas colunas VSIA, QIP, RMM e ipPROCESS	Grau de importância na coluna Especificação ipQUALITY
0 - 2 ☺s	-
3 - 5 ☺s	☺
6 - 7 ☺s	☺☺
8 - 10 ☺s	☺☺☺
11 - 12 ☺s	☺☺☺☺

Tabela 3.2. Especificação dos Artefatos do Usuário propostos pelo ipQUALITY.

Artefatos	VSIA	QIP	RMM	**ipPROCESS**	Especificação **ipQUALITY**
1. Artefatos Comuns	☺	☺☺	☺☺☺	☺	☺☺☺
2. Documentação Geral	☺☺☺	☺☺☺	☺☺☺	☺☺☺	☺☺☺
2.1 Descrição Lógica	☺☺☺	☺☺☺	☺☺☺	☺☺	☺☺
2.2 Integração	☺☺☺	☺☺	☺☺☺		☺☺
3. Manual de referência do programador		☺☺☺	☺☺☺		☺
4. Implementação	☺☺☺	☺☺	☺	☺	☺☺
5. Verificação	☺☺☺	☺☺	☺	☺☺	☺☺☺
6. Distribuição	☺☺☺	☺☺☺	☺	☺	☺☺

Com o intuito de atender aos atributos de qualidade do ipQUALITY apresentados anteriormente, a disciplina *Deployment* apresentada neste trabalho define um fluxo de atividades que gera os artefatos propostos pelo ipQUALITY, visto que estes artefatos representam os produtos definidos pelos maiores organismos padronizadores da área de IP-cores para serem fornecidos aos seus usuários.

4. Trabalho Proposto

Conforme foi visto no capítulo anterior, apesar do ipPROCESS estabelecer uma série de atividades e artefatos com o objetivo de desenvolver IP-core de qualidade, de acordo com o que foi estabelecido com o cliente, ele não provê atividades que garantam que o IP-core será disponibilizado aos seus usuários finais. Ou seja, atualmente não há no ipPROCESS atividades que tratem do fornecimento do IP-core a seus usuários e também não existe a definição de artefatos que devem ser distribuídos juntamente com o IP-core com o intuito de fornecer as informações necessárias à sua utilização. Por sua vez, o ipQUALITY estabelece além de atributos de qualidade para IP-cores, um conjunto de artefatos que devem ser disponibilizados aos usuários do IP-core, com o objetivo de facilitar o reuso deste. Sendo assim, pode-se perceber a necessidade da utilização do ipQUALITY no contexto do ipPROCESS a fim de tratar da distribuição do IP-core aos seus usuários. Esta utilização se dará através da definição de um fluxo de atividades que produza, durante o desenvolvimento do IP-core, os artefatos propostos pelo ipQUALITY e que promova o fornecimento do IP-core a seus usuários finais. Esse fluxo de atividades e artefatos propostos estarão reunidos em uma nova disciplina do ipPROCESS, a disciplina Deployment.

A disciplina Deployment tem como objetivos produzir, reunir e fornecer os artefatos e informações necessárias para:

- Facilitar a escolha do IP-core;
- Diminuir o esforço de integração do VC em um SoC.

A figura abaixo apresenta a relação da disciplina Deployment com as demais disciplinas do ipPROCESS durante o desenvolvimento do IP-core. Como pode ser visto, essa disciplina é executada concorrentemente às demais disciplinas.

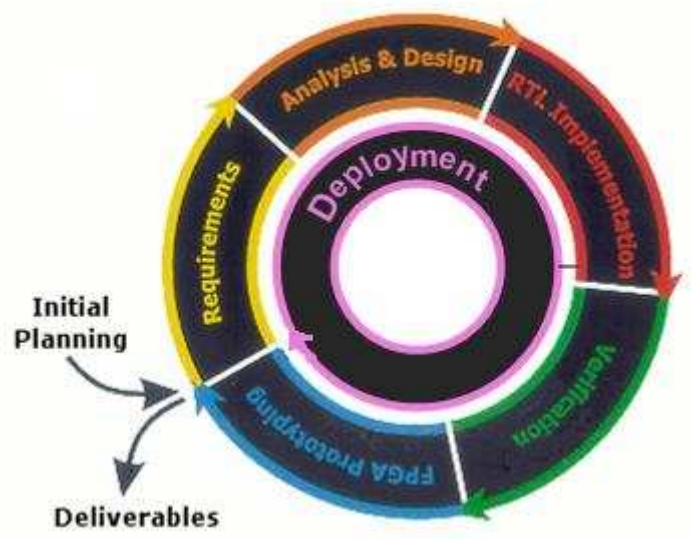

Figura 4.1: Relacionamento da disciplina *Deployment* com as demais disciplinas do ipPROCESS.

Ao longo deste capítulo será apresentada a disciplina Deployment, explicando em detalhes seu fluxo de trabalho, suas atividades e artefatos, assim como seus papéis. A Figura 4.2 apresenta a modelagem em SPEM [26] dessa disciplina através de um diagrama de pacote.

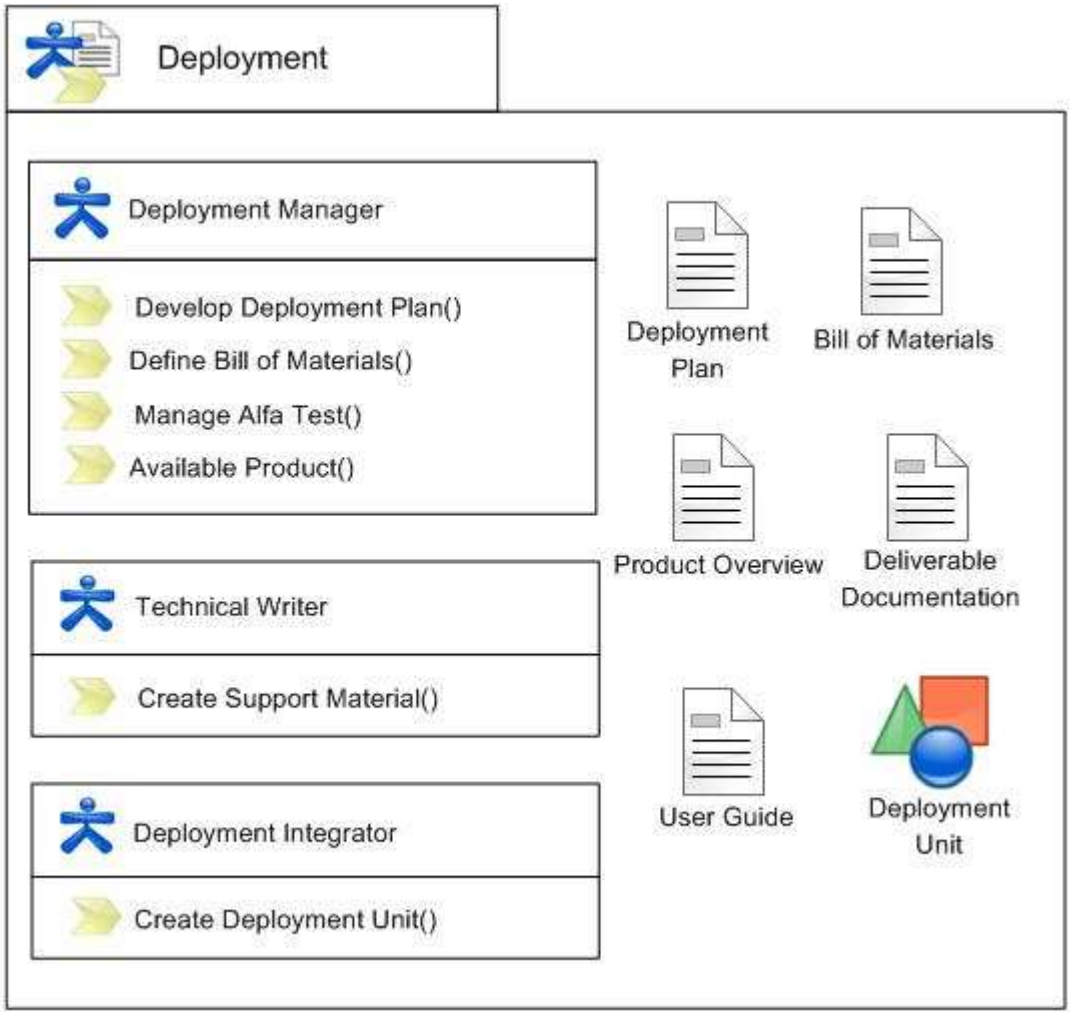

Figura 4.2: Visão Geral da Disciplina Distribuição.

Através deste diagrama de pacote podemos identificar a estrutura da disciplina:

- Artefatos produzidos: Deployment Plan, Bill of Materials, Product Overview, Deliverable Documentation, User Guide e Deployment Unit.
- Papéis envolvidos: Deployment Manager, Technical Writer, Deployment Integrator.
- Atividades do papel Deployment Manager: Develop Deployment Plan, Define Bill of Materials, Manage Alfa Test e Available Product.
- Atividade do papel Technical Writer: Create Support Material.
- Atividade do papel Deployment Integrator: Create Deployment Unit.

Para a realização de seus objetivos, a disciplina *Deployment* apresenta o fluxo de trabalho mostrado na Figura 4.3.

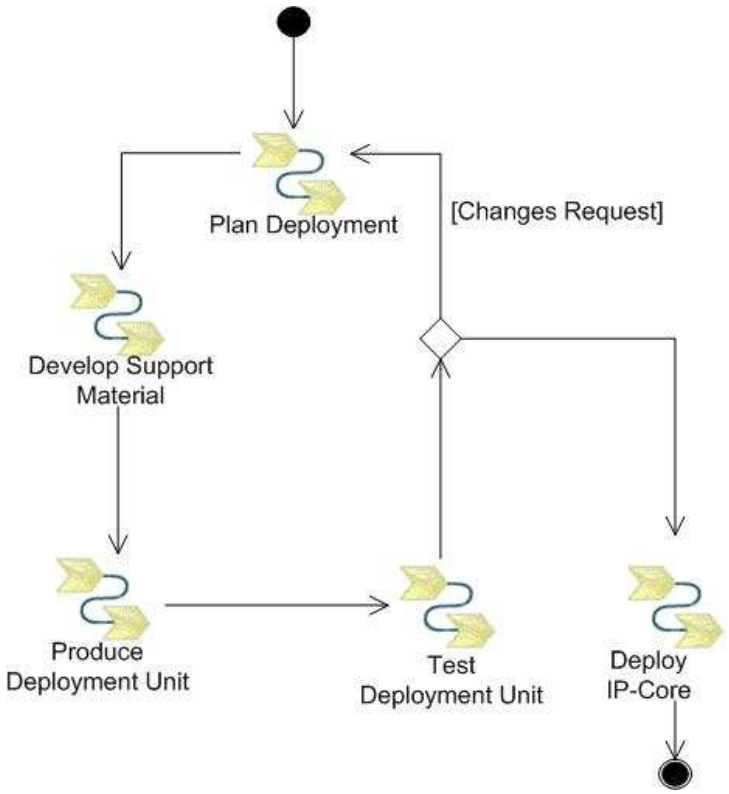

Figura 4.3: Fluxo de trabalho da Disciplina *Deployment*.

Observando o fluxo de trabalho da disciplina *Deployment*, notamos que o início da distribuição do IP-core se dá através de seu planejamento, as atividades relacionadas a esse planejamento estão reunidas no grupo ***Plan Deployment***. Após a realização do planejamento da distribuição, essa disciplina propõe que sejam desenvolvidos materiais que serão fornecidos aos usuários do IP-core contendo as informações necessárias para uma correta escolha e reuso destes, isto se dá através do grupo ***Develop Support Material.*** Uma vez prontos todos os artefatos que serão distribuídos, estes devem ser reunidos em uma unidade de distribuição, isto é realizado através das atividades do grupo ***Produce Deployment Unit.*** Depois de produzida a unidade de distribuição, deve-se testá-la a fim de garantir que os artefatos contidos nesta unidade são suficientes para uma correta utilização do IP-core, este teste se dá através do grupo ***Test Deployment Unit.*** Se nessa etapa for identificado que a unidade de distribuição não fornece os dados e materiais necessários para um correto reuso do IP-core, deve-se voltar ao início do fluxo de trabalho. Caso contrário, a unidade de distribuição pode ser disponibilizada aos usuários, isso se dá através do grupo ***Deploy IP-core.***

Cada um dos grupos de atividades que formam o fluxo de trabalho da disciplina *Deployment* será explicado a seguir, porém, antes disso serão explicados os papéis envolvidos nesta disciplina.

- *Deployment Manager:* Responsável pelo planejamento e gerência das atividades relacionadas com a distribuição do IP-core. É necessário que esse possua como habilidades a Comunicação e Coordenação, a fim de se manter atualizado sobre o status do desenvolvimento do produto e comunicar as necessidades das atividades de distribuição para os demais

membros da organização, Capacidade de Planejamento, para assegurar que a distribuição do IP-core seja feita dentro do prazo estabelecido e com os recursos disponíveis, e Orientado por Metas e Proativo, para planejar e orientar a conclusão do produto pelas várias equipes.

- *Technical Writer:* É o responsável pela elaboração da documentação de suporte para os usuários finais. Como habilidade, o *Technical Writer* necessita ter experiência ou treinamento em escrita técnica, além de ter boas habilidades de comunicação, pois pode ser preciso que ele entreviste membros da equipe desenvolvedora a fim de elaborar uma documentação correta e útil.
- *Deployment Integrator:* É equivalente ao papel de Gerente de Configuração do RUP. No contexto da disciplina *Deployment*, tem como responsabilidade a garantia do controle de revisões e mudanças dos artefatos que compõem o produto. Deve ser hábil em princípios de gerenciamento de configuração e, preferencialmente, ter experiência ou treinamento no uso de ferramentas de Gerenciamento de Configuração.

4.1. GRUPO PLAN DEPLOYMENT

Baseando-se na disciplina de Implantação do RUP e na fase de Planejamento e Especificação do RMM, este grupo contém as atividades responsáveis pela realização do planejamento da distribuição do IP-core. Para planejar sua distribuição, é preciso considerar como e quando o IP-core estará disponível ao seu usuário final, além disso, é necessário especificar os artefatos que formarão o produto que será distribuído e definir os recursos necessários à essa distribuição.

A Figura 4.4 mostra o fluxo de atividades deste grupo.

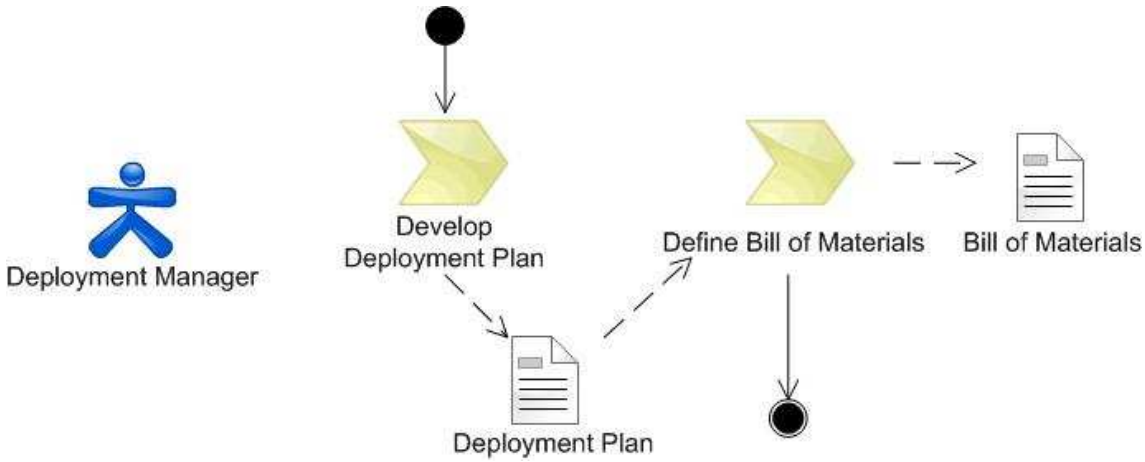

Figura 4.4: Fluxo de atividades do grupo *Plan Deployment*.

Como podemos observar através da figura acima, o início do fluxo de atividades do grupo *Plan Deployment* se dá com a execução da atividade *Develop Deployment*

Plan, responsável pelo planejamento da distribuição do IP-core. Como resultado dessa atividade tem-se o artefato *Deployment Plan*, onde estará descrito o planejamento definido na atividade passada. Por último, é executada a atividade *Define Bill of Materials*, nesta atividade são listados todos os materiais que formarão o produto que será distribuído aos usuários, o resultado desta atividade é o documento *Bill of Materials* que consiste em uma listagem dos artefatos que formarão o produto distribuído, acrescentado das mudanças ocorridas na atual versão do IP-core, além da descrição de erros e problemas identificados. O *Deployment Manager* é o papel responsável pela realização das atividades e elaboração dos artefatos desse grupo.

Tanto as atividades quanto os artefatos deste grupo são provenientes do RUP, pois assim como para projetos de software, estes também podem ser aplicados no contexto de distribuição de IP-cores.

As tabelas a seguir detalham as atividades e os artefatos ilustrados na Figura 4.4.

Tabela 4.1. Detalhamento da atividade *Develop Deployment Plan*.

Atividade: Develop Deployment Plan
Objetivo: Planejar a distribuição do IP-core, definido como e quando o IP-core será distribuído e identificando os recursos necessário à essa distribuição.
Passos: 1. Definir como e quando o IP-core será distribuído; 2. Listar as atividades necessárias para a distribuição assim como seus responsáveis; 3. Elaborar um cronograma para as atividades listadas; 4. Definir como será realizada a assistência e treinamento aos usuários, caso haja necessidade; 5. Listar os recursos necessários para a distribuição do IP-core.
Artefatos de Entrada: Informações acerca do plano de desenvolvimento do IP-core, assim também como informações sobre o plano da iteração atual, além disso, são necessárias informações acerca de critérios de aceitação do produto por parte do cliente.
Artefatos de Saída: *Deployment Plan.*
Responsável: *Deployment Manager.*

Tabela 4.2. Detalhamento da Atividade *Define Bill of Materials*.

Atividade: Define Bill of Materials
Objetivo: Criar uma lista com os artefatos que compõem o produto que será distribuído. As mudanças nas versões dos artefatos são descritas assim como os erros e problemas encontrados na atual versão do produto.
Passos: 1. Listar todos os itens liberados; 2. Manter a Lista de Materiais.
Artefatos de Entrada: *Deployment Plan,* além de informações acerca do plano de desenvolvimento do IP-core, informações sobre o plano da iteração atual, como também informações acerca de critérios de aceitação do produto por parte do cliente.
Artefatos de Saída: *Bill of Materials.*
Responsável: *Deployment Manager.*

Tabela 4.3 Detalhamento do Artefato Deployment Plan.

Artefato: Deployment Plan
Descrição: Contém a descrição do planejamento da distribuição do IP-core, considerando como e quando o IP-core será distribuído, como também as atividades e os recursos necessários para que isto ocorra.
Entrada para Atividades: *Create Deployment Unit, Manage Alfa Test* e *Available product.*
Saída de Atividades: *Develop Deployment Plan.*
Responsável: *Deployment Manager.*

Tabela 4.4 Detalhamento do Artefato Bill of Materials.

Artefato: Bill of Materials
Descrição: Lista as partes que formam determinada versão do produto, como também descreve as mudanças efetuadas na versão e os *bugs* existentes.
Entrada para Atividades: *Create Deployment Unit.*
Saída de Atividades: *Define Bill of Materials.*
Responsável: *Deployment Manager.*

Como podem ser observados, os campos referentes aos artefatos de entrada das atividades *Plan Deployment* e *Define Bill of Materials* não contém apenas nomes de artefatos, mas também uma descrição das informações necessárias à execução destas atividades. Isso ocorre porque o ipPROCESS não possui artefatos que contenham esse tipo de informação. Diante disto, para a execução deste trabalho, essas informações serão assumidas como premissas e sua geração no ipPROCESS pode ser alvo de trabalhos futuros.

Sobre o artefato *Bill of Materials*, é necessário que este seja atualizado a cada iteração, para que com isso possa ser garantido que o projeto tenha uma lista sempre atualizada dos artefatos que compõem o *build* do produto.

4.2. GRUPO DEVELOP SUPPORT MATERIAL

Como já foi discutido neste trabalho, o fornecimento de uma documentação que contenha informações acerca da funcionalidade, arquitetura, restrições, entre outros aspectos do IP-core é fundamental no processo de sua escolha e reuso. Este detalhamento do fluxo de trabalho tem o objetivo de produzir documentos que forneçam essas informações aos usuários do IP-core.

A norma NBR ISO/IEC 12119 - Tecnologia da Informação – Pacotes de software – Teste e requisitos de qualidade estabelece que cada pacote de software deve ter os seguintes itens:

- **Descrição do Produto:** Fornece informações acerca da documentação do usuário, programas e se existirem, sobre os dados.
- **Documentação do Usuário:** Deve descrever todas as funções do programa estabelecidas na descrição do produto e as que os usuários tenham acesso.
- **Guias de Teste:** Para a identificação do correto funcionamento do produto.

Também é estabelecido pela norma NBR ISO/IEC 12119 que a descrição do produto contenha a identificação da descrição do produto, identificação do produto e seu fornecedor, descrição das tarefas executadas pelo produto, sua conformidade a documentos de requisitos, seus requisitos de hardware e software, descrição de sua interface com outros produtos, uma listagem dos itens a serem entregues, como também informações sobre sua instalação, suporte e manutenção.

Adaptando a norma NBR ISO/IEC 12119 para o contexto da distribuição de IP-cores, a disciplina *Deployment* estabelece que sejam entregues aos usuários os seguintes documentos:

- ***Product Overview*:** Documento expondo as propriedades do IP-core, com o principal objetivo de auxiliar os potenciais usuários da adequação do produto antes de sua aquisição.
- ***User Guide*:** Conjunto completo de informações necessárias para a utilização do IP-core. Esse documento engloba informações detalhadas

sobre a funcionalidade, arquitetura, implementação, verificação e integração do IP-core.

- ***Deliverable Documentation:*** Lista todos os itens que formam o produto distribuído, suas versões, problemas e *bugs* conhecidos, além de termos e condições de uso.

Por sua vez, o padrão IEEE 1063 – Padrão para documentação do usuário de um determinado software estabelece requisitos mínimos para documentação de usuários do software, documentação tanto impressa como eletrônica. Sua especificação está voltada principalmente na definição da estrutura e formatos do conteúdo da documentação do usuário, sendo a maior parte das recomendações voltadas à estruturação do documento. Os documentos gerados pela disciplina *Deployment* e que serão fornecidos aos usuários do IP-core seguirão as seguintes orientações desse padrão relacionadas à estruturação:

- Os documentos devem possuir uma estrutura uniforme, ou seja, funcionalidades relacionadas devem estar agrupadas em uma mesma seção ou capítulo, das mais simples às mais complexas;
- Não deve haver redundância em uma documentação bem estruturada.

Em relação ao formato do conteúdo da informação da documentação proposto pelo padrão IEEE 1063, este preza pela completude e acurácia dos documentos. Segundo esse padrão, a completude de um documento se dá na apresentação tanto de instruções como de informações de referência. Instruções são vistas como informações de como utilizar o software, já informações de referência são todas as instâncias de elementos sob documentação, como o conjunto de comandos válidos, por exemplo. Já em relação à acurácia, a documentação deve refletir fielmente as características do produto distribuído. Tanto a completude quanto a acurácia são atributos de qualidade já tratados pelo ipQUALITY, e assim como o restante de sua especificação devem ser atendidos na elaboração dessa documentação.

A fim de diminuir o esforço necessário para produzir esses documentos, as informações contidas neles devem ser documentadas incrementalmente durante todo o desenvolvimento do IP-core, para que isto ocorra são necessárias mudanças em algumas disciplinas do ipPROCESS. Na seção abaixo serão apresentadas as mudanças propostas.

4.2.1. Mudanças Propostas para o ipPROCESS

Para que o esforço do desenvolvimento da documentação que será distribuída aos usuários do IP-core não esteja concentrada em uma única atividade, este trabalho propõe que as informações que serão distribuídas sejam documentadas ao longo de todo o desenvolvimento do IP-core, para isso mudanças são propostas nas seguintes disciplinas do ipPROCESS:

- ***Analysis & Design:*** Como vimos no capítulo anterior, tem-se como resultado da disciplina *Analysis & Design* a definição de uma arquitetura e projeto para o IP-core, onde são definidos os módulos que o compõem,

suas interfaces e protocolos de comunicação. Durante a definição do *design* do IP-core também podem ser identificados os registradores que o compõem, como também a definição das instruções necessárias à programação do mesmo. Este trabalho propõe que além dos artefatos já produzidos por esta disciplina, que como já foi visto, descrevem a arquitetura definida, interfaces, protocolos e padrões de projetos aplicáveis, seja produzido mais um documento, o *Programmer's Reference Manual*. Esse documento deve fornecer informações sobre os registradores programáveis do IP-core e detalhar o conjunto de instruções disponíveis para programá-lo. A figura abaixo apresenta o artefato *Programmer's Reference Manual* sendo gerado pela atividade *Design Components.*

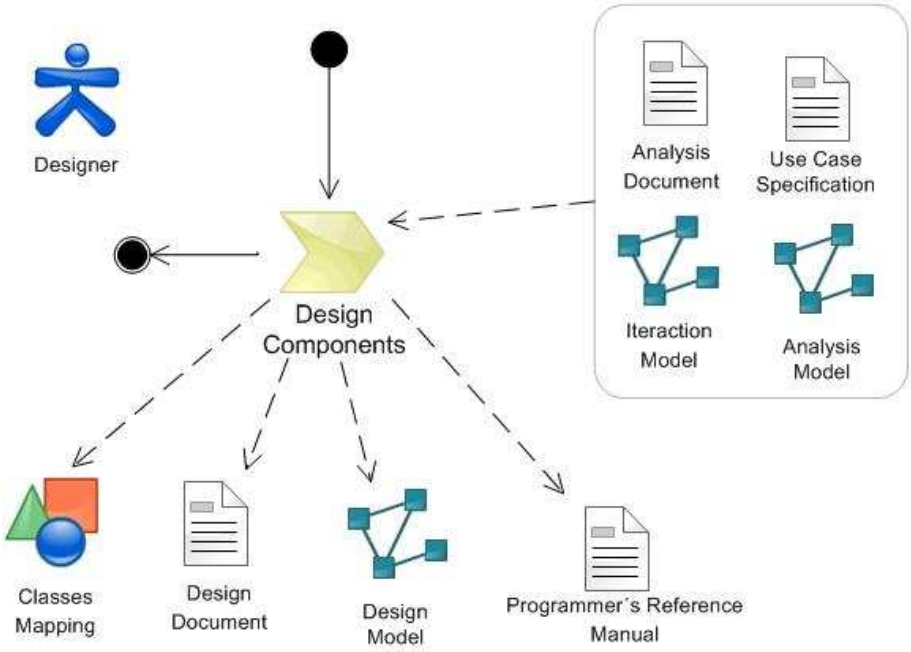

Figura 4.5. Modificação proposta para a disciplina *Analysis & Design*

O artefato proposto para esta disciplina, o *Programmer's Reference Manual*, é detalhado na tabela abaixo.

Tabela 4.5. Detalhamento do artefato Programmer's Reference Manual.

Artefato: Programmer´s Reference Manual
Descrição: Descreve um conjunto de instruções, informações sobre os registradores e de como programar o dispositivo.
Entrada para Atividades: *Develop Support Materials.*
Saída de Atividades: *Design Components.*
Responsável: *Designer.*

- ***FPGA Prototyping:*** As modificações propostas para a disciplina *FPGA Prototyping* vão além da criação de novos artefatos. Um grupo de atividades, o *Create Implementation Deployment*, é adicionado ao fluxo de atividades desta disciplina com o objetivo de elaborar para os usuários um conjunto de artefatos que proporcionem uma integração satisfatória do IP-core em um projeto de SoC. Como pode ser observado através das descrições das atividades dessa disciplina feitas no capítulo anterior, o fluxo de atividades da disciplina obedece à seguinte ordem: planejamento dos módulos que serão prototipados, seguido de síntese dos componentes, após isso os módulos são cossimulados e por último são testados, caso haja mais componentes a serem prototipados, o ciclo inicia-se novamente, caso contrário, a disciplina é finalizada. Este trabalho propõe que caso não haja mais componentes a serem prototipados e se foi planejado que no final da iteração vigente um *release* será entregue ao usuário, um ambiente de distribuição do IP-core prototipado seja elaborado. Esse ambiente é composto por artefatos que descrevem para o usuário o ambiente utilizado na prototipação do IP-core pela equipe de desenvolvimento, além da documentação da mensuração de atributos do protótipo. Esses artefatos são compostos por *Implementation Description* - documentação de certos atributos do IP-core como potência, área, freqüência, entre outros, - *Prototyping Manual*- manual contendo informações acerca do ambiente necessário para repetir a prototipação no ambiente do usuário, e *Prototyping Environment*- composto por *scripts* de síntese e de cossimulação, além dos arquivos binários da prototipação. Além da criação do novo grupo de atividades, este trabalho também propõe mudanças no documento *FPGA Prototyping Plan.* Este, que atualmente apresenta informações a cerca do planejamento de quais módulos serão prototipados em cada iteração, deverá também identificar se o resultado da prototipação formará um *release* a ser entregue ao usuário.

  A Figura 4.6 apresenta o novo fluxo de atividades da disciplina *FPGA Prototyping.*

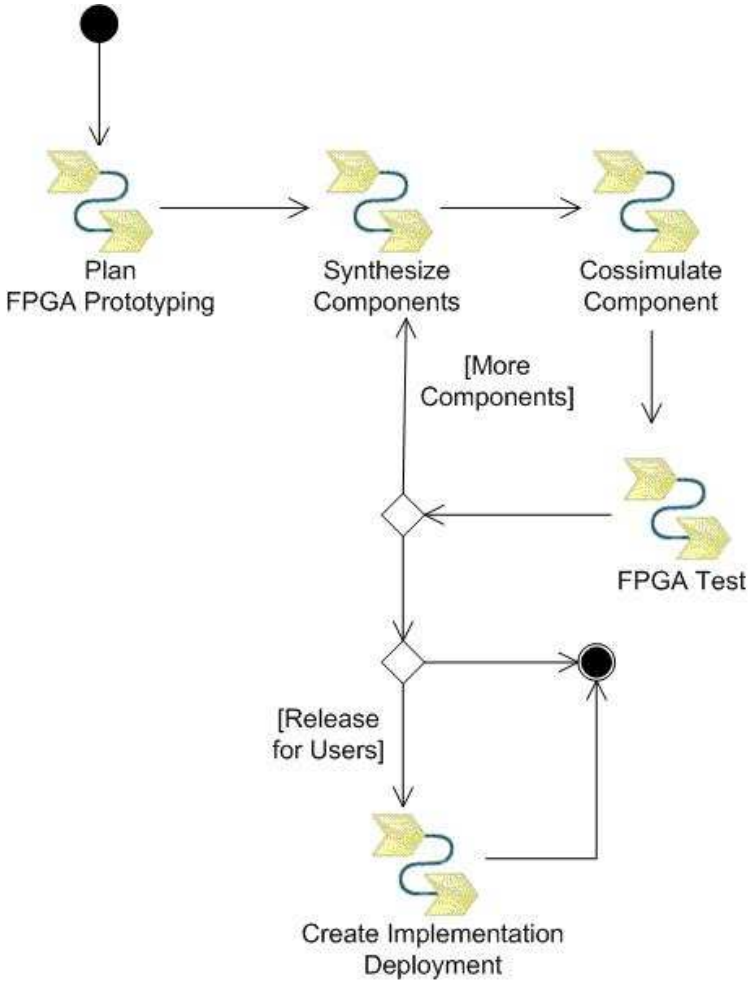

Figura 4.6. Novo fluxo de atividades da disciplina *FPGA Prototyping*.

A Figura 4.7 apresenta o detalhamento das atividades que compõem o grupo *Create Implementation Deployment*. A atividade *Create Prototyping Environment* é responsável pela elaboração do ambiente de prototipação que será distribuído aos usuários, assim como pela geração de um manual explicando o funcionamento desse ambiente. Por sua vez, a atividade *Gather Implementation Information* reúne informações acerca do IP-core prototipado e as documenta no artefato *Implementation Descripton.*

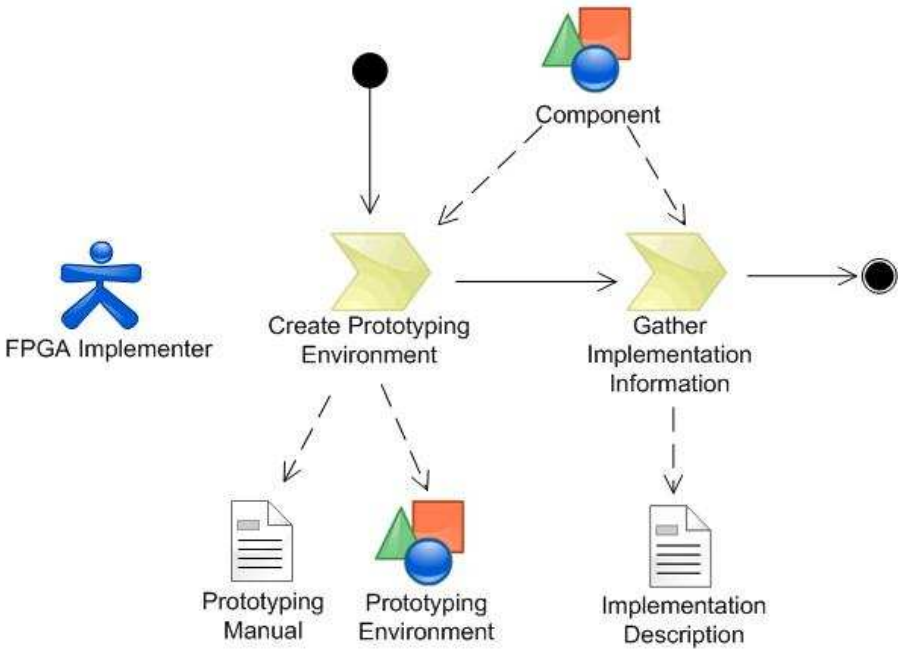

Figura 4.7. Detalhamento do grupo Create Implementation Deployment.

As atividades e artefatos ilustrados na Figura 4.7 encontram-se descritas nas tabelas seguintes.

Tabela 4.6. Detalhamento da atividade Create Prototyping Environment.

Atividade: Create Prototyping Environment
Objetivo: Desenvolver um conjunto de artefatos para o usuário do IP-core que permita com que este possa configurar seu ambiente de acordo com o ambiente de prototipação usado no desenvolvimento do IP-core. Este conjunto é composto por *scripts*, documentação do fluxo de ferramentas e metodologias utilizadas.
Passos: 1. Desenvolver *scripts* de síntese e cossimulação; 2. Documentar o fluxo de ferramentas e a metodologia utilizada.
Artefatos de Entrada: *FPGA Prototyping Plan e Component.*
Artefatos de Saída: *Prototyping Manual e Prototyping Environment.*
Responsável: *FPGA Implementer.*

Tabela 4.7. Detalhamento da atividade *Gather Implementation Information*.

Atividade: Gather Implementation Information
Objetivo: Mensurar e documentar determinados atributos do IP-core prototipado, como área, potência, latência, entre outros.
Passos: 1. Listar os atributos do IP-core que devem ser mensurados; 2. Obter o IP-core prototipado e verificado; 3. Mensurar os atributos listados; 4. Documentar os resultados da medição.
Artefatos de Entrada: *Component.*
Artefatos de Saída: *Implementation Description.*
Responsável: *FPGA Integrator.*

Tabela 4.8. Detalhamento do artefato *Prototyping Manual*.

Artefato: Prototyping Manual
Descrição: Apresenta informações acerca do ambiente necessário para repetir a prototipação do IP-core no ambiente do usuário.
Entrada para Atividades: *Develop Support Materials.*
Saída de Atividades: *Create Prototyping Environment.*
Responsável: *FPGA Implementer.*

Tabela 4.9. Detalhamento do artefato Prototyping Environment.

Artefato: Prototyping Environment
Descrição: Composto por *scripts* de síntese e de cossimulação, além dos arquivos binários da prototipação. Possibilita ao usuário configurar seu ambiente de acordo com o ambiente de prototipação usado no desenvolvimento do IP-core.
Entrada para Atividades: *Create Deployment Unit.*
Saída de Atividades: *Create Prototyping Environment.*
Responsável: *FPGA Implementer.*

Tabela 4.10. Detalhamento do artefato Implementation Description.

Artefato: Implementation Description
Objetivo: Descreve os resultados da mensuração de certos atributos do IP-core, como potência, freqüência, área, entre outros, além de descrever o nível de abstração da implementação do IP-core e apresentar seus diagramas de tempo.
Entrada para Atividades: *Develop Support Materials.*
Saída de Atividades: *Gather Implementation Informations.*
Responsável: *FPGA Integrator.*

- ***Verification:*** Através das descrições das atividades da disciplina *Verification*, pode-se observar que o fluxo de atividades desta disciplina compreende o seguinte esquema: primeiro, define-se o escopo da verificação a ser realizada no IP-core, em seguida, constrói-se o ambiente necessário à verificação, depois a verificação é executada, caso os testes não tenham sido suficientes para uma completa verificação do IP-core volta-se ao início do fluxo de atividades, caso contrário, a disciplina é finalizada. As mudanças propostas nesta disciplina são o acréscimo de uma atividade, a elaboração de um novo artefato e mudança em outro. A atividade proposta, a *Gather Verification Components*, forma um novo grupo no fluxo de atividades, o grupo *Create Verification Deployment*. A atividade *Gather Verification Components* é responsável por reunir os *scripts* e componentes de verificação utilizados, em uma unidade que proporcione aos usuários um ambiente para a verificação do IP-core. A Figura 4.8 apresenta o novo fluxo de atividades da disciplina *Verification*. A atividade que compõe o grupo *Create Verification Environment* é apresentada na Figura 4.9. O artefato onde é proposta uma mudança é o documento *Verification Plan*, é necessário que além de definir a abordagem que será utilizada na verificação, as atividades e recursos necessários, conforme consta na sua definição no capítulo anterior, ele defina se será utilizada alguma biblioteca de referência na verificação do IP-core, e se caso for, explicitar qual biblioteca de referência. Bibliotecas de referência estão entre os itens definidos pelo ipQUALITY para serem fornecidos aos usuários do IP-core, porém em nenhum ponto do atual fluxo de atividades da disciplina *Verification* é tratado qual biblioteca de referência será utilizada.

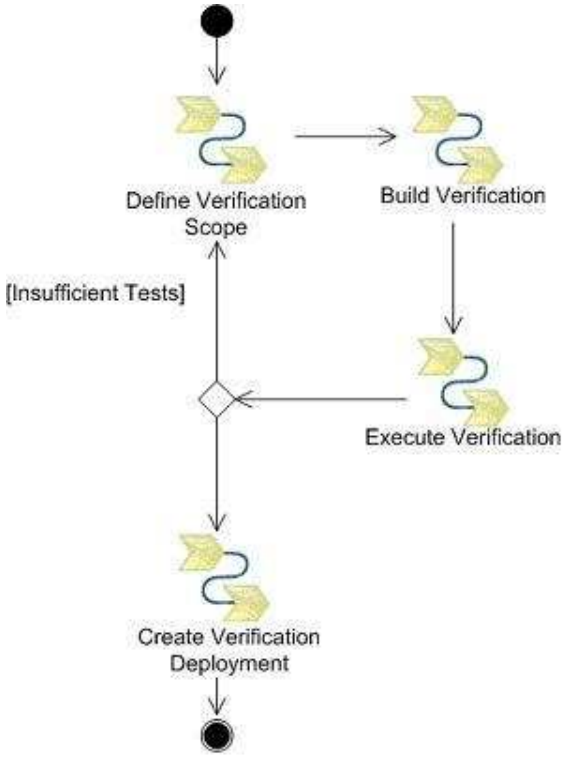

Figura 4.8. Fluxo proposto para a disciplina *Verification*.

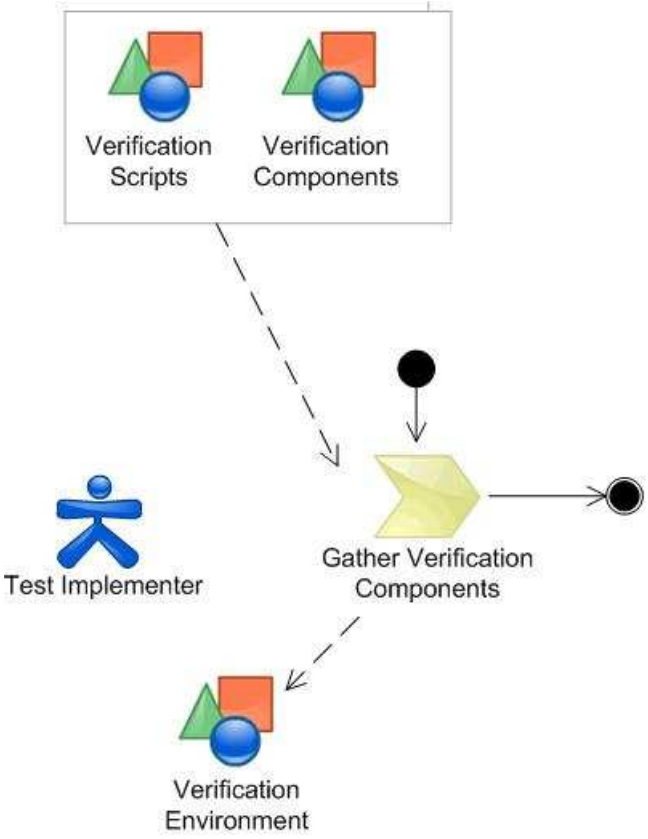

Figura 4.9. Grupo *Create Verification Deployment*.

As tabelas abaixo detalham a atividade e o artefato propostos para essa disciplina.

Tabela 4.11. Detalhamento da atividade *Gather Verification Components.*

Atividade: Gather Verification Components.
Objetivo: Disponibilizar um ambiente que proporcione ao usuário verificar o IP-core.
Passos: 1. Reunir os casos de teste utilizados na verificação do IP-core; 2. Reunir os componentes de verificação; 3. Reunir os scripts de verificação; 4. Criar uma unidade de verificação englobando os componentes e scripts de verificação, além dos casos de teste.
Artefatos de Entrada: *Verification Scripts* e *Verification Components.*
Artefatos de Saída: *Verification Environment.*
Responsável: *Test Implementer.*

Tabela 4.12. Detalhamento do artefato *Verification Environment*

Artefato: Verification Environment.
Descrição: Apresenta o ambiente necessário para a realização da verificação do IP-core por seus usuários. É composto por casos de teste, componentes e *scripts* de verificação.
Entrada para Atividades: *Create Deployment Unit.*
Saída de Atividades: *Gather Verification Components.*
Responsável: *Test Implementer.*

Com as modificações propostas para o ipPROCESS descritas anteriormente, este terá um conjunto de novos artefatos, dentre estes artefatos, toda documentação agora produzida se reunirá com outros documentos gerados pelo ipPROCESS para formar dois dos documentos que serão entregues aos usuários, o *Product Description* e o *User Guide*. A Figura 4.10 dá uma visão geral do grupo *Develop Support Material*, sendo que suas atividades e artefatos são detalhados nas próximas tabelas.

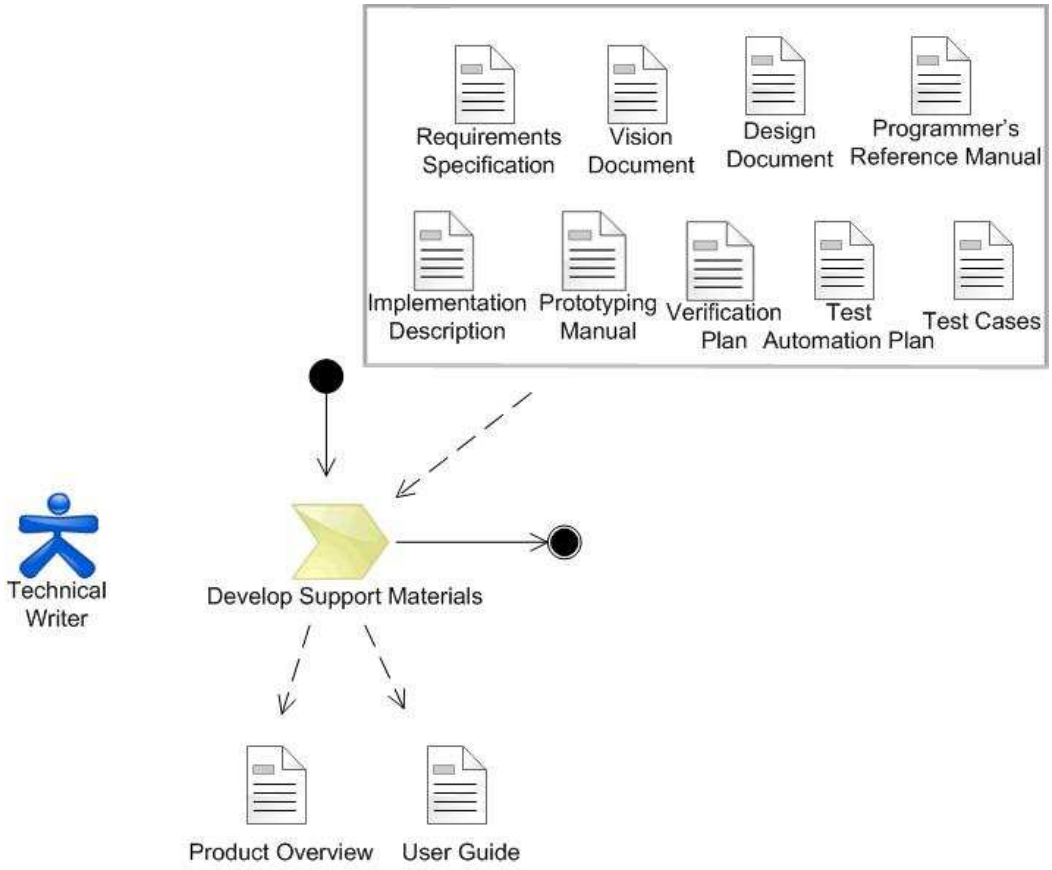

Figura 4.10. Visão geral do grupo Develop Support Material.

Tabela 4.13. Detalhamento da atividade *Develop Support Materials*.

Atividade: Develop Support Materials
Objetivo: Desenvolver o material de suporte para o usuário.
Passos: 1. Reunir certos artefatos elaborados ao longo do desenvolvimento do IP-core (*Requirements Specification*, *Vision Document*, *Design Document*, *Programmer´s Reference Manual*, *Implementation Description*, *Prototyping Manual*, *Verification Plan*, *Test Automation Plan* e *Test Cases*); 2. Dividir as informações dos artefatos reunidos entre os artefatos *Product Description* e *User Guide.*
Artefatos de Entrada: *Requirements Specification*, *Vision Document*, *Design Document*, *Programmer´s Reference Manual*, *Implementation Description*, *Prototyping Manual*, *Verification Plan*, *Test Automation Plan* e *Test Cases.*
Artefatos de Saída: *Product Overview* e *User Guide.*
Responsável: *Technical Writer.*

Tabela 4.14. Detalhamento do artefato *Product Overview*.

Artefato: Product Overview
Descrição: Apresenta uma visão geral do IP-core, mostrando superficialmente sua funcionalidade, compatibilidade com padrões e requisitos de integração. Auxiliando assim os potenciais usuários da adequação do produto antes de sua aquisição.
Entrada para Atividades: *Create Deployment Unit.*
Saída de Atividades: *Develop Support Materials.*
Responsável: *Technical Writer.*

Tabela 4.15. Detalhamento do artefato *User Guide*.

Artefato: User Guide
Descrição: Apresenta detalhadamente todas as informações necessárias ao reuso do IP-core. Além de descrever de uma forma bem mais detalhada as informações contidas no artefato *Product Overview*, contém informações acerca da implementação, verificação, integração e restrições do IP-core.
Entrada para Atividades: *Create Deployment Unit.*
Saída de Atividades: *Develop Support Materials.*
Responsável: *Technical Writer.*

4.3. GRUPO PRODUCE DEPLOYMENT UNIT

Tendo como bases o detalhamento de fluxo de trabalho Produzir Unidade de Implantação da disciplina Implantação do RUP e a fase de Produção do RMM, a finalidade deste grupo é a criação de uma unidade de distribuição (conjunto de itens que serão entregues aos usuários do IP-core) suficientemente completa com a qual o usuário possa reusar corretamente o IP-core.

Para a criação da Unidade de Implantação é necessário ter-se um controle minucioso das versões de todos os artefatos utilizados, é preciso também a garantia de que são utilizados nesta etapa da disciplina as versões mais estáveis dos artefatos. Para que isso ocorra, é necessária uma política de controle e gerenciamento de mudanças para o desenvolvimento do IP-core. Atualmente o ipPROCESS não trata

desse aspecto, por ultrapassar o escopo deste trabalho, será assumido que os artefatos utilizados estão em suas versões mais estáveis.

Como podemos observar na Figura 4.11, o papel responsável pela criação da Unidade de Distribuição é o *Deployment Integrator*.

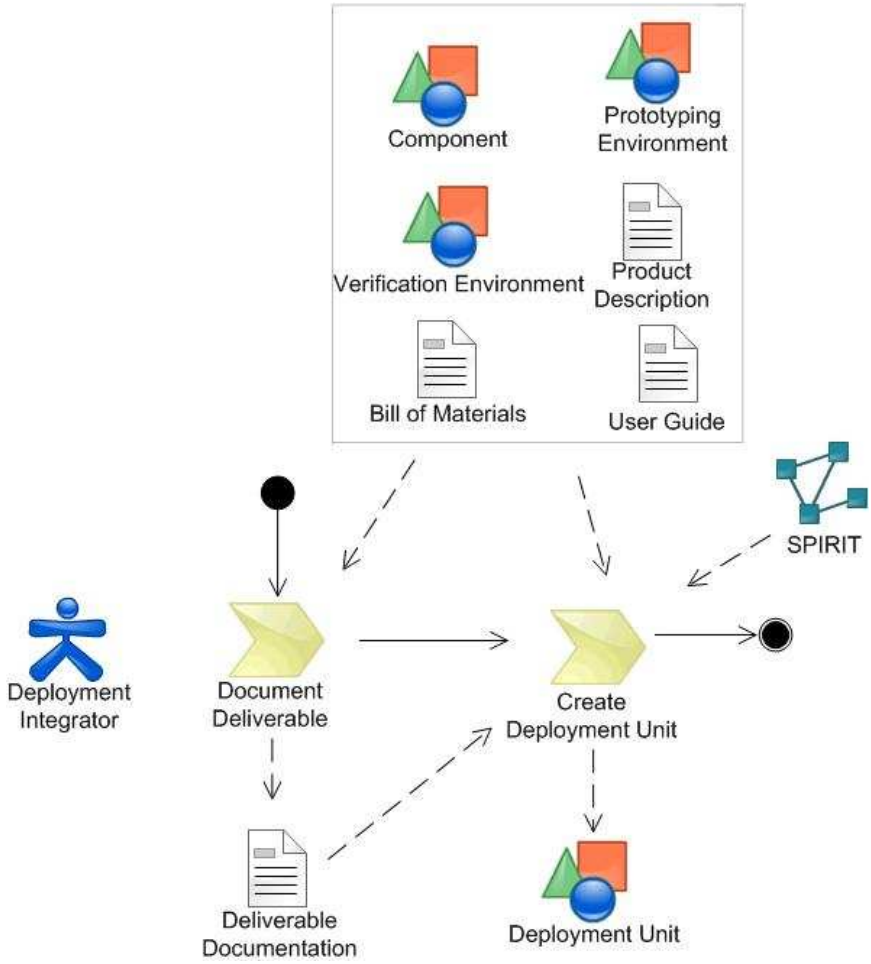

Figura 4.11. Visão geral do grupo *Produce Deployment Unit*.

Na figura acima, o grupo *Produce Deployment Unit* é formado por duas atividades, a *Document Deliverable* e a *Create Deployment Unit*. A atividade *Document Deliverable* é responsável pela documentação dos itens que formam o produto distribuído, suas versões, problemas e *bugs* conhecidos, além de termos e condições de uso. Por sua vez, a atividade *Create Deployment Unit* elabora a unidade de distribuição que será fornecida aos usuários.

O padrão de distribuição definido pelo consórcio SPIRIT, o IP-XACT, já mencionado neste documento, é utilizado aqui como um guia de empacotamento de IP-cores. Descrevendo os dados do IP-core através de um esquema XML, o IP-XACT possibilita a captura das informações de arquitetura e configuração do IP-core necessárias para integrá-lo em qualquer plataforma SoC [27]. É importante ressaltar que sendo um guia, este padrão não se encontra "amarrado" à disciplina

Deployment, podendo deste modo, a Unidade de Distribuição ser empacotada de acordo com outro padrão.

Abaixo se encontram detalhadas as atividades e artefatos deste grupo.

Tabela 4.16. Detalhamento da atividade *Document Deliverable*.

Atividade: Document Deliverable
Objetivo: Documentar todos os artefatos que irão ser distribuídos aos usuários do IP-core, definindo suas versões, problemas e *bugs* conhecidos.
Passos: 1. Listar os itens que serão distribuídos; 2. Documentar seus *bugs* e problemas; 3. Documentar as mudanças da versão atual.
Artefatos de Entrada: *Component, Prototyping Environment, Verification Environment, Product Description, Bill of Materials* e *User Guide.*
Artefatos de Saída: *Deliverable Documentation.*
Responsável: *Deployment Integrator.*

Tabela 4.17. Detalhamento da atividade *Create Deployment Unit.*

Atividade: Create Deployment Unit
Objetivo: Criar uma unidade de implantação suficientemente completa que permita o uso e verificação do IP-core.
Passos: 1. Verificar como o IP-core será distribuído; 2. Verificar quais artefatos irão compor o produto disponível aos usuário; 3. Reunir os artefatos necessários; 4. Empacotar os artefatos de acordo com algum padrão de distribuição.
Artefatos de Entrada: *Deployment Plan, Bill of Materials, RTL Code, Prototyping Environment, Verification Environment, Product Description, Bill of Materials e User Guide.*
Artefatos de Saída: *Deployment Unit.*
Responsável: *Deployment Integrator.*

Tabela 4.18. Detalhamento do artefato *Deliverable Documentation*.

Artefato: Deliverable Documentation
Descrição: Apresentar a descrição de todos os itens que formam o produto distribuído, suas versões, problemas e *bugs* conhecidos, além de termos e condições de uso.
Entrada para Atividades: *Create Deployment Unit.*
Saída de Atividades: *Document Deliverable.*
Responsável: *Deployment Integrator.*

Tabela 4.19. Detalhamento do artefato *Deployment Unit*.

Artefato: Deployment Unit
Descrição: É o produto que será distribuído aos usuários. Consiste no IP-core e um conjunto de artefatos necessários para seu reuso e verificação.
Entrada para Atividades: *Manage Beta Test e Available Product.*
Saída de Atividades: *Create Deployment Unit.*
Responsável: *Deployment Integrator.*

4.4. TEST DEPLOYMENT UNIT

Baseando-se na fase de Teste Alfa e Release do RMM, a disciplina *Deployment* estabelece que a unidade de implantação resultante da atividade *Create Deployment Unit*, explicada anteriormente, seja testada a fim de garantir que ela está completa e pronta para ser utilizada pelo *IP-Integrator*. Entende-se por completa e pronta o fato do artefato *Deployment Unit* conter todos os artefatos definidos no documento *Bill of Materials*, e que, através destes seja possível o reuso do IP-core, logo, é testado a corretude e usabilidade dos artefatos que formam o produto que será distribuído.

A Figura 4.12 dá uma visão geral deste detalhamento de fluxo de trabalho.

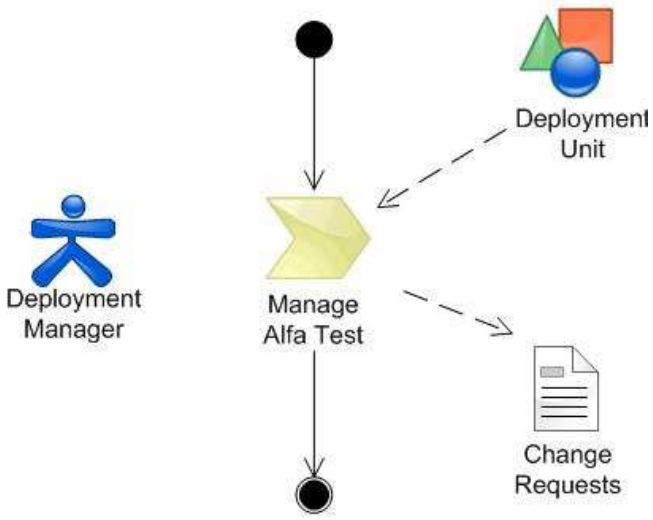

Figura 4.12. Visão geral do grupo *Test Deployment Unit*.

O *Deployment Manager* é o responsável pelo gerenciamento do teste do *Deployment Unit,* isto é realizado através da atividade *Manage Alfa Test.* É necessário que esse teste seja realizado por alguém que não pertença à equipe de desenvolvimento do IP-core, para que seu resultado possa refletir a real situação dos artefatos que compõem o *Deployment Unit*. Como resultado da atividade *Manage Alfa Test* pode-se ter o artefato *Change Requests*, este artefato descreverá as solicitações de mudanças caso tenha-se constatado algum problema com o *Deployment Unit*. Abaixo encontram-se as descrições da atividade e do artefato deste grupo.

Tabela 4.20. Detalhamento da atividade *Manage Alfa Test*.

Atividade: Manage Alfa Test
Objetivo: Gerenciar o teste alfa da Unidade de Implantação com o objetivo de verificar se este está completo e pronto para ser distribuído e repassar à equipe de desenvolvimento as mudanças necessárias, caso seja preciso.
Passos: 1. Verificar se o *Deployment Unit* está pronto para a realização do teste; 2. Selecionar a pessoa que testará o *Deployment Unit;* 3. Iniciar o teste; 4. Revisar resultados; 5. Monitorar o estabelecimento das mudanças necessárias.
Artefatos de Entrada: *Deployment Unit.*
Artefatos de Saída: *Change Requests.*
Responsável: *Deployment Manager.*

Tabela 4.21. Detalhamento do artefato Change Requests.

Artefato: Change Requests
Descrição: Documento que apresenta as mudanças requeridas ao *Deployment Unit.* Essas mudanças são resultados do teste feito no *Deployment Unit*, caso este não esteja pronto o suficiente para ser utilizado com sucesso pelos usuários do IP-core.
Entrada para Atividades: *Develop Deployment Plan.*
Saída de Atividades: *Manage Alfa Test.*
Responsável: *Deployment Manager.*

Como o ipPROCESS não possui um conjunto de atividades relacionadas ao gerenciamento de configuração e controle de mudanças, não há neste processo atividades que tratem especificamente das mudanças requeridas após a execução da atividade *Manage Alfa Test.* Para não ultrapassar o escopo deste trabalho, a disciplina *Deployment* propõe que essas mudanças sejam levadas em conta durante o planejamento da distribuição do produto da iteração posterior àquela onde foram requeridas as mudanças.

4.5. DEPLOY IP-CORE

Uma vez finalizado com sucesso o teste alfa do artefato *Deployment Unit*, ou seja, tendo verificado que este possui os artefatos e informações necessárias e suficientes para o reuso do IP-core, pode-se agora disponibilizá-lo aos seus usuários finais. A maneira como o IP-core será disponibilizado aos seus usuários varia de acordo com o projeto, podendo ser, por exemplo, através de um dispositivo de armazenamento físico ou disponibilização numa base de IP-cores. É importante ressaltar que a maneira que o IP-core será disponibilizado foi definida durante a execução da atividade *Develop Deployment Plan,* e encontra-se documentada no artefato *Deployment Plan.*

O IP-core é distribuído aos seus usuários através da atividade *Available Product*. Temos como resultado dessa atividade o artefato *Term of Acceptance*, que consiste num documento no qual o cliente expressa sua aceitação em relação ao IP-core recebido. Uma visão geral do grupo *Deploy IP-core* é apresentada na figura abaixo.

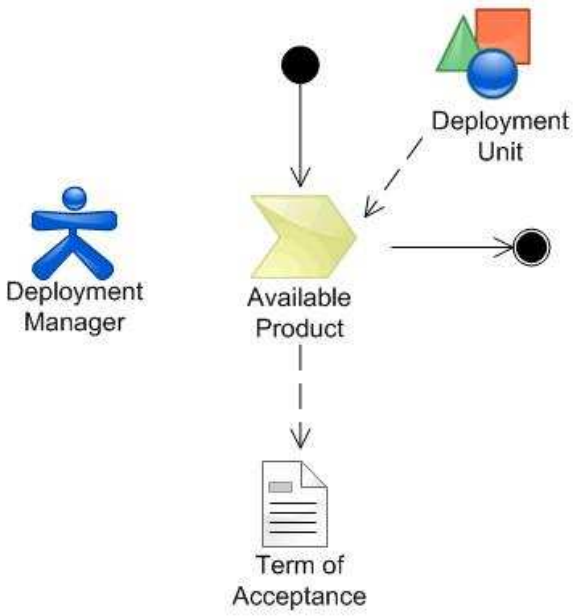

Figura 4.13. Visão geral do grupo *Deploy Product*.

Tabela 4.22. Detalhamento da atividade *Available Product*.

Atividade: Available Product
Objetivo: Disponibilizar o artefato *Deployment Unit* aos usuários.
Passos: 1. Disponibilizar o *Deployment Unit* como foi estabelecido na atividade *Develop Deployment Plan.*
Artefatos de Entrada: *Deployment Unit.*
Artefatos de Saída: *Term of Acceptance*.
Responsável: *Deployment Manager.*

Tabela 4.23. Detalhamento do artefato *Term of Acceptance*.

Artefato: Term of Acceptance
Descrição: Documento onde o cliente expressa sua aceitação em relação ao IP-core recebido.
Entrada para Atividades: *None.*
Saída de Atividades: *Available Product.*
Responsável: *Deployment Manager.*

5. Aplicando o Fluxo de Atividades – Estudo de Caso

A fim de demonstrar que a documentação proposta pela disciplina *Deployment* pode ser elaborada com informações geradas por outras disciplinas do ipPROCESS e que é necessário o acréscimo de algumas informações em alguns artefatos produzidos por essas disciplinas, utilizou-se como estudo de caso um já finalizado projeto do Brazil-IP, o LCD Controller and Driver.

A seguir será descrito o projeto LCD Controller and Driver, assim como a geração da documentação da disciplina *Deployment* a partir de seus artefatos, por fim será concluído o estudo de caso.

5.1. SOBRE O LCD CONTROLLER AND DRIVER

O projeto LCD Controller and Driver faz parte do conjunto de projetos desenvolvidos pelo BRAZIL-IP. Iniciado em 2006 com o objetivo de validar a versão 2.0 do ipPROCESS e treinar alunos em projetos de IP-cores, este projeto consistiu no desenvolvimento de um controlador de display LCD (*Liquid Crystal Display*) alfanumérico, com duas linhas de 16 caracteres cada.

Displays LCD são interfaces de saída muito úteis em sistemas eletrônicos, como por exemplo, em impressoras e máquinas de café. Eles facilitam a interação do usuário com o sistema, mostrando informações relevantes do sistema assim como retornando o resultado obtido em algum tipo de processamento.

Como um dos seus intuitos era a validação da versão 2.0 do ipPROCESS, o desenvolvimento do projeto LCD Controller and Driver seguiu todas as disciplinas definidas no citado processo. Como resultado dessa fase o LCD Controller and Driver produziu os seguintes artefatos:

- ***Requirements****: Glossary, Vision Document, Requirements Specification, Use Case Specification* e *Use Case Model.*
- ***Analysis & Design****: Analysis Document, Design Document, Class Mapping. Analysis Model* e *Design Model.*
- ***RTL Implementation:*** *Implementation Plan* e Código-fonte.
- ***Verification****: Verification Plan, Test Cases* e *Testbench.*
- ***FPGA Prototype****: FPGA Prototype Plan* e módulo prototipado.

5.2. A DOCUMENTAÇÃO DA DISCIPLINA DEPLOYMENT

Pelo fato do LCD Controller and Driver ser um projeto já finalizado, a aplicação do fluxo de atividades da disciplina *Deployment* foi realizada a partir dos

resultados obtidos com a finalização das demais disciplinas do ipPROCESS. Sendo assim, os artefatos necessários à distribuição do LCD Controller and Driver não puderam ser elaborados de uma forma incremental.

O objetivo principal deste estudo de caso foi a verificação do uso de algumas informações geradas por outras disciplinas na elaboração da documentação proposta pela disciplina *Deployment*. Todos os artefatos produzidos durante a execução deste estudo de caso encontram-se no apêndice deste documento, assim como seus *templates*. A documentação do estudo de caso utilizada na elaboração desses artefatos encontra-se anexo a esse documento.

5.2.1. Deployment Plan e Bill of Materials

Conforme estabelecido no fluxo de atividades da disciplina *Deployment*, apresentado no capítulo anterior, o início da distribuição de um IP-core se dá através da realização do planejamento dessa distribuição. Esse planejamento é feito com a execução de duas atividades, são elas: *Develop Deployment Plan* e *Define Bill of Materials*.

A execução da atividade *Develop Deployment Plan*, como foi apresentada no capítulo anterior, gera o documento *Deployment Plan*, que apresenta a descrição do plano definido para a distribuição do IP-core. Para a elaboração desse documento são necessárias informações acerca do plano de desenvolvimento do IP-core, informações sobre o plano da iteração atual, como também informações acerca de critérios de aceitação do produto por parte do cliente. Pelo fato do ipPROCESS não produzir essas informações, elas foram assumidas como premissas na elaboração dessa disciplina. Desse modo, a elaboração do artefato *Deployment Plan* para o projeto LCD Controller and Driver não utilizou nenhum dos artefatos produzidos durante o desenvolvimento desse projeto. Sua elaboração baseou-se no fluxo de trabalho proposto pela disciplina *Deployment*, para a definição de quais atividades seriam necessárias para a distribuição do IP-core, além de pesquisas sobre formas de distribuição de IP-cores, para definir como o LCD Controller and Driver seria distribuído.

No artefato *Deployment Plan* do LCD Controller and Driver, ficou estabelecido que haveriam duas formas de distribuição do LCD Controller and Driver, uma seria através de um CD (*Compact Disk*) e a outra através do repositório do projeto, neste último caso, uma nova pasta seria acrescentada ao repositório, contendo a unidade de distribuição do projeto. Em relação aos recursos necessários a essa distribuição, foi estabelecido que os seguintes recursos são necessários:

- Recursos Humanos: um escritor técnico (*Technical Writer*) e uma pessoa para testar a unidade de distribuição.
- Recursos de Software: editor de texto (MS Word).
- Recursos de Hardware: Um computador pessoal com 512 MB de memória.

A atividade *Define Bill of Materials,* por sua vez, gera o documento *Bill of Material.* Esse documento lista os artefatos que irão compor o produto que será distribuído, descrevendo também as mudanças ocorridas nas versões desses artefatos, além de erros e problemas encontrados na atual versão do produto. Para a elaboração desse documento para o projeto LCD Controller and Driver, foi feita uma comparação dos artefatos propostos pelo ipQUALITY para serem entregues aos usuários do IP-core com os artefatos produzidos durante o desenvolvimento do LCD Controller and Driver. Ficou estabelecido que os artefatos que iriam compor o pacote de distribuição do LCD Controller and Driver seriam os seguintes:

- *LCD Controller and Driver Product Overview*: documento que apresenta uma visão geral do LCD Controller and Driver, mostrando sem muitos detalhes sua funcionalidade, compatibilidade com padrões e requisitos de integração.
- *LCD Controller and Driver User Guide:* documento que apresenta detalhadamente todas as informações necessárias ao reuso do LCD Controller and Driver.
- *LCD Controller and Driver Implementation Deployment:* Composto pela implementação do controlador de display LCD no nível de abstração RTL e do artefato *Prototyping Manual.*
- *LCD Controller and Driver Verification Environment:* Composto por casos de teste e componentes de verificação.

Dentre os quatro artefatos listados acima, o único que já possuía uma versão produzida durante o desenvolvimento do LCD Controller and Driver era o *LCD Controller and Driver User Guide*. Como essa é a primeira versão desse projeto que será distribuída aos usuários, não houveram mudanças efetuadas na versão do produto distribuído, também não foi detectada a existência de *bugs.*

5.2.2. Product Overview, Deliverable Documentation e User Guide.

A disciplina *Deployment* estabelece que após a realização do planejamento da distribuição do IP-core, sejam elaborados documentos a serem entregues aos usuários do IP-core, são eles o *Product Overview*, o *Deliverable Documentation* e o *User Guide*. Esses documentos devem auxiliar os usuários na escolha e reuso do IP-core. A elaboração desses documentos é realizada através do grupo *Develop Support Material* do fluxo de trabalho dessa disciplina.

O artefato *Product Overview* do LCD Controller and Driver tem o objetivo de auxiliar os potenciais usuários deste controlador na descrição de sua adequação antes da aquisição do mesmo. Para isso, esse artefato apresenta uma breve explicação sobre quais tipos de projetos podem utilizar o LCD Controller and Driver, que conforme já foi descrito nesse documento, são projetos eletrônicos que necessitam fornecer aos usuários algum tipo de informação ou resultado de processamento. Além disso, esse documento apresenta uma visão geral sobre as

funcionalidades, características, modos de operação e os padrões com os quais o LCD Controller and Driver é compatível. Nesse estudo de caso, as funcionalidades descritas foram as seguintes:

- Imprimir caracteres no display;
- Limpar o display;
- Mover o cursor;
- Mudar o número de linhas do display.

Por sua vez, as características listadas foram:

- Impressão dos caracteres em matrizes de 5x8 ou 5x10 pontos;
- Impressão em displays de 1 ou 2 linhas;
- Memória ROM com 240 mapeamentos de caracteres armazenados.

Os modos de operação do LCD Controller and Driver descritos foram os seguintes:

- Impressão em displays de 1 ou 2 linhas;
- Impressão de caracteres no modo ocidental ou oriental;
- Caracteres impressos com matrizes de 5x8 ou 5x10 pontos.

Em relação aos padrões compatíveis com o LCD Controller and Driver, foram listados os seguintes padrões:

- Hitachi;
- ASCII.

As informações descritas acima (funcionalidade, características, modos de operação e padrões aplicados) foram todas provenientes do artefato *Vision Document*, produzido na disciplina *Requirements*.

Uma segunda parte do documento *Product Overview* contém informações relacionadas aos requisitos necessários à integração do IP-Core em um projeto de SoC. Esses requisitos referem-se às informações sobre o nível de abstração da implementação do IP-core e características necessárias à plataforma de implementação, como informações sobre interfaces e processamento requerido, por exemplo. No caso do LCD Controller and Driver, este foi implementado nas linguagens SystemC e Verilog RTL. Sobre interfaces, é necessário o uso da interface OCP-IP para comunicar-se com esse controlador. Em relação à informação sobre nível de implementação, não há artefatos no ipPROCESS que forneça esse tipo de informação, para a elaboração deste tópico do documento *Product Overview* foi necessário obter a implementação do IP-Core para que se pudesse saber em qual linguagem este tinha sido implementado. Informações sobre requisitos adicionais para plataforma de implementação, como interfaces, podem ser obtidos a partir do artefato *Design Document*, já a informação sobre processamento requerido para o uso do IP-Core, nenhum artefato do ipPROCESS apresenta esse tipo de informação.

O artefato *Deliverable Documentation* apresenta informações sobre o produto distribuído, as mudanças incorporadas na atual versão do IP-core, os termos e

condições para o uso do IP-core e informações de contatos para informações e suporte.

As informações do *Deliverable Documentation* sobre o produto distribuído estão relacionadas à listagem dos artefatos que serão entregues aos usuários, sua disponibilidade (quando o IP-core estará disponível para ser fornecido aos usuários), versão histórica - descreve as mudanças ocorridas nas versões do IP-core, *bugs* conhecidos e informações relacionadas à unidade de distribuição, como título, data de publicação e organização dos artefatos. Dentre essas informações, dados sobre quais artefatos serão fornecidos aos usuários, versão histórica, mudanças ocorridas nas versões do IP-Core e descrição de *bugs* conhecidos são provenientes do artefato *Bill of Materials*, que como vimos, também é proposto pela disciplina *Deployment*, não tendo outro artefato no ipPROCESS que forneça esse tipo de informação.

Sobre informações acerca das mudanças ocorridas nas versões do LCD Controller and Driver, estas não são encontradas em nenhum de seus artefatos, para a obtenção desse tipo de informação é necessário um gerenciamento de configuração e mudanças, atualmente o ipPROCESS não trata deste aspecto.

Termos e condições de uso são informações que estão relacionadas à política da empresa desenvolvedora, esse tipo de informação é obtido durante a fase de planejamento da distribuição do IP-Core. No caso do LCD Controller and Driver, sua utilização é permitida apenas para uso acadêmico, uma vez que ele foi desenvolvido para esse fim.

Em relação ao documento *User Guide*, este deve apresentar informações detalhadas sobre certos aspectos do IP-core, são eles: sua funcionalidade, arquitetura, implementação, restrições, integração, verificação e instruções de como programá-lo. Durante a elaboração desse documento para o LCD Controller and Driver, parte dessas informações foram obtidas de documentos produzidos por outras disciplinas do ipPROCESS durante o desenvolvimento desse controlador. A seção do *User Guide* que apresenta informações gerais sobre o produto distribuído, informações estas relacionadas aos seguintes aspectos: características do IP-core, padrões com os quais o IP-core é compatível, sua arquitetura geral, modos de operação e a descrição de IP-cores que tem a funcionalidade incorporada por este IP-Core, foi elaborada com informações provenientes dos artefatos *Vision Document* (características do IP-core e compatibilidade com padrões), *Requirements Specification* (modos de operação) e *Design Document* (arquitetura geral e descrição de IP-cores).

As informações sobre a arquitetura do LCD Controller and Driver, reunidas na seção *System Logic Description* do *User Guide*, tiveram alguns dados provenientes do artefato *Design Document*, como informações sobre portas e protocolos do IP-core. Outras porém, não são geradas por nenhum dos artefatos elaborados durante o desenvolvimento desse controlador, é o caso de informações sobre diagramas de tempo. Por sua vez, as informações contidas no *User Guide* relativas à implementação do IP-core descrevem essa implementação por meio de métricas de certos atributos, como freqüência, potência, latência, número de pinos, entre outros. Em nenhum ponto da atual versão do ipPROCESS, esses atributos são mensurados e documentados, desta forma, para a obtenção dessas informações para o desenvolvimento do *User Guide* do LCD Controller and Driver foi necessário utilizar um ferramenta EDA, o Quartus, juntamente com a implementação desse IP-Core.

As informações sobre as restrições do LCD Controller and Driver, que estão no *User Guide,* são informações relativas à metodologia e fluxos de ferramentas utilizados em seu desenvolvimento, assim como instruções de como executar esse IP-core e de como executar sua verificação. As informações acerca de como executar a verificação podem ser obtidas através do documento *Test Automation Plan,* gerado pela disciplina *Verification,* porém, esse documento não foi produzido durante o desenvolvimento do LCD Controller and Driver. Sendo assim, a obtenção dessa informação assim como das demais, não puderam ser obtidas através da documentação anteriormente produzida, sua aquisição só foi possível através dos desenvolvedores desse IP-core.

Todas as informações sobre a verificação do IP-core, necessárias para a elaboração do *User Guide,* podem ser obtidas através dos seguintes documentos gerados durante o desenvolvimento deste IP-core: *Verification Plan*, *Test Cases* e *Test Automation Plan*. Como foi dito anteriormente, pelo fato de que durante o desenvolvimento do LCD Controller and Driver não foi produzido o documento *Test Automation Plan,* não possível a obtenção de certas informações, como a estrutura dos scripts utilizados na verificação, por exemplo.

As informações necessárias para programar o LCD Controller and Driver encontram-se reunidas na seção *Programmer's Reference Manual* do *User Guide.* Esta seção é formada por instruções de como programar o IP-core, assim como também informações sobre seus registradores programáveis. Essas informações não puderam ser adquiridas através de algum documento gerado durante o desenvolvimento do LCD Controller and Driver, sendo obtidas apenas com os desenvolvedores desse IP-core.

5.2.3. Deployment Unit

Depois de produzir os documentos explicados anteriormente (*Deployment Plan, Bill of Materials, Product Overview, User Guide e Deliverable Documentation*), estes foram reunidos com outros artefatos produzidos durante o desenvolvimento do LCD Controller and Driver (implementação em RTL, *Components Verification* e *Verification Scripts*), formando assim o artefato *Deployment Unit*. Por último, o LCD Controller and Driver foi definido de acordo com o padrão SPIRIT.

5.2.4. Changes Requests e Term of Acceptance

De acordo com o fluxo de tarefas da disciplina *Deployment*, após ser produzida a unidade de distribuição, é necessário que essa seja testada a fim de garantir que ela está completa e pronta para ser utilizada pelos usuários do IP-core. Como resultado deste teste, pode-se ter o artefato *Changes Requests*, que consiste na documentação das mudanças requeridas à unidade de distribuição. Durante a execução deste estudo de caso, a unidade de distribuição não pôde ser testada devido a problemas

com o cronograma deste trabalho, sendo assim, o artefato *Changes Requests* não foi elaborado.

Assim como ocorreu com o artefato *Changes Requests*, o artefato *Term of Acceptance* não foi gerado durante a execução desse estudo de caso. Já que esse documento expressa a aceitação, por parte do cliente, do IP-core recebido, e não houve fornecimento do LCD Controller and Driver para nenhum cliente.

5.3. CONCLUSÃO

Pode-se concluir com a realização deste estudo de caso que, o desenvolvimento do LCD Controller and Driver seguindo a versão atual ipPROCESS não gerou todos os artefatos necessários ao reuso deste IP-core. Um exemplo disto é a carência de informações acerca dos registradores programáveis, fluxos de ferramentas e metodologias utilizadas, por exemplo. Logo, há uma gama de informações que ainda precisam ser documentadas durante o desenvolvimento do IP-core. Sendo assim, para diminuir o esforço necessário para a elaboração de uma documentação ao usuário, é necessário mudanças em algumas disciplinas do ipPROCESS, conforme foi proposto por este trabalho.

A tabela abaixo apresenta as informações que precisam ser disponibilizadas aos usuários do LCD Controller and Driver, mas não foram geradas durante seu desenvolvimento. A segunda coluna dessa tabela mostra os documentos que este trabalho propõe serem desenvolvidos a fim de fornecer as informações citadas, e por sua vez, a última coluna apresenta a disciplina que cada um desses documentos deve ser gerado.

Tabela 5.1: Informações não encontradas durante o estudo de caso.

Informações	Documentos Propostos	Disciplina
Descrições dos registradores programáveis.	*Programmer's Reference Manual*	*Analysis & Design*
Instruções de como programar o dispositivo.	*Programmer's Reference Manual*	*Analysis & Design*
Nível de Implementação	*Implementation Description*	*RTL Implementation*
Fluxo de Ferramentas	*Prototyping Manual*	*FPGA Prototyping*
Metodologias Utilizadas	*Prototyping Manual*	*FPGA Prototyping*

6. Conclusões e Trabalhos Futuros

De acordo com o exposto ao longo deste trabalho, o aumento da complexidade dos projetos de circuitos integrados, associado a um mercado cada vez mais competitivo, com *time-to-market* decrescente e onde o cumprimento de prazos e custos, além da boa qualidade dos produtos desenvolvidos são vitais para a sobrevivência das empresas, ocasionou o surgimento de um novo paradigma de desenvolvimento de projeto de semicondutores, o paradigma *System-On-Chip* (SoC), que consiste no reuso de módulos com diferentes funcionalidades e pré-verificados (IP-cores). Com esse novo paradigma de desenvolvimento, surgiu a necessidade de além de produzir IP-cores de qualidade, desenvolver uma série de artefatos que auxiliem na escolha e no reuso dos mesmos.

Nesse contexto surgiu o ipPROCESS, um processo de desenvolvimento de soft IP-cores, com prototipação em FPGA. Mesmo provendo uma abordagem disciplinada para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento de componentes de hardware, o ipPROCESS não define atividades que tratem do fornecimento aos usuários, de informações a serem distribuídas juntamente com o IP-core para escolha e reuso deste.

Sendo assim, este trabalho definiu um conjunto de atividades e artefatos para serem incluídos ao ipPROCESS, a fim de fornecer aos usuários do IP-core, os *IP-Integrators*, as informações necessárias e suficientes à integração deste em um SoC. Essas atividades e artefatos foram reunidos em uma disciplina, a disciplina *Deployment*.

Com a inserção da disciplina *Deployment* ao ipPROCESS, a distribuição do IP-core será feita de modo iterativo e incremental, fazendo com que o esforço necessário à essa distribuição seja distribuído ao longo de todo desenvolvimento do IP-core.

No capítulo 5 foi apresentado um estudo de caso a fim de demonstrar que parte das informações necessárias aos documentos estabelecidos pela disciplina *Deployment* são geradas durante as demais disciplinas do ipPROCESS.

Dentre os benefícios obtidos com este trabalho, pode-se destacar:

- Uma nova disciplina para o ipPROCESS;
- Uma reestruturação no fluxo de atividades e artefatos de outras disciplina do ipPROCESS.

A continuidade deste trabalho se dará através da validação da disciplina *Deployment* através de vários estudos de caso, pois conforme descrito neste trabalho, o estudo de caso aqui relatado utilizou um projeto já finalizado, não podendo com isso, validar o fluxo de atividades da disciplina proposta.

Referências

[1] Wikipedia. Lei de Moore. Url: http://pt.wikipedia.org/wiki/Lei_de_Moore. Acesso em: 29/04/2008.

[2] VÖRG, Andreas; RADETZKI, Martin; ROSENSTIEL, Wolfgang. *Measurement of IP Qualification Costs and Benefits.* In: Proceedings of the conference on Design, automation and test in Europe, vol. 2, 2004, Washington, USA. p. 20.996.

[3] DOLPHI INTEGRATION. *The enabler of mixed signal System-on-Chip.* Disponível em: http://www.dolphin.fr/corporate/investors/investors_design.html. Acesso em: 27/05/2008.

[4] SAVAGE, Warren; CHILTON, John; CAMPOSANO, Raul. *IP Reuse in the System on a Chip Era,* Madrid: 13th International Symposium on System Synthesis (ISSS'00) p. 2. 2000.

[5] RMM. Reuse Methodology Manual for System-On-A-Chip Designs. 3.ed Kluwer Academic Press, 2002.

[6] HAASE, Jürgen. *Design Methodology for IP Providers.* Hannover, Germany. 1999.

[7] IpPROCESS. *Modelagem do processo ipPROCESS.* 2006. Disponível em: http://www.lincs.org.br/ipprocess. Acesso em: 01/05/2008.

[8] RUP. *Rational Unified Process*. 2002. Disponível em: http://www-306.ibm.com/software/awdtools/rup. Acesso em: 01/05/2008.

[9] PALMA, José; MORAES Fernando; CALAZANS Ney. *Métodos para Desenvolvimento e Distribuição de IP Cores.* Faculdade de Informática - PUCRS Disponível em: http://www.dimap.ufrn.br/~ivan/Topicos/2001_scr2001_IP_cores.pdf. Acesso em: 27/05/2008.

[10] SANTOS, Francielle. *IP-QUALITY- Identificação e Análise de Atributos de Qualidade de soft IP-cores.* Recife: UFPE, 2006.

[11] VSIA. *Site do Virtual Socket Alliance.* 2007. Disponível em: http://www.vsia.org. Acesso em: 01/05/2008.

[12] IEEE. *Site do Instituto de Engenheiros Elétricos e Eletrônicos.* 2008. Disponível em: http://www.ieee.org/portal/site. Acesso em: 01/05/2008.

[13] SPIRIT. *Site do Consórcio SPIRIT.* 2008. Disponível em: http://www.spiritconsortium.org. Acesso em: 02/05/2008.

[14] IBM Rational Software. *Site da IBM Rational Software.* Disponível em: http://www-306.ibm.com/software/rational/. Acesso em: 10/05/2008.

[15] IBM. *Site da IBM.* Disponível em: http://www.ibm.com. Acesso em: 10/05/2008.

[16] CIN. *Site da disciplina Análise e Projeto de Sistema.* Disponível em: http://www.cin.ufpe.br/~if718/. Acesso em: 20/06/2008

[17] MOTOROLLA. *Site da Motorola.* Disponível em: http://www.motorola.com. Acesso em: 21/05/2008.

[18] R BUSINESS NETWORK. *Motorola Offers Public First Comprehensive Set of Semiconductor Reuse Standards as Part of System-On-Chip Technology.* Disponível em: http://findarticles.com/p/articles/mi_m0EIN/is_1999_Dec_16/ai_58271361. Acesso em: 21/05/2008.

[19] BENNER BUSINESS NETWORK. *Motorola Announces Design Documentation Standard; Provides a Key to Reduced Cycle Times and Increased Standardization.* Disponível em: http://findarticles.com/p/articles/mi_m0EIN/is_2000_Jan_17/ai_58610362. Acesso em: 21/05/2008.

[20] FREESCALE. *Site da Freescale.* Disponível em: http://www.freescale.com. Acesso em: 21/05/2008.

[21] AZUARA, Luis; DORSCH Rainer. *Design of Systems-on-a-Chip for Communication.* Disponível em: http://www.iti.uni-stuttgart.de/~rainer/Lehre/SOCfCA01/Presentation9/designReuse.pdf. Acesso em: 21/05/2008.

[22] IEEE. *Site da IEEE.* Disponível em: http://www.ieee.org.br/index.shtml. Acesso em: 22/05/2008.

[23] ISO. *Site da ISSO.* Disponível em: http://www.iso.org. Acesso em: 22/05/2008.

[24] BRAZIL-IP. *Site do projeto Brazil-IP.* Disponível em: http://www.brazilip.org.br. Acesso em: 22/05/2008.

[25] PRADO, Bruno. *Curso de Verificação Funcional.*

[26] QIP. *IP Quality Pillar.* Disponível em: http://www.vsia.org/pillars/IP_Quality_Pillar.htm. Acesso em: 03/06/2008.

[27] SPEM. *Especificação do SPEM.* Disponível em: http://www.omg.org/technology/documents/formal/spem.htm. Acesso em: 03/06/2008.

[28] ANDRADE, Millena. *IPZip- Ferramenta de Criação de Pacotes de Distribuição de IP Cores.* Recife: UFPE, 2007.

Apêndice A – Templates da Documentação Proposta pela Disciplina *Deployment*

Neste apêndice encontram-se os *templates* dos documentos propostos pela disciplina *Deployment*.

ipPROCESS

Deployment Plan

<Module Name>

<Project Name>

Customer: <Name>

<Customer or Project logo>

Version : <Document version>

[Note: The following template is provided for use with the ipPROCESS. Text enclosed in square brackets and displayed in grey italics (style=ip_comment) is included to provide guidance to the author and should be deleted before publishing the document.]

Revision History

Date	Version	Description	Author
<mm/dd/yy>	<x.y>	<detail>	<name>

1. Introduction

[This topic should give a document overview, describing the purpose of this document.]

Use this style to describe a section {*style* = *ip_normal*}

1.1 Scope

[Describe the scope of this document, including the associated project, and what is affected or influenced by this plan.]

Use this style to describe a section {*style* = *ip_normal*}

1.2 Definitions, Acronyms and Abbreviations (optional)

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret this document. This information may be provided by reference to the project's Glossary.]

Use this style to describe a subsection {*style* = *ip_normal*}

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Name	Detail {style=ip_table_text}

1.3 Document Structure

[This subsection describes what the rest of this document contains and explains how the document is organize]

Use this style to describe a subsection {*style* = *ip_normal*}

- Use this style to list items {style = ip_list}
- <Section 2 – Section 2 Name: section description... >
- <Section N – Section N Name: section description...>
- Section <N+1> – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Deployment Planning

[This section should contain the deployment planning, identifying how it will be deployed, the necessary activities and its responsibles. A schedule for deployment activities should be developed.]

Use this style to describe a section {*style* = *ip_normal*}

2.1 Product Deployment

[Describe how the product will be deployed.]

2.2 Deployment Activities

[Describe the necessary activities to deploy the IP-Core.]

2.3 Schedule

[Create a schedule for the activities identified in the section above.]

3. Licensing

[This section provides information about how the deployed product will be licensed.]

Use this style to describe a section {*style = ip_normal*}

4. Training and Assistance

[This section should identify if training is necessary to use the deployed product and how will be done the assistance for the users.]

Use this style to describe a section {*style = ip_normal*}

5. Resources

[This section identifies the necessary resources for product deployment.]

Use this style to list references *{style= ip_normal}*

5.1 Human Resources

[This section provides a list of the necessary human resources for product deployment.]

Use this style to list references *{style= ip_normal}*

5.2 Software Resources

[This section provides a list of the necessary software resources for product deployment.]

Use this style to list references *{style= ip_normal}*

5.3 Hardware Resources

[This section provides a list of the necessary hardware resources for product deployment.]

Use this style to list references *{style= ip_normal}*

6. References

[This section provides a complete list of all documents referenced elsewhere in this document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

[1] Use this style to list references *{style= ip_reference}*

[2] Document Title; <File name and link >;

[3] Title; Number (if applicable); Date; Institution, Team responsible for the document; Document link;

ipPROCESS

Bill of Material

<Module Name>

<Project Name>

Customer: <Name>

<Customer or Project logo>

Version : <Document version>

[Note: The following template is provided for use with the ipPROCESS. Text enclosed in square brackets and displayed in grey italics (style=ip_comment) is included to provide guidance to the author and should be deleted before publishing the document.]

Revision History

Date	Version	Description	Author
<mm/dd/yy>	<x.y>	<detail>	<name>

1. Introduction

[This topic should give a document overview, describing the purpose of this document.]

Use this style to describe a section {*style* = *ip_normal*}

1.1 Scope

[Describe the scope of this document, including the associated project, and what is affected or influenced by this plan.]

Use this style to describe a section {*style* = *ip_normal*}

1.2 Definitions, Acronyms and Abbreviations (optional)

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret this document. This information may be provided by reference to the project's Glossary.]

Use this style to describe a subsection {*style* = *ip_normal*}

Table 1. Definitions, acronyms and abbreviations.

Term	Description
Name	Detail {style=ip_table_text}

1.3 Document Structure

[This subsection describes what the rest of this document contains and explains how the document is organize]

Use this style to describe a subsection {*style* = *ip_normal*}

- Use this style to list items {style = ip_list}
- <Section 2 – Section 2 Name: section description... >
- <Section 3 – Section 3 Name: section description...>
- <Section N – Section N Name: section description...>
- Section <N+1> – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Version Description

[This section shows the elements that form the deployed product, the changes occurred in the actual version and the known errors and problematic features.]

Use this style to describe a section {*style* = *ip_normal*}

2.1 Inventory of Material

[List all the physical media, such as CDs, floppies, and so on, and associated documentation that make up the product version being released. Identify numbers, titles, abbreviations, dates, versions and release numbers as applicable.]

2.2 Inventory of IP-Core Files

[List all HDL files, scripts files and verification files that make up the IP-Core being released and its installation and verification environments. Identify numbers, titles, abbreviations, dates, versions, and release numbers as applicable.]

2.3 Changes

[List all changes incorporated into the product version since the previous version. Describe the effect of each change on product use or operation, as applicable.]

2.4 Known Errors and Problematic Features

[This section must describes the known errors and problematic features of the current version of the product, describe steps that can be taken to recognize, avoid, correct or handle any problematic features.]

3. References

[This section provides a complete list of all documents referenced elsewhere in this document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

[1] Use this style to list references *{style= ip_reference}*

[2] Document Title; <File name and link >;

[3] Title; Number (if applicable); Date; Institution, Team responsible for the document; Document link;

[4] …

ipPROCESS

Product Overview

<Module Name>

<Project Name>

Customer: <Name>

<Customer or Project logo>

Version : <Document version>

[Note: The following template is provided for use with the ipPROCESS. Text enclosed in square brackets and displayed in grey italics (style=ip_comment) is included to provide guidance to the author and should be deleted before publishing the document.]

Revision History

Date	Version	Description	Author
<mm/dd/yy>	<x.y>	<detail>	<name>

1. Introduction

[This topic should give a document overview, describing the purpose of this document.]

Use this style to describe a section {*style* = *ip_normal*}

1.1 Scope

[Describe the scope of this document, including the associated project, and what is affected or influenced by this plan.]

Use this style to describe a section {*style* = *ip_normal*}

1.2 Definitions, Acronyms and Abbreviations (optional)

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret this document. This information may be provided by reference to the project's Glossary.]

Use this style to describe a subsection {*style* = *ip_normal*}

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Name	Detail {style=ip_table_text}

1.3 Document Structure

[This subsection describes what the rest of this document contains and explains how the document is organize]

Use this style to describe a subsection {*style* = *ip_normal*}

- Use this style to list items {style = ip_list}
- <Section 2 – Section 2 Name: section description... >
- <Section 3 – Section 3 Name: section description...>
- <Section N – Section N Name: section description...>
- Section <N+1> – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.4 Contact Information

[This subsection provides a list of the points of organizational contact (POC) that may be needed by the document reader for informational and troubleshooting purposes.]

Use this style to describe a subsection {style = ip_table_text}

Table 2: Contact Information.

Name	*[The name of the organization]*
Address	*[The location of the organization]*
Division	*[Provides additional details concerning the specific division name or department within the organization, and the name of the primary individual for initial contact]*
Primary Contact	*[Provides the name of the primary individual for initial contact]*
Phone	*[A primary contact phone number]*
E-mail	*[The primary contact email address]*
Fax	*[The primary contact fax number]*
URL	*[The URL (Uniform Resource Locator) for finding additional information on the World Wide Web]*

2. Product Overview

2.1 Product Capabilities

[This section provides a list of user's benefits.]

Use this style to list references *{style= ip_normal}*

2.2 IP-Core Features

[This subsection must be used for summarize the key features of the VC, providing a means for VC providers to list additional information that differentiate the VC].

Use this style to describe a section {*style = ip_normal*}

2.3 Modes of Operation

[This subsection describes the modes of operation that cause the VC to change its functional behavior.]

Use this style to describe a section {*style = ip_normal*}

2.4 Standards Compliance

[This section must be used for list the main features and benefits of the VC, standards which the VC is compliant, indicating the specific revision of each standard applicable to the VC.]

Use this style to describe a section {style = ip_table_text}

Table 3: List of the applicable standards.

Organization	Name	Version	Revision
[Name of the organization that published the standard]	*[The identifier and title of the standard]*	*[The release version number of the standard applicable to the VC being described]*	*[The release version revision number of the standard (if applicable)]*

3. Integration Informations

3.1 Qualification Levels

[This attribute indicates the level at which the VC has been implemented, providing useful information to implementers.

Usage Notes: Pick list

(Architecture_Simulation | RTL_Verification | Gate _Verification | Formal_Verification |

Static_Timing_Analysis | FPGA | Prototype | Production | Others).]

Use this style to describe a section {style = ip_normal}

3.2 Integration Requirements

[This section must describe additional implementation platform requirements, including: minimum processing power required of the embedded computing engine or the particular design skills required of the implementation team. Example:

CPU technology;

Implementation Capability;

Logic library;

Application Expertise;

Interfaces.]

Use this style to describe a section {style = ip_normal}

4. References

[This section provides a complete list of all documents referenced elsewhere in this document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

[1] Use this style to list references *{style= ip_reference}*

[2] Document Title; <File name and link >;

[3] Title; Number (if applicable); Date; Institution, Team responsible for the document; Document link;

[4] …

ipPROCESS

Deliverable Documentation

<Module Name>

<Project Name>

Customer: <Name>

<Customer or Project logo>

Version : <Document version>

[Note: The following template is provided for use with the ipPROCESS. Text enclosed in square brackets and displayed in grey italics (style=ip_comment) is included to provide guidance to the author and should be deleted before publishing the document.]

Revision History

Date	Version	Description	Author
<mm/dd/yy>	<x.y>	<detail>	<name>

1. Introduction

[This topic should give a document overview, describing the purpose of this document.]

Use this style to describe a section {*style* = *ip_normal*}

1.1 Scope

[Describe the scope of this document, including the associated project, and what is affected or influenced by this plan.]

Use this style to describe a section {*style* = *ip_normal*}

1.2 Definitions, Acronyms and Abbreviations (optional)

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret this document. This information may be provided by reference to the project's Glossary.]

Use this style to describe a subsection {*style* = *ip_normal*}

Table 1. Definitions, acronyms and abbreviations.

Term	Description
Name	Detail {style=ip_table_text}

1.3 Document Structure

[This subsection describes what the rest of this document contains and explains how the document is organize]

Use this style to describe a subsection {*style* = *ip_normal*}

- Use this style to list items {style = ip_list}
- <Section 2 – Section 2 Name: section description... >
- <Section 3 – Section 3 Name: section description...>
- <Section N – Section N Name: section description...>
- Section <N+1> – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Deliverable Product

Use this style to describe a section {*style = ip_normal*}

2.1 Deliverables List

[This subsection should provide a comprehensive itemization of all the component parts of the VC.

Usage Notes: If the deliverable is not specified by VSIA, choose an appropriate name different from existing VSIA deliverable names and provide an explanation in the remark.]

Use this style to describe a section {style = ip_table_text}

Table 0.1: List of the VC Deliverables.

Name	Filename	Format	Description	Comply
[Concatenate VSIA doc, version, section and Deliverable name. Or name of the artifact]	*[Name of the file containing the named deliverable]*	*[Data format, applicable standard, and version of the format]*	*[Data format, applicable standard, and version of the format]*	☐
				☐
				☐
				☐

2.2 Availability

[This subsection defines that the VC is available for transfer or provides a target date for availability.]

Use this style to describe a section {style = ip_normal}

2.3 Version History

[This section contains a version history of the VC, describing changes and versions releases associated with the VC. It should identify the current release and be coordinated with the claims.]

Use this style to describe a section {style = ip_table_text}

Table 0.2: VC Version History.

Version	Description	Date
[Version code of the VC being described]	*[Description of the version]*	*[Date value.(yyyymmdd)]*

2.4 Known Bugs and Problematic Features

[This section must describes the known errors and problematic features of the current version of the product, describe steps that can be taken to recognize, avoid, correct or handle any problematic features.

Usage Notes: Here put the major bugs, using the bug id/bug discussion pair's format. Concatenating several bugs per entry is allowed. Remember that the fully detailed bug list is a deliverable document.]

- Use this style to describe a section {style = ip_list}
- Bug ID – Bug discussion.

2.5 Transfer Package Information

[This section contains:

Title, revision, and publication date of the deliverables;

Organization of the deliverables in terms of transfer media types and archive/file structure;

Part numbers and product nomenclature.]

Use this style to describe a section {style = ip_normal}

3. Version Changes

[List all changes incorporated into the product version since the previous version. Describe the effect of each change on product use or operation, as applicable.]

Use this style to describe a section {style = ip_normal}

4. Standard Terms and Conditions

[This section contains standard terms and conditions governing the evaluation and subsequent incorporation of VC provider intellectual property into VC user systems including:

Evaluation procedures;

Technical support and training;

Applicable patents, trademarks, and copyrights;

Business model;

Licensing requirements;

Applicable royalties;

Export limitations;

Warranties, liability limitations, disclaimers.]

Use this style to describe a section {style = ip_normal}

5. Contact

5.1 Supporting VC Partner (Recommended)

[This section identifies partners (and their complete contact information) who are willing to supply one or more services such as: design service, EDA support, foundry service, or consulting service.]

Use this style to describe a subsection {style = ip_table_text}

Table 0.3: Supporting VC partners.

Name	*[The name of the organization]*
DUNS	*[DUNS number for the named supplier]*
Address	*[The location of the organization]*
Division	*[A single division and individual contact name]*
Primary Contact	*[Provides the name of the primary individual for initial contact]*
Phone	*[A primary contact phone number]*
E-mail	*[The primary contact email address for the named supplier]*
Fax	*[The primary contact fax number]*
URL	*[TThe URL for named supplier]*
Service Type	*[Lists the service types]*

5.2 IP Provider Contact

[This section contains VC provider contact information, including address, email address, web URL, contact names for sales, support, licensing etc.]

Use this style to describe a subsection {style = ip_table_text}

Table 0.4: Contact Information.

Name	*[The name of the organization]*
DUNS	*[Dun & Bradstreet ID of the VC Provider organization]*
Address	*[The location of the organization]*
Division	*[Provides additional details concerning the specific division name or department within the organization, and the name of the primary individual for initial contact]*
Primary Contact	*[Provides the name of the primary individual for initial contact]*
Phone	*[A primary contact phone number]*
E-mail	*[The primary contact email address]*
Fax	*[The primary contact fax number]*
URL	*[The URL (Uniform Resource Locator) for finding additional information on the World Wide Web]*

6. References

[This section provides a complete list of all documents referenced elsewhere in this document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

[1] Use this style to list references *{style= ip_reference}*

[2] Document Title; <File name and link >;

[3] Title; Number (if applicable); Date; Institution, Team responsible for the document; Document link;

[4] …

ipPROCESS

Programmer´s Reference Manual

<Module Name>

<Project Name>

Customer: <Name>

<Customer or Project logo>

Version : <Document version>

[Note: The following template is provided for use with the ipPROCESS. Text enclosed in square brackets and displayed in grey italics (style=ip_comment) is included to provide guidance to the author and should be deleted before publishing the document.]

Revision History

Date	Version	Description	Author
<mm/dd/yy>	<x.y>	<detail>	<name>

1. Introduction

[This topic should give a document overview, describing the purpose of this document.]

Use this style to describe a section {*style* = *ip_normal*}

1.1 Scope

[Describe the scope of this document, including the associated project, and what is affected or influenced by this plan.]

Use this style to describe a section {*style* = *ip_normal*}

1.2 Definitions, Acronyms and Abbreviations (optional)

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret this document. This information may be provided by reference to the project's Glossary.]

Use this style to describe a subsection {*style* = *ip_normal*}

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Name	Detail {style=ip_table_text}

1.3 Document Structure

[This subsection describes what the rest of this document contains and explains how the document is organize]

Use this style to describe a subsection {*style* = *ip_normal*}

- Use this style to list items {style = ip_list}
- <Section 2 – Section 2 Name: section description... >
- <Section 3 – Section 3 Name: section description...>
- <Section N – Section N Name: section description...>
- Section <N+1> – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.4 Contact Information

[This subsection provides a list of the points of organizational contact (POC) that may be needed by the document reader for informational and troubleshooting purposes.]

Use this style to describe a subsection {style = ip_table_text}

Table 2: Contact Information.

Name	*[The name of the organization]*
Address	*[The location of the organization]*
Division	*[Provides additional details concerning the specific division name or department within the organization, and the name of the primary individual for initial contact]*
Primary Contact	*[Provides the name of the primary individual for initial contact]*
Phone	*[A primary contact phone number]*
E-mail	*[The primary contact email address]*
Fax	*[The primary contact fax number]*
URL	*[The URL (Uniform Resource Locator) for finding additional information on the World Wide Web]*

2. Instruction Set

[This section should be delivered the complete set of instructions available to the programmer. Explain the implicit syntax.]

Use this style to list references *{style= ip_normal}*

3. Register Information

[This section should be presented and described the registers available to the programmer.]

Use this style to describe a section {*style = ip_normal*}

4. Software Information (Instructions to Program the Device)

[This section includes an API with software requirements and instructions to interface this VC. How to use the VC, taken into consideration the instruction set and register information delivered previously.]

4.1 API Support

[This subsection details about any APIs available for the VC.]

Use this style to describe a section {style = ip_table_text}

Table 3: Information of VC APIs.

API Name	API Standard Reference	API Description
[Name of the API]	*[Title and/or document number of the published source which controls content of API]*	*[Relevant application information of the API]*

5. References

[This section provides a complete list of all documents referenced elsewhere in this document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

[1] Use this style to list references *{style= ip_reference}*

[2] Document Title; <File name and link >;

[3] Title; Number (if applicable); Date; Institution, Team responsible for the document; Document link;

[4] …

ipPROCESS

Implementation Description

<Module Name>

<Project Name>

Customer: <Name>

<Customer or Project logo>

Version : <Document version>

[Note: The following template is provided for use with the ipPROCESS. Text enclosed in square brackets and displayed in grey italics (style=ip_comment) is included to provide guidance to the author and should be deleted before publishing the document.]

Revision History

Date	Version	Description	Author
<mm/dd/yy>	<x.y>	<detail>	<name>

1. Introduction

[This topic should give a document overview, describing the purpose of this document.]

Use this style to describe a section {*style* = *ip_normal*}

1.1 Scope

[Describe the scope of this document, including the associated project, and what is affected or influenced by this plan.]

Use this style to describe a section {*style* = *ip_normal*}

1.2 Definitions, Acronyms and Abbreviations (optional)

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret this document. This information may be provided by reference to the project's Glossary.]

Use this style to describe a subsection {*style* = *ip_normal*}

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Name	Detail {style=ip_table_text}

1.3 Document Structure

[This subsection describes what the rest of this document contains and explains how the document is organize]

Use this style to describe a subsection {*style* = *ip_normal*}

- Use this style to list items {style = ip_list}
- <Section 2 – Section 2 Name: section description... >
- <Section 3 – Section 3 Name: section description...>
- <Section N – Section N Name: section description...>
- Section <N+1> – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.4 Contact Information

[This subsection provides a list of the points of organizational contact (POC) that may be needed by the document reader for informational and troubleshooting purposes.]

Use this style to describe a subsection {style = ip_table_text}

Table 2: Contact Information.

Name	*[The name of the organization]*
Address	*[The location of the organization]*
Division	*[Provides additional details concerning the specific division name or department within the organization, and the name of the primary individual for initial contact]*
Primary Contact	*[Provides the name of the primary individual for initial contact]*
Phone	*[A primary contact phone number]*
E-mail	*[The primary contact email address]*
Fax	*[The primary contact fax number]*
URL	*[The URL (Uniform Resource Locator) for finding additional information on the World Wide Web]*

2. Structure Models

[Here should be showed in a graphical manner the structural model of the IP-core provided.]

Use this style to describe a section {style = ip_normal}

3. Qualification Levels

[This attribute indicates the level at wich the VC has been implemented, providing useful information to implementers.

Usage Notes: Pick list

*(Architecture_Simulation \ RTL_Verification | Gate_Verification \ Formal_Verification *

Static_Timing_Analysis | FPGA \ Prototype \ Production \ Others).]

Use this style to describe a section {style = ip_normal}

4. Performance Information (CM)

[This subsection must be used describe the VC performance claims

Including:

Frequency of operation, clock (measured or estimated);

Normalized measured (s) (MIPs, bps, etc. - measured or estimated);

Power consumption (measured or estimated);

Signal-to-noise ratio (SNR), gain,etc. (mixed signal VCs);

Architectural performance numbers (e.g. bit rate).]

Use this style to describe a section {style = ip_normal}

4.1 Frequency

[This subsection indicates the clock speed or data rate at which the VC operates.]

Use this style to describe a section {style = ip_table_text}

Table 3: Frequency values for the VC.

Minimum	Typical	Maximum
[Description of the minimum value of the primary operation speed of the VC (Hertz)]	*[Description of the typical value of the primary operation speed of the VC (Hertz)]*	*[Description of the maximum value of the primary operation speed of the VC (Hertz)]]*

4.2 Power Consumption

[This subsection provides information about the power consumption of the VC at a specified operation frequency.]

Use this style to describe a section {style = ip_table_text}

Table 4: Power consumption values for the VC.

Minimum	Typical	Maximum
[Description of the minimum value of the power consumption of the VC (Watt)]	*[Description of the typical value of the power consumption of the VC (Watt)]*	*[Description of the maximum value of the power consumption of the VC (Watt)]]*

4.3 Throughput

[This subsection provides information about the rate at which the VC produces or process data.]

Use this style to describe a section {style = ip_table_text}

Table 5: Throughput of the VC.

bps	*[Throughput represented in bits per second]*
fps	*[Throughput represented in frames per second]*
MIPS	*[Throughput represented in millions of instructions per second]*
sps	*[Throughput represented in samples per second]*

4.4 Latency

[This subsection describes the maximum number of input data signals consumed before the first output is produced.]

Use this style to describe a section {style = ip_table_text}

Table 6: Latency of the VC.

Mode	System Level Interface	Clock	sec
[The mode of VC operation for which the Latency is applicable]	*[Name the specific system-level interface associated with the mode]*	*[Latency from input to output measured in terms of system clock cycles]*	*[Latency from input to output measured in seconds]*

5. Form Information (conditional)

[This subsection must be used for describe the hardness of the VC (soft, firm or hard), including:

Estimated or exact gate count;

Area;

Pin counts (input, output, test);

Dimensions.]

Use this style to describe a section {style = ip_normal}

5.1 Gate Count

[This subsection defines the number of NAND gates for the design in the specific library used by the VC provider for verification.]

Use this style to describe a section {style = ip_table_text}

Table 7: Gate count for the VC.

Minimum	Typical	Maximum
[Minimum equivalent number of NAND gates (Gate)]	*[Typical equivalent number of NAND gates (Gate)]*	*[Maximum equivalent number of NAND gates (Gate)]*

5.2 Bit Count

[This subsection defines the number of memory bits in the VC.]

Use this style to describe a section {style = ip_table_text}

Table 8: Bit count for the VC.

Minimum	Typical	Maximum
[Minimum number of bits (Bit)]	*[Typical number of bits (Bit)]*	*[Maximum number of bits (Bit)]*

5.3 Area (Mandatory for Hard and AMS VCs)

[This subsection defines the total area of the VC. For soft and firm VCs, indicate in a remark if estimated, or provide the area resulting from a verification run and/or previous implementation run(s).]

Use this style to describe a section {style = ip_table_text}

Table 9: Total area for the VC.

Minimum	Typical	Maximum
[Minimum mplemented/estimated area for the entire VC (mm^2)]	*[Typical implemented/estimated area for the entire VC (mm^2)]*	*[Maximum mplemented/estimated area for the entire VC (mm^2)]*

5.4 Logic Area (Mandatory for Hard and AMS VCs)

[This subsection defines the total area of the VC. For soft and firm VCs, indicate in a remark if estimated, or provide the area resulting from a verification run and/or previous implementation run(s).]

Use this style to describe a section {style = ip_table_text}

Table 10: Logic area for the VC.

Minimum	Typical	Maximum
[Minimum implemented/estimated area of logic in the VC (mm²)]	*[Typical implemented/estimated area of logic in the VC (mm²)]*	*[Maximum implemented/estimated area of logic in the VC (mm²)]*

5.5 Memory Area (Mandatory for Hard and AMS VCs)

[This subsection defines the total area of the VC. For soft and firm VCs, indicate if estimated, or provide the area resulting from a verification run.]

Use this style to describe a section {style = ip_table_text}

Table 11: Memory area for the VC.

Minimum	Typical	Maximum
[Minimum implemented/estimated area of memory in the VC (mm²)]	*[Typical implemented/estimated area of memory in the VC (mm²)]*	*[Maximum implemented/estimated area of memory in the VC (mm²)]*

5.6 Pin Count

[This subsection defines the number of I/Os (pins) of various types that acess the VC.]

Use this style to describe a section {style = ip_table_text}

Table 12: Pin count.

Input	*[Number of input pins]*
Output	*[Number of output pins]*
Bi-directional	*[Number of bi-directional pins]*
Test	*[Number of tester pins]*
Other	*[Number of other types of pins. Indicate the types in a remark]*

5.7 Flip Flop Count

[This subsection defines the number of flip-flops per clock domain of the VC.]

Use this style to describe a section {style = ip_table_text}

Table 13: Flip flop count.

Value	Clock domain
[Number of flip-flops in the named clock domain]	*[Name of the clock domain for the reported flip-flop count]*

6. Simulation Environment

[This section is used to explain how to simulate the IP-core. If scripts are delivered for this purpose, they have to be referenced and explained here.]

Use this style to describe a section {style = ip_normal}

7. Timing Diagrams

[This section includes the graphical illustration of clocks, data interfaces, status, etc, identifying the valid inputs and outputs and portraying step by step sequential VC behavior.

VC providers should include detailed information covering false path designations, timing errors that can be ignored, and set-u and holde condition as appropriated.]

Use this style to describe a section {style = ip_normal}

8. References

[This section provides a complete list of all documents referenced elsewhere in this document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

[1] Use this style to list references *{style= ip_reference}*

[2] Document Title; <File name and link >;

[3] Title; Number (if applicable); Date; Institution, Team responsible for the document; Document link;

[4] …

ipPROCESS

Prototype Manual

<Module Name>

<Project Name>

Customer: <Name>

<Customer or Project logo>

Version : <Document version>

[Note: The following template is provided for use with the ipPROCESS. Text enclosed in square brackets and displayed in grey italics (style=ip_comment) is included to provide guidance to the author and should be deleted before publishing the document.]

Revision History

Date	Version	Description	Author
<mm/dd/yy>	<x.y>	<detail>	<name>

1. Introduction

[This topic should give a document overview, describing the purpose of this document.]

Use this style to describe a section {*style* = *ip_normal*}

1.1 Scope

[Describe the scope of this document, including the associated project, and what is affected or influenced by this plan.]

Use this style to describe a section {*style* = *ip_normal*}

1.2 Definitions, Acronyms and Abbreviations (optional)

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret this document. This information may be provided by reference to the project's Glossary.]

Use this style to describe a subsection {*style* = *ip_normal*}

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Name	Detail {style=ip_table_text}

1.3 Document Structure

[This subsection describes what the rest of this document contains and explains how the document is organize]

Use this style to describe a subsection {*style* = *ip_normal*}

- Use this style to list items {style = ip_list}
- <Section 2 – Section 2 Name: section description... >
- <Section 3 – Section 3 Name: section description...>
- <Section N – Section N Name: section description...>
- Section <N+1> – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.4 Contact Information

[This subsection provides a list of the points of organizational contact (POC) that may be needed by the document reader for informational and troubleshooting purposes.]

Use this style to describe a subsection {style = ip_table_text}

Table 2: Contact Information.

Name	*[The name of the organization]*
Address	*[The location of the organization]*
Division	*[Provides additional details concerning the specific division name or department within the organization, and the name of the primary individual for initial contact]*
Primary Contact	*[Provides the name of the primary individual for initial contact]*
Phone	*[A primary contact phone number]*
E-mail	*[The primary contact email address]*
Fax	*[The primary contact fax number]*
URL	*[The URL (Uniform Resource Locator) for finding additional information on the World Wide Web]*

2. Design Constraints and Compability (CM)

[This is an optional section. Use it if you want to provide information about the constraints or compability of the delivered VC under the integration point of view.]

Use this style to describe a section {style = ip_normal}

3. Reference Environment

[This section contains specific descriptions of technology used in VC development, necessary for its successful deployment, specifying tools, flows and technology.

The VC Provider also must define the context in which all features, characteristics and performance claims were established and validated, including:

A generic technology library (typically, one readily available in the industry);

The design flow;

The design tools used;

The options, parameters, and all other information used to implement the VC in this environment.]

Use this style to describe a section {style = ip_normal}

3.1 Tools, Flows and Methodologies

[This section documents the reference environment, enabling the VC integrator to evaluate the appropriateness of the VC to his/her application. This must contain a minimum of:

Electronic design automation (EDA) tool environment (identifying version details);

Release status of compatible ASIC reference libraries;

Sources and release status of any compilers and linkers required;

Itemization of any specific tools used for performance measurements (SNR, International Telecommunications Union (ITU) standards compliance, etc.);

Boundary conditions (range of operation, scalability) over which the stated performance claims can be validated.]

Use this style to describe a section {style = ip_normal}

3.1.1 EDA Tools

[This attribute lists all the EDA tools used by the VC provider, with versions and the completely description from the EDA environment.]

Use this style to describe a section {style = ip_table_text}

Table 3: EDA tools used in the chain.

Name	Version	Environmental Considerations
[Name of the EDA tool]	*[Version of the named EDA tool]*	*[Any necessary considerations]*

3.1.2 Support Tools

[This attribute details for any additional software tools.]

Use this style to describe a section {style = ip_table_text}

Table 4: Support tools used.

Name	Version	Description
[Name of the support tool]	*[Version of the named support tool]*	*[Any necessary considerations]*

3.1.3 Reference Designs

[This attribute lists the reference designs and reference platforms. (evaluation boards, design kits, evaluation models and evaluation chips).]

Use this style to describe a section {style = ip_normal}

4. Technology Reference Library

[Here should be presented the reference library used during the implementation.]

Use this style to describe a section {style = ip_table_text}

Table 5: Technology reference library.

Library Name	Version	Vendor
[Name of the library]	*[Version used]*	*[Organization that delivers the library]*

5. IP-Core Execution

[This section must explain how to configure the environment and how to execute the IP-core delivered.]

Use this style to describe a section {style = ip_normal}

6. References

[This section provides a complete list of all documents referenced elsewhere in this document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

[1] Use this style to list references *{style= ip_reference}*

[2] Document Title; <File name and link >;

[3] Title; Number (if applicable); Date; Institution, Team responsible for the document; Document link;

[4] …

Apêndice B – Documentação Produzida Durante o Estudo de Caso

ipPROCESS

Deployment Plan

LCD

LCD_Controller_and_Driver

Customer: Brazil-IP

Version : 1.4

Revision History

Date	Version	Description	Author
05/15/2008	1.0	First definitions of deployment plan.	André Feitoza de Mendonça
05/17/2008	1.1	Deployment activities added.	André Feitoza de Mendonça
05/30/2008	1.2	Deployment activities added.	André Feitoza de Mendonça
06/04/2008	1.3	Revision of the complete document.	Daniele Santos.
06/11/2008	1.4	Insertion of a table in the subection Humam Resource of the section Resources.	Daniele Santos.

1. Introduction

The need an easy integration of the IP-core into a wide variety of project, make the designers not only deliver the RTL code of the IP-core, a set of artifacts is also delivered to the client.

This document is intended to describe the deployment plan of the IP-core LCD Controller and Driver. The deployment plan defines when and how de artifacts will be delivered to the client and the necessary activities for this deployment.

1.1 Scope

The information related to the ways of distribute the artifacts is present in this document. The distribution activities are also described here.

1.2 Definitions, Acronyms and Abbreviations

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Artifact	Product that will be delivered.
Testbench	Plataform to test the IP-core.
RTL code	Product that will be delivered.

1.3 Document Structure

- Section 2 – Deployment Planning: it presents when and how the product will be deployed.
- Section 3 – Licensing: it presents the rules of using the artifacts delivered to the client.
- Section 4 – Training and Assistance: informs how will be done the assistance and training for user of the product, if necessary.
- Section 5 – Resources: gives all the human, software and hardware resources used in the deployment of the project.
- Section 6 – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Deployment Planning

This section presents the deployment plan, it shows when and how the IP-Core will be deployed and the necessary activities for this deployment.

2.1 Product Deployment

The deployment unit of the LCD Controller and Driver, i.e, unit that contains all artifacts that will be deployed with LCD Controller, will be deployed behind a special folder of the project repository and behind a CD.

2.2 Deployment Activities

This subsection lists the necessary activities for the deployment of the LCD Controller and Driver.

Table 2: Description of the deployment activity: Define Bill of Materials.

Deployment Activity	Define Bill of Materials
Description	Create a complete list of the materials that compounds the deployment unit of the LCD Controller and Driver .
Role	Deployment Manager
Input	None.
Output	Bill of Material Document.

Table 3: Description of the deployment activity: Develop Support Materials.

Deployment Activity	Develop Support Materials
Description	The necessary materials to use and to verify the LCD Controller and Driver are developed.
Role	Technical Writer
Input	The following documents: Functionality Description, System Logic Description, Implementation Information, Prototyping Manual and Verification Guide.
Output	Product Description and User Guide.

Table 4: Description of the deployment activity: Create Deployment Unit.

Deployment Activity	Create Deployment Unit
Description	The artifacts are assembled to create a unit that will be delivered to the client.
Role	Deployment Integrator
Input	Components, Verification Deployment, Prototype Deployment, Guide, Product Description.
Output	Deployment Documentation and Deployment Unit.

Table 5. Description of the deployment activity: Manage Beta Test.

Deployment Activity	Manage Beta Test
Description	To manage the beta test of the Deployment Unit.
Role	Deployment Manager
Input	Deployment Unit.
Output	Changes Request.

Table 6. Description of the deployment activity: To Available Product.

Deployment Activity	To Available Product.
Description	To available product for its users.
Role	Deployment Manager
Input	Deployment Unit.
Output	None.

2.3 Schedule

The table below shows the schedule for the activities listed in theprevious subsection.

Table 7: Deployment Schedule.

Deployment Activity	***Begin***	***Deadline***
Define Bill of Materials	05/30/2008	4 days
Develop Support Materials	06/03/2008	10 days
Create Deployment Unit	06/14/2008	4 days
Manage Beta Test	06/18/2008	3 days
To Available Product.	06/22/2008	1 days

3. Licensing

The distribution of substantively modified versions of the LCD Controller and Driver is prohibited without the explicit permission of the copyright holder. Distribution of the LCD Controller and Driver or its derivative in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.

4. Training and Assistance

The client, to use the LCD Controller and Driver correctly, must have a minimum knowledge in the semiconductor area. The documents of the LCD Controller and Driver are organized, which make the search of information quick. The easy understanding of the artifacts eliminates any client training.

5. Resources

This section shows the resources necessary to distribute the artifacts to the users.

5.1 Human Resources

Table 8: Human Resources.

Role	**Specific Responsibilities or Comments**
Technical Writer	Develop the support materials.
FPGA Implementer	Develop the necessary environment

	(scripts, binary files, prototype manual) for the user to prototype the LCD Controller and Driver.
Test Designer	Develop a verification guide, document that contains necessary information for the users to verify the LCD Controller and Driver.
Test Implementer	Develop an environment for the verification of the LCD Controller and Driver.

5.2 Software Resources

The development of the most of the artifacts uses texts editors, such as Word and Context. Except the RTL Code and the Testbench, that use the IDE Eclipse and Quartus tool.

5.3 Hardware Resources

The computers must have, at least, 512 MB RAM memory to allow the heavy processing of the synthesis RTL tools. This is the only one requirement to the computers in this project.

6. References

[1] SANTOS, Daniele. Deployment discipline.

[2] RUP. *RUP Home page*. Available in: http://www-306.ibm.com/software/awdtools/rup. Access in: 06/04/2008.

ipPROCESS

Bill of Material

LCD

LCD_Controller_and_Driver

Customer: Brazil-IP

Version : 1.1

Revision History

Date	Version	Description	Author
05/24/2008	1.0	First definitions of the Bill of Material document.	André Feitoza de Mendonça
06/04/2008	1.1	Revision of the complete document.	Daniele Santos

1. Introduction

The necessity of develop a IP-core capable to be applied in a wide variety of projects demands additional information to be released. An unit is delivered containing, besides the RTL code, integration´s information.

1.1 Scope

This document presents a description of the components included in the LCD Controller and Driver deployed. Additional information, such as identification codes and version of the artifacts are also present here.

1.2 Definitions, Acronyms and Abbreviations

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Artifact	Product that will be delivered
HDL	Hardware Description Language,
Testbench	Plataform to test the IP-core
RTL code	A register level description of the IP-core

1.3 Document Structure

- Section 2 – Version Description: it lists all artifacts that form the actual version of the LCD Controller and Driver.
- Section 3 – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Version Description

This section shows the components that form the deployed unit of the LCD Controller and Driver product.

2.1 Inventory of Material

The artifacts will be available in a CD. The end of the name files in this CD indicates their version. For instance: LCD_Controller_and_Driver_User_Guide_1_1.

2.2 Inventory of Files

The following table presents the files released to the client. It is also provided a short description of this files.

Table 2: Inventory of the LCD Controller and Driver Files.

Artifact	*Version*	*Description*
LCD_Controller_and_Driver Product Overview.	1.0	This document presents a overview of the LCD Controller and Driver functionalities.
LCD Controller and Driver User Guide.	1.1	It presents a detailed informations about the IP-Core, to facilitate its use.
LCD Controller and Driver Deliverable Documentation.	1.0	This document presents a list of the components that form the deployed product.
RTL Code	1.0	The implementation of the IP-core in a HDL language, in this case, Verilog.
Prototyping Deployment	1.0	The enviroment of the FPGA prototype of the LCD Controller and Driver. It contains synthesis scripts, cossimulate scripts and prototype manual.
Verification Environment	1.0	An environment for verification of the LCD Controller and Driver. It contains testbench, verification scripts and test cases.

2.3 Changes

The change occured only in the LCD Controller and Driver User Guide, its informations were divided in the following documents: LCD_Controller_and_Driver Product Overview, LCD Controller and Driver User Guide and LCD Controller and Driver Deliverable Documentation.

2.4 Known Errors and Problematic Features

No errors are found in the LCD Controller and Driver until now

3. References

[1] SANTOS, Daniele. Deployment discipline.

[2] RUP. *RUP Home page*. Available in: http://www-306.ibm.com/software/awdtools/rup. Access in: 06/04/2008.

ipPROCESS

Product Overview

LCD

LCD_Controller_and_Driver

Customer: Brazil-IP

Version : 1.0

Revision History

Date	Version	Description	Author
06/15/2008	1.0	Elaboration of the document.	Daniele Santos

1. Introduction

This document presents an overview about the LCD Controller and Driver. Its purpose is to help its potential users in the adequacy of this product before its aquisition.

1.1 Scope

The sections contained in this document presents informations about the capabilities and features of the LCD Controller and Driver, beyond informations about the standards aplicable to this product, the necessary requirements for integration of this IP-core in the SoC project and finally, the level at which the VC has been implemented.

1.2 Document Structure

This document is organized with the following sections:

- Section 2 – Product Informations: this section presents informations about the capabilities, features, mode of operation and the standards compliance with the LCD Controller and Driver.
- Section 3 – Integration Informations: this section provides informations about the integration of the LCD Controller and Driver in a SoC project.
- Section 4 – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.3 Contact Information

For more details about the product described in this document, please contact the responsible people for it.

Table 1: Contact Information.

Name	*Brazil-IPii*
Address	*Cidade Universitária - 50732-970 Recife-PE*
Division	*Centro de Informática*
Primary Contact	*Daniele Santos*
Phone	*+55 81 2126-8430*
E-mail	*dps@cin.ufpe.br*
URL	*www.cin.ufpe.br/~dps*

2. Product Informations

This section presents informations about the capabilities, features, modes of operations and standars compliance with the LCD Controller and Driver.

2.1 Product Capabilities

The LCD Displays are used in many electronics project to facilitate the interaction of the user with the project. LCD Displays can show relevant information for the use of project, as well as returning to the user results gotten in some type of processing; the current have a controller, an integrated circuit that is in the proper display.

The LCD Controller and Driver is an alphanumeric display LCD controller, it contains 32 (two lines of 16) characters of 5x8 or 5x10 bits. It executes various instructions; it can to clear the display, to move the cursor, to change the lines number of the display, change the printing mode of characters and them size, beyond this, it is capability to print two hundred forty different characters.

Figure 1: LCD Display.

2.2 IP-Core Features

The main features of the LCD Controller and Driver are:

- 5 x 8 and 5 x 10 dot matrix possible;
- 2-line display of up characters per line;
- Character generator ROM: 240 characters.

2.3 Modes of Operation

The LCD Controller and Driver posses the following modes of operation:

- One or Two lines;
- Characters printed in the occidental and oriental mode;
- Characters printed in 5x8 or 5x10 dots matrix.

2.4 Standards Compliance

The table below presents the standards compliance in the LCD Controller and Driver.

Table 2: List of the applicable standards.

Organization	Name	Version	Revision
Hitachi	HD44780	Not defined.	0.0
-	ASCII	Not defined.	Not defined.

3. Integration Informations

This section presents informations related to the integration of the LCD Controller and Driver in a plataform SoC.

3.1 Qualification Levels

The Ip-core described in this document, the LCD Controller and Driver, was implemented in the HDL (hardware description language) languages SystemC and Verilog. The LCD Controller also prototyped in FPGA (Field Programable Gate Array).

3.2 Integration Requirements

The communication of the LCD Controller and Driver must be done behind the OCP-IP interface.

4. References

[1] LCD Controller and Driver Vision Document; Version 1.7.

[2] LCD Controller and Driver Requirements Specification; Version 1.9.

ipPROCESS

Deliverable Documentation

LCD

LCD_Controller_and_Driver

Customer: Brazil-IP

Version : 1.0

Revision History

Date	Version	Description	Author
06/10/2007	1.0	Elaboration of the document.	Daniele Santos

Approvals List

Name	Role
Daniele Santos	Developer

1. Introduction

For a good use of the LCD Controller and Driver, an artifacts set is deployed for its users. This documents presents a description of the artifacts deployed with the LCD Controller and Driver.

1.1 Scope

This document describes the artifacts deployed with the LCD Controller and Driver, describing its availability, history versions and other aspects. Also is presented standard terms and conditions to use the LCD Controller and Driver.

1.2 Document Structure

This document is structured with the following sections:

- Section 2 – Deliverable Product: this section provides a detailed descriptions of the products deployed with the LCD Controller and Driver.
- Section 3 – Version Changes: this section presents the changes occured in the actual version in relation to the previous version.
- Section 4 – Standard Terms and Conditions: it contains standard terms and conditions governing the evaluation and subsequent incorporation of the LCD Controller and Driver into systems.
- Section 5 – Contact: it informs the contacts for more information about the deployed product.
- Section 6 – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Deliverable Product

This section presents the artifacts that are deployed for the users of the LCD Controller and Driver. It shows informations about the availability of the deployed IP-Core, its known bugs and problems features, beyond the informations about transfer package.

2.1 Deliverables List

All artifacts that compose the deployed product are listed in the table below.

Table 2: List of the VC Deliverables.

Name	Filename	Format	Description	Comply
Product Overview	LCD_Controller_and_Driver_Product_Overview.doc	.doc	It presents an overview of the LCD Controller and Driver functionalities. The deployed version is 1.0	☒
Deliverable Documentation	LCD_Controller_and_Driver_Deliverable_Documentation.doc	.doc	It presents a list with the artifacts deployed with the LCD Controller and Driver and it explain some characters them like know bugs, version changes and standard term, for example.	☒
User Guide	LCD_Controller_and_Driver_User_Guide.doc	.doc	It presents a detailed informations set about the LCD Controller and Driver, explaining its architecture, implementation, verification and other aspects.	☒
Prototyping Environment	LCD_Controller_and_Driver_Prototyping_Environment.zip	.zip	It contains the necessary files for the user to prototype the LCD Controller and Driver. It contains source code, binary files, synthesis and cossimulate scripts.	☒
Verification Environment	LCD_Controller_and_Driver_Verification_Environment.zip	.zip	It contains the necessary enviroment for the user to verify the LCD Controller and Driver. For this, are available verification componentes, like testbench, and test cases.	☒

2.2 Availability

The LCD Controller and Driver will be available for transfer after its deployment unit, set of artifacts that will be deployed, is tested. The goal of this test is to verify the usability of the deployment Unit. This test must realized in the end of the moth of June. So, the LCD Controller and Driver will be available in the home of the month of July.

2.3 Version History

The table below presents the version history of the LCD Controller and Driver.

Table 3: VC version history.

Version	Description	Date
2.0	Version with all functionalities of the LCD Controller and Driver implemented.	2007/07/16

2.4 Known Bugs and Problematic Features

No errors are found in the LCD Controller and Driver until now.

2.5 Transfer Package Information

2.5.1 Package

VC Name

LCD Controller and Driver

VC ID

LCD_v1.0

2.5.2 Package Information

Archive Format

Zip.

3. Version Changes

How this is the first version of the LCD Controller and Driver, there aren´t changes in this version in relation to a previous version.

4. Standard Terms and Conditions

4.1 Support Models

Clasification

- Maintenance
- Application support

4.2 Fee Models

Clasification

Other.

4.3 Applicable Patents

No patents.

5. Contact

This section presents the contact for more informations about the product described in this document.

5.1 IP Provider Contact

Table 3: Contact Information.

Name	Brazil-IP Network.
DUNS	None
Address	Centro de Informática UFPE - Caixa Postal 7851 - Cidade Universitária - 50732-970 - Recife/PE Brasil.
Division	GRECO - Grupo de Engenharia da Computação.
Primary Contact	Daniele Santos
Phone	+55 81 2126-8430
E-mail	dps@cin.ufpe.br
Fax	+55 81 2126-8430
URL	www.brazilip.org.br

6. References

[1] LCD Controller and Driver Bill of Materials; Version: 1.1

ipPROCESS

User Guide

LCD

LCD_Controller_and_Driver

Customer: Brazil-IP

Version 1.2

Revision History

Date	Version	Description	Author
07/12/2007	1.0	Description of general information.	Daniele Santos
07/16/2007	1.1	Finish of the document.	Daniele Santos/Rebeka Gomes
06/19/2008	1.2	Delete of the section Deliverable Documentation and revision of the other sections.	Daniele Santos

Approvals List

Name	Role
Daniele Santos	Developer

1. Introduction

The purpose of this document is to present to the user of the LCD Controller and Driver the necessary information for good use its.

1.1 Document Overview

This document is composed by the following sections:

- Section 2 – General Information: It provides the general information about the LCD Controller and Driver.
- Section 3 – System Logic Description: It contains the external description of how LCD Controller and Driver functions and nature of its interfaces on the system logic level.
- Section 4 – Integration Information: It provides necessary information to the LCD integrator.
- Section 5 - Programmer's Reference Manual: It presents information for programmers that will use the LCD Controller and Driver.
- Section 6 - Implementation: It contains information about the LCD implementation.
- Section 7 – Verification: It describes the necessary structure to the verification effort.
- Section 8 – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.2 Document Structure

This document is organized with the following sections:

- Section 2 – Product Informations: this section presents informations about the capabilities, features, mode of operation and the standards compliance with the LCD Controller and Driver.
- Section 3 – Integration Informations: this section provides informations about the integration of the LCD Controller and Driver in a SoC project.
- Section 4 – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.3 Definitions, Acronyms and Abbreviations

This table is used for specify any terms, acronyms and abbreviations used some times in this text.

Table 1: Definitions, acronyms and abbreviations.

Term	Description
Reference model	It is a IP-Core representation developed in a abstract level.
ip-PROCESS	Developed process for Soft IP with prototyping in FPGA.
Testbench	It is structure developed for the IP-Core simulation.

1.4 Contact Information

For more information about the product described in this document, please contact the responsible people for it.

Table 0.1: Contact Information.

Name	*Brazil-IP*
Address	*Cidade Universitária - 50732-970 Recife-PE*
Division	*Centro de Informática*
Primary Contact	*Daniele Santos*
Phone	*+55 81 2126-8430*
E-mail	*dps@cin.ufpe.br*
Fax	*+55 81 2126-8430*
URL	*www.brazilip.org.br*

2. General Informations

The LCD Displays are used in many electronics project to facilitate the interaction of the user with the project, with the LCD, it can show relevant information for the use of project, as well as returning to the user results gotten in some type of processing; the current have a controller, an integrated circuit that is in the proper display. The LCD Controller and Driver is an alphanumeric display LCD controller, it contains 32 (two lines of 16) characters of 5x8 or 5x10 bits.

The LCD Controller and Driver execute various instructions; it can to clear the display, to move the cursor, to change the lines number of the display, change the printing mode of characters and them size, beyond this, it is capability to print two hundred forty different characters.

Figure 1: LCD Display.

2.1 Functionality Overview

This subsection presents the LCD's functionality behind different aspects, they are: its architecture, its modes of operation and finally its equivalent parts or cores.

2.1.1 Architecture

The LCD's architecture is formed by four main modules, they are: DecoderCommand, TopCharacter, Cursor and Driver.

The DecoderCommand module is responsible for the decoding of the commands received by the LCD and the confirmation of the execution of the last received command; depending of the command received, this module communicate itself with the TopCharacter or Cursor module for one them process the received instruction.

The TopCharacter module is responsible for manipulate the characters that will be printed on the display; this module is composed by following internal modules: PrintChar and DataDisplayArray. The PrintChar module is responsible by the necessary process for to store in the RAM memory the mapping of the characters that will be printed on the display, it posses a ROM memory that has stored the mapping of all characters that can printed on the display; the DataDisplayArray is a RAM memory, responsible for store the mapping characters that are been printed on the display.

The Driver module controls the print frequency on the display of the characters stored in the RAM memory from DataDisplayArray module.

The cursor module is responsible by functionalities related with cursor's manipulation.

The LCD Controller and Driver communicates with the user behind the CommandInterface_DecoderCommand protocol.

The CommandInterface_DecoderCommand protocol represents the interaction between the user and the DecoderCommand module. This interaction is done through the sending of the command to the LCD Controller and Driver and the confirmation of the execution of the last command read by the LCD Controller and Driver.

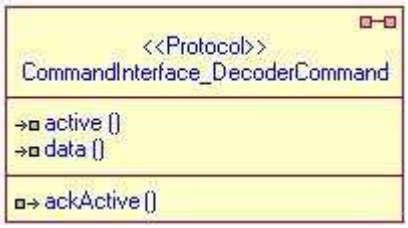

Figure 2: CommandInterface_DecoderCommand.

- bool active: It indicates that command sent by the user can be read.
- sc_uint<8> data: Command sent for the LCD Controller and Driver by the user.
- bool ackActive: It indicates for the user the end of the execution of the last command read by the LCD Controller and Driver.

The ReadMaps protocol represents the interaction between the LCD Controller and Driver and the display in relation to the printing of the characters stored in the DataDisplayArray module.

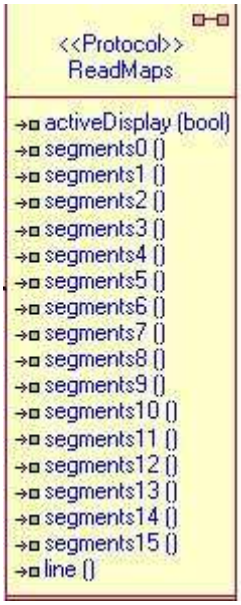

Figure 3: ReadMaps protocol.

Out signals:

- activeDisplay: It indicates that the character's mapping stored in the ROM memory from the DataDisplayArray module are been read and printed on the display.
- segment 0 ...15 (0 to 15): the signals are responsible to send the characters mapping from DataDisplayArray module for the display, these signals are sc_uint<5>.
- line: It indicates: It indicates which line of the Character's mapping is been printed on the display.

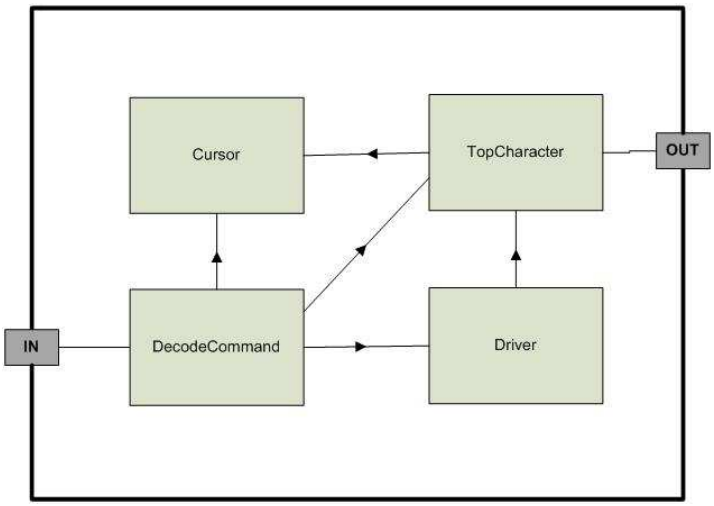

Figure 4: LCD Architecture.

2.1.2 Modes of Operation

The LCD Controller and Driver possess two modes of operation, occidental mode and oriental mode. In the occidental mode, the characters are printed from the left for the right direction, printing from the first line for the second line, if the display has two lines. In the oriental mode, the characters are printed from the right to the left direction, if the display has two lines, from second line to the first.

2.2 IP-core Features

Some characteristics of the LCD Controller and Driver differentiate it of other LCD's controller, this features are listed below:

- 5 x 8 or 5 x 10 dot matrix possible;
- Character generator ROM: 240 characters;
- 2-line display of up character per line;
- Wide range of instructions functions:

  - Clear Display, Return Home, Entry Mode Set, Cursor or Display Shift, Function Set, Writing Characters, Display On/Off Control.

2.3 Standards Compliance

The LCD Controller and Driver's instructions obey the HITACHI standard, for more information, look the table below.

Table 2: List of the applicable standards.

Organization	Name	Version	Revision
Hitachi	HD44780	Not defined.	0.0
-	ASCII	Not defined.	Not defined.

2.4 Qualification Levels

The LCD Controller and Driver deployed is implemented in Verilog RTL and its was prototyped in FPGA.

3. System Logic Description

This section presents the external description of LCD's functioning and its interfaces on the system logic level.

3.1 Functionality

The LCD Controller and Driver possesses the following functionalities:

- Clear Display: The characters printed on the display disappear and the cursor goes to the left edge of the display (in the first line) if the LCD's mode of operation is occidental, if its mode is oriental, the cursor goes to the last position of the last line.
- Return Home: Returns the cursor to its original position if it was shifted. If the LCD's mode of operation is occidental mode, the original position of cursor is the first position of the first line, if the mode of operation is oriental mode; the cursor goes to last position of the last line.
- Cursor Shift Right: Shifts the cursor to the right direction.
- Cursor Shift Left: Shifts the cursor to the left direction.
- Entry Mode Set: Sets the direction of characters printing and the cursor displacement, from the right to the left direction (occidental mode), from the left for the right direction (oriental mode).
- Display On/Off: Enable or disable the display functioning.
- Cursor On/Off: Enable or disable the cursor printing in the display.
- Function Set: Sets the number of display lines, 0or 2 lines, the character font, 5 x 8 or 5 x 10 dots matrix possible.
- Print Char On Display: Print characters on the display according the code sent by the user.
- Blink: When requested, it blinks the cursor position character.

3.2 Interfaces

This subsection shows the interfaces of the LCD Controller and Driver, explaining its ports and structures.

3.2.1 Port Descriptions

The table below shows behind the system-level view the LCD Controller and Driver's ports

Port Name	Interface Name	Port Type Control	Port Type Data	Transaction Type	Port Flow	Description
clock	-	-		System clock.	Independent signal.	
reset	-	-		System reset.	Independent signal.	
active	CommandInterface_DecoderCommand	Enable the DecoderCommand functioning.	bool	Communication with the DecoderCommand module.	Sending of activation order to the DecoderCommand module.	It indicates command ava in data port c read.
data	CommandInterface_DecoderCommand	-	sc_uint<8>	Communication with the DecoderCommand module.	Sending of data for DecoderCommand module.	It contains command sent user.
ackActive	CommandInterface_DecoderCommand	It indicates the end of the execution of the last command received by the DecoderCommand.	bool	Communication with the user.	Sending of the confirmation of the last command execution received.	It indicates th of the executi the last com read by the Controller Driver.
ackWrite	PrintCharacter	It indicates the end of the execution of the last command received by the PrintChar module.	bool	Communication with the DecoderCommand module.	Sending of the confirmation of the last command execution received by the TopCharacter module.	It indicates the end of a operation related to the characters.
code	PrintCharacter	-	sc_uint<8>	Communication with the PrintChar module.	Sending of data for Top Character module.	The code related to the command received by LCD Controller.

write	PrintCharacter	Enable the PrintChar module decode its received command.	bool	Communication with the PrintChar module.	Sending of activation order to the PrintChar module.	It indicates that the command received by the PrintChar module must be executed.
blink	Blink	Enable the blink of the cursor position.	bool	Communication with the PrintChar module.	Sending of blink command to the DataDisplayArray module.	It indicates the cursor position character must blink.
ackBlink	Blink	It confirms the received of the blink signal.	bool	Communication with the DataDisplayArray module.	Sending of the confirmation of the blink command received by DataDisplayArray module.	It indicates that blink command was received.
ocidentalMode	ModeSet	Enable the occidental mode of the LCD.	bool	Communication with the DataDisplayArray module.	Sending of the occidental mode to the DataDisplayArray module.	It sets the LCD to function in occidental mode.
orientalMode	ModeSet	Enable the oriental mode of the LCD.	bool	Communication with the DataDisplayArray module.	Sending of the oriental mode to the DataDisplayArray module.	It sets the LCD to function in oriental mode.
ackOcidental	ModeSet	It confirms that the occidental command was received.	bool	Communication with the DecoderCommand module.	Sending of the confirmation of the occidental mode command received by DataDisplayArray module.	It confirms that the controller was adjusted to function in occidental mode.
ackOriental	ModeSet	It confirms that the oriental command was received.	bool	Communication with the DecoderCommand module.	Sending of the confirmation of the oriental mode command received by DataDisplayArray	It confirms that the controller was adjusted to function in oriental mode.

					module.	
linesNumber	NumberLines	It indicates the lines number of the display.	sc_uint<2>	Communication from the DecoderCommand module for the DataDisplayArray module.	Sending the lines number for the DataDisplayArray module.	It sets the lines number of the display.
ackLinesNumber	NumberLines	It indicates that the information of lines number was received by the DataDisplayArray module.	bool	Communication from the DataDisplayArray module for the DecoderCommand module.	Sending of the confirmation of the lines number was received by DataDisplayArray module.	It confirms that the lines number of the display was adjusted.
cursorOn	CursorOnOff	It indicates that the cursor must be printed on the display.	bool	Communication from the DecoderCommand module for the DataDisplayArray module.	Sending the printing cursor request.	It enables to print the cursor on the display.
ackCursorOn	CursorOnOff	It indicates that the information of printing cursor was received by the DataDisplayArray module.	bool	Communication from the DataDisplayArray module for the DecoderCommand module.	Sending of the confirmation of the printing cursor request was received by DataDisplayArray module.	It confirms the printing cursor request was received.
shiftRight	ShiftCursor	It indicates the command for the cursor to shift for the right direction.	bool	Communication from the DecoderCommand module to the Cursor module.	Sending of the command for the cursor to shift to the right direction.	It sends the command from DecoderCommand module to the Cursor module to shift the cursor to the right direction.
shiftLeft	ShiftCursor	It indicates the command for the cursor to shift for the left direction	bool	Communication from the DecoderCommand module to the Cursor module.	Sending of the command for the cursor to shift to the left direction.	It sends the command from DecoderCommand module to the Cursor module to shift the cursor

						to the left direction.
returnHome	ShiftCursor	It indicates the command for the cursor to shift for the first position of the display.	bool	Communication from the DecoderCommand module to the Cursor module.	Sending of the command for the cursor to go to the first position of the display.	It sends the returnHome command from DecoderCommand module to the Cursor module.
ackShiftRight	ShiftCursor	It indicates that the command of the shift right the cursor was realized.	bool	Communication from the Cursor module to the DecoderCommand module.	Sending of the confirmation that the shift right cursor command was realized.	It sends the confirmation of the cursor displacement to the right direction.
ackShiftLeft	ShiftCursor	It indicates that the command of the shift left the cursor was realized.	bool	Communication from the Cursor module to the DecoderCommand module.	Sending of the confirmation that the shift left cursor command was realized.	It sends the confirmation of the cursor displacement to the left direction.
ackReturn	ShiftCursor	It indicates that the return home command was realized.	bool	Communication from the Cursor module to the DecoderCommand module.	Sending of the confirmation that the return home command was realized.	It sends the confirmation of the execution of the ReturnHome command by Cursor module.
displayOn	DisplayOnOff	It enables the functioning of the LCD Controller and Driver.	bool	Communication from the user to the DecoderCommand module.	Sending of the request for the LCD Controller to function.	It enables the display functioning.
cursorPosition	value	It indicates the position cursor.	sc_uint<5>	Communication from the Cursor module for the TopCharacter module.	Sending of cursor position.	It indicates the cursor position.
activeDisplay	ReadMaps	It indicates that the character mapping is been printed o the display.	bool	Communication from the TopCharacter module with the display.	Indicating that the mapping characters are printed on the display.	It indicates to the display that the characters mapping are been printed.

line	ReadMaps	-	sc_uint<5>	Communication from the TopCharacter module with the display.	Indicating that line of the mapping characters are printed on the display.	It contains the character line that is been printed on the display.
segment0	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the first character that will be printed on the display.	It contains the mapping of the first character that is printed on the display related to the value contained in the line port.
segment1	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the second character that will be printed on the display.	It contains the mapping of the second character that is printed on the display related to the value contained in the line port.
segment2	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the third character that will be printed on the display.	It contains the mapping of the third character that is printed on the display related to the value contained in the line port.
segment3	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the forth character that will be printed on the display.	It contains the mapping of the fourth character that is printed on the display related to the value contained in the line port.
segment4	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the fifth character that will be printed on the display.	It contains the mapping of the fifth character that is printed on the display related to the value contained in the line port.
segment5	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the	Sending of the mapping of the sixth character that will be printed on	It contains the mapping of the sixth character that is printed on the display

				display.	the display.	related to the value contained in the line port.
segment6	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the seventh character that will be printed on the display.	It contains the mapping of the seventh character that is printed on the display related to the value contained in the line port.
segment7	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the eighth character that will be printed on the display.	It contains the mapping of the eighth character that is printed on the display related to the value contained in the line port.
segment8	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the nineth character that will be printed on the display.	It contains the mapping of the ninth character that is printed on the display related to the value contained in the line port.
segment9	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the tenth character that will be printed on the display.	It contains the mapping of the tenth character that is printed on the display related to the value contained in the line port.
segment10	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the eleventh character that will be printed on the display.	It contains the mapping of the eleventh character that is printed on the display related to the value contained in the line port.
segment11	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the twelfth character that will be printed on the display.	It contains the mapping of the twelfth character that is printed on the display related to the value

						contained in the line port.
segment12	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the thirteenth character that will be printed on the display.	It contains the mapping of the thirteenth character that is printed on the display related to the value contained in the line port.
segment13	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the fourteenth character that will be printed on the display.	It contains the mapping of the fourteenth character that is printed on the display related to the value contained in the line port.
segment14	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the fifteenth character that will be printed on the display.	It contains the mapping of the fifteenth character that is printed on the display related to the value contained in the line port.
segment15	ReadMaps	-	sc_uint<5>	Communication from the DataDisplayArray module with the display.	Sending of the mapping of the sixteenth character that will be printed on the display.	It contains the mapping of the sixteenth character that is printed on the display related to the value contained in the line port.

3.2.2 Protocols

The LCD Controller and Driver communicates with the user and with the display behind defined protocols, it internal modules communicates them behind protocols. The external LCD protocols are: CommandInterface_DecoderCommand, responsible by the communication behind the LCD Controller and the user, and the ReadMaps, this is responsible for the communication between the LCD Controller and the display.

The protocols used by the internal modules are listed below:

- AcessDDRAM: It represents the access to the RAM memory of the DataDisplayArray module.
- Blink: This protocol represents the interaction between the DecoderCommand module and the DataDisplayArray module in relation of the blink of the cursor position character.
- CursorOnOff: It represents the interaction between the DecoderCommand module and the DataDisplayArray module in the relation to the cursor's printing.
- CursorRight: This protocol represents the interaction between the PrintChar module and the Cursor module in relation to the sending of the command for the shift the cursor for the right direction.
- DisplayOnOff: It represents the interaction between the DecoderCommand module
- ModeSet: This protocol represents the interaction between the DecoderCommand module and the PrintChar module in relation to the order of memory's position that will be stored the character's mapping.
- NumberLines: This protocol represents the interaction between the DecoderCommand module and the PrintChar module, Cursor module and Driver module in relation of the amount of the character's lines that it can be printed on the display.
- PrintCharacter: It represents the interaction between the DecodeCommand module and the PrintChar module. The purpose of this interaction is the storage of the character's mapping relative to the codes (that it represents a character) sent by the user in the mapCharacters memory from the DataDisplayArray module.
- ShiftCursor: This protocol represents the interaction between the DecoderCommand module and the Cursor module in relation to the sending of the commands for to shift the cursor.
- Value: This protocol represents the interaction between the Cursor module and the DataDisplayArray module in relation to the sending of the cursor's position.

3.3 Integration Requirements

There isn't a special integration requirement for the LCD Controller and Driver.

4. Integration Information

This section presents important information about the development of the LCD to who will use LCD in other systems.

4.1 Reference Environment

Here has a specification of the environment necessary to development of the LCD.

4.1.1 Tools, Flows and Methodologies

First the LCD and its test structure were implemented in C/C++ using the SystemC library utilizing Eclipse with gcc compiler.

To make the synthesis the Cynthesizer was used. In the prototyping phase the EDA Quartus was used to simulate and prototype the LCD in a FPGA.

The methodology utilized was ipProcess, a process for Soft IP with prototyping in FPGA that was developed in Universidade Federal de Pernambuco.

- **EDA Tools**

The table below shows the EDA tools used in the development of the LCD Controller and Driver.

Table 4: EDA tools used in the chain.

Name	Version	Environmental Considerations
Rational Rose Real Time	2003	UML Case tool from Rational Suite Enterprise
Eclipse	3.2	SUN integrated development environment (IDE)
Quartus	5.0	Altera Corporation EDA
Cynthesizer	3.3	Tool to synthesize SystemC code in Verilog code provide by FORTE Design Systems.

- **Support Tools**

Some tools that provide support to control version and process managing was used in the LCD conception.

Table 5: Support tools used.

Name	Version	Description
CVS	-	Control Version tool.
dotProject	2.0.4	Tool for software management.

- **Reference Designs**

The LCD Controller and Driver was prototyped in Cyclone II Evaluation Board.

4.2 How to Configure the Reference Environment

This section explains how to configure the environment to execute the LCD and its test structure.

4.2.1 IP-core Execution

To execute the LCD you need open the Quartus EDA, in the menu File choose the option Open Project, put the path that the LCD source was saved and select the file LCD_Controller_and_Driver.

To simulate it you should open the option New File and create a Waveform file. With a double click in this file all disposed signals will be listed, so choose the signals you want to view, initialize the input values and then click in the simulation button.

To prototype in a FPGA you should click in the Programmer button and choose the hardware that was in communication with the FPGA, finally click in the start button to load the LCD in the FPGA.

4.2.2 Verification Execution

To execute the LCD verification structure you should following the same steps of the IP-core execution, but open the project file LCD_Verification that should be in the folder Verification.

5. Programmer's Reference Mannual

This section provides necessary information for the programmers that will use de LCD Controller and Driver.

5.1 Instruction Set

The LCD Controller and Driver supports the following instructions set:

Table 6: Instruction Set.

Name	Opcode	Description
Display On/Cursor On	0X0E	Enable the display functioning and the cursor is printed on the display.
Display On/Cursor Off	0X0C	Enable the display functioning and the cursor isn't printed on the display.
Occidental Mode	0X06	Enable the direction of the characters printing from the left for the right direction.
Oriental Mode	0X04	Enable the direction of the characters printing from the right for the left direction, from the last line for the first line.
Display – one line\ 5X8 dots matrix	0X05	Enable the display to function with one line and to print characters in the format 5X8 dots matrix.
Display – one line\ 5X10 dots matrix	0X0A	Enable the display to function with one line and to print characters in the format 5X10 dots matrix.
Display – two lines	0X07	Enable the display to function with two lines.
Clear Display	0X01	Clear the displays, i.e., the characters printed on the display are erased ant the cursor shifted for the first position.
Return Home	0X02	Displacement the cursor for the first position of the display.
Cursor Shift Left	0X0d	Shift the cursor for the left direction.

Cursor Shift Right	0X09	Shift the cursor for the right direction.
Display off	0x0d	Disable the display functioning.
Blinking	0X0f	Enable the blink of the cursor position character.

To print characters on the display, the command is according the desired character, obeying the figure below:

Figure 2: Character mappings.

For a good functioning of the LCD Controller and Driver, it is necessary sets some parameters like number lines, character font and mode of operation, for this, is necessary to obey the following instruction order:

- Enable the display, for this, send the **0X0C** 0r **0X0E** command (see the difference between them in beginning of this section).
- After, sets the number lines and the characters font, for this, send the **0X05**, **0X07** or **0X0A** command.

- Sets the display operation model, **0X06** for the occidental mode, or **0X04** for the oriental mode.
- Now, the LCD Controller is enabled for it functioning.

6. Implementation

The figure below shows how the LCD Controller and Driver is structured:

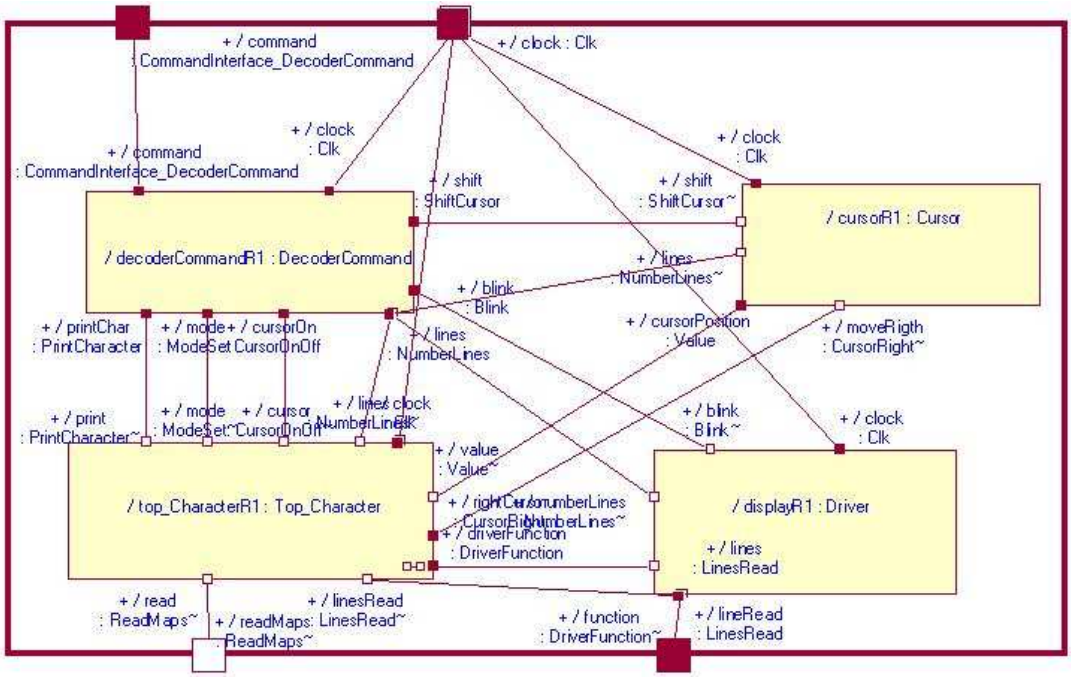

Figure 7: Structure model of the LCD Controller and Driver.

6.1 Performance Information

This subsection shows performance information about the LCD Controller and Driver.

6.1.1 Frequency

The table below shows the values of the minimum, typical and maximum frequency obtained in the LCD Controller and Driver.

Table 2: Frequency values for the VC.

Typical
56,66 MHz

6.2 Form Information

This section describes the hardness of the LCD Controller and Driver.

6.2.1 Bit Count

The table below shows the number of memory bits in the LCD Controller and Driver.

Table 3: Bit count for the VC.

Typical
130.464

6.2.2 Register Count

The table below shows the number of registers presents in the LCD Controller and Driver.

Table 4: Register count.

Typical
978

6.2.3 Pin Count

This subsection defines the number of pins that access the LCD Controller and Driver.

Table 5: Pin count.

Input	4
Output	19
Bi-directional	0
Test	0
Other	0

6.3 Technology Reference Library

The table below shows the reference library used during the LCD implementation.

Table 6: Technology Reference Library

Library Name	Version	Vendor
SystemC	2.1	Open SystemC Initiative (OSCI)

7. Verification

This section describes the structure needed to the verification effort of the LCD Controller and Driver.

7.1 Verification Approach

The verification technique used in the LCD Controller and Driver consisted in to generate a set of inputs for the DUV and the Reference Model and to compare the output responses of the DUV to those of a reference designer, for this, was build a testbench. This technique was used in the LCD implementation in systemC and in RTL.

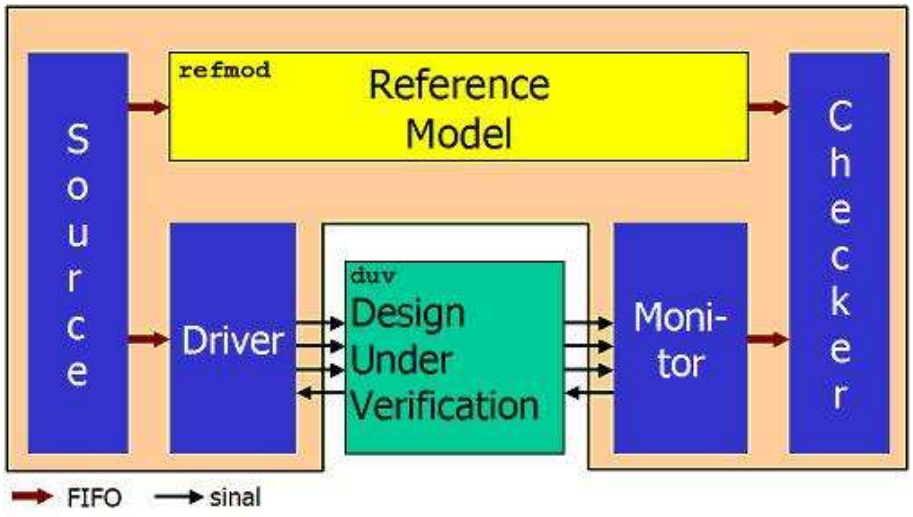

Figure 9: Verification Approach.

For the verification of the LCD prototyped in FPGA, other verification technique was used; in this technique, we don't have a reference model, we simulated the LCD in RTL level and it outputs was stored in a ROM memory, the same inputs used in the LCD simulation in RTL are used in the LCD prototyped and its outputs are compared with the ones stored in the ROM memory previously cited, obtaining the verification results behind the programmable leds of the Cyclone II FPGA.

7.2 Environmental Needs

This section shows the environmental needs to reproduce the verification environment used in the LCD Controller and Driver.

7.2.1 Base hardware System

The following table sets forth the system resources for the test effort presented in the Verification Approach section.

Table 7: System resources for verification effort.

Resource	Quantity	Name or Type
FPGA	1	Altera
Processor 1.5 GHz	1	No preferred type.

7.2.2 Base Software Elements in the Verification Environment

The following base software elements are required in the verification environment for execute the LCD verification described in the Verification Approach section.

Table 8: Software elements for verification effort.

Software Name	Version	Type and other notes
Quartus	5.0	
Eclipse	3.0	C/C++ Development tools
gcc compiler	3.3.6	
SystemC, C++ library	2.0.1	

7.3 Tools Flow

For reproduce the LCD verification, it is necessary to use some tools. For the verification of the LCD in the behavior level, it is used the IDE´s Eclipse; in the verification of the LCD in the RTL level, it is used the Cynthecizer tool, finally, the Quartus tool and the Cyclone II FPGA is used for the verification of the LCD prototyped.

7.4 Test Cases

In the LCD verification, it was realized compliance tests, corner tests and random tests, verifying if the LCD functionalities are correctly implemented. The test cases are described in Appendix A.

7.5 Scripts

In the actual version of the LCD Controller and Driver verification, there weren't implement scripts.

7.6 Coverage

Below, it explains the coverage criteria provided by verification realized in the LCD Controller and Driver.

7.6.1 Functional Coverage

The functional coverage criteria was tests all LCD's functionalities for complete, observing if their main flows and secondary flows described in the LCD Controller and Driver Use Case Specification are realized correctly.

7.6.2 Code Coverage

It wasn't made the measurement of the code coverage carried through for the tests realized in the LCD Controller and Driver.

8. References

[1] BrazilIP ipPROCESS: http://www.brazilip.org.br/ipprocess

[2] LCD Controller and Driver Requirements Specification;

[3] LCD Controller and Driver Design Document;

[4] LCD Controller and Driver Verification Plan;

[5] LCD Controller and Driver Test Cases Document

Anexo A – Conjunto de Artefatos Propostos pelo ipQUALITY

Artefatos	Especificação ipQUALITY
1. Artefatos Comuns	
I. Código RTL (sintetizável)	☺☺☺☺
II. Scripts de instalação	☺
III. Scripts de síntese	☺☺
2. Documentação Geral	
I. Descrição da funcionalidade	☺☺☺
II. Lista de artefatos	☺☺
III. Benefícios do IP-core	☺☺☺
IV. Compatibilidade com padrões	☺☺☺
2.1. Descrição Lógica	
I. Funcionalidade	☺☺☺
II. Interfaces	☺☺☺
III. Arquitetura	☺☺☺
IV. Diagramas de temporização	☺☺
V. Requisitos de integração	☺☺☺
VI. Suíte de testes do sistema	☺

Artefatos	Especificação ipQUALITY
2.2. Integração	
I. Restrições de Projeto	☺
II. Compatibilidade de Projeto	☺
III. Ambiente de Referência	☺☺
IV. Como configurar o ambiente de referência - IP	☺☺
V. Como configurar o ambiente de referência - verificação	☺☺
VI. Ferramentas	☺☺
VII. Fluxos	☺
VIII. Metodologias	☺☺
2.3. Manual de Referência do Programador	
I. Conjunto de Instruções	☺
II. Informações sobre registradores	☺
III. Instruções de como programar o dispositivo	☺
2.4. Implementação	
I. Modelos de estrutura	☺☺
II. Modelos de estimativa de potência	☺☺
III. Estimativa de área	☺☺
IV. Restrições de projeto (tempo, clock, testes e ambiente)	☺
V. Ambiente de simulação	☺☺☺
VI. Biblioteca de referência da tecnologia	☺☺

Artefatos	Especificação ipQUALITY
3. Verificação	
I. Documentação da verificação	☺☺☺
II. Ambiente de verificação	☺☺☺
III. Biblioteca de referência para verificação	☺☺
IV. Testbench	☺☺☺☺
V. Fluxo de ferramentas (verificação)	☺☺
VI. Scripts de verificação	☺☺☺
4. Distribuição	
I. Versões do IP-core	☺☺
II. *Bugs* do VC	☺☺

Anexo B – Documentação do Estudo de Caso

ipPROCESS

Vision Document

LCD Controller and driver

Project Fênix

BrazilIP

Version : 1.7

Revision History

Date	Version	Description	Author
01/17/2006	1.0	Description of introduction and scope.	Daniele Santos.
01/20/2006	1.1	Description of product features.	Daniele Santos.
01/28/2006	1.2	Description of stakeholders and user description topic and project purpose topic.	Daniele Santos.
02/01/2006	1.3	Correction of document's template.	Daniele Santos.
02/02/2006	1.4	Changes in the licensing topic and applicable standards.	Daniele Santos.
02/08/2006	1.5	Changes in the table of contents and user topic.	Daniele Santos.
05/16/2005	1.6	Change in the description of the Product Overview topic.	Daniele Santos.
05/18/2006	1.7	Change in the user environment topic.	Daniele Santos.

1. Introduction

The purpose of the Vision Document is to gather and to display all of the information about the stakeholders and users of LCD Controller and driver. Their roles, needs and how the LCD Controller and driver fulfils these needs.

1.1 Scope

This Vision Document presents general information about the project of LCD Controller and Driver, describing its purpose, identifying its stakeholders and users and showing the product through different views, influencing all the project's development.

1.2 Document Structure

This subsection describes what the rest of this document contains and explains how the document is organized.

- Section 2 – Project Purpose: this section describes the project purpose, or either, the product definition and the definition of the problem that the product has to solve.
- Section 3 – Stakeholders and User Description: this section identifies the stakeholders and the users of the product.
- Section 4 – Product Overview: this section gives a product overview.
- Section 5 – Other Product Requirements: this section provides a list of the general product requirements.
- Section 6 – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Project Purpose

The LCDs are used in many electronics project to facilitate the interaction of the user with the project, with the LCD, it can show relevant information for the use of project, as well as returning to the user results gotten in some type of processing; the current have a controller, an integrated circuit that is in the proper display. The purpose of this project is the construction of a LCD Controller and Driver that it can supply this necessity observed in the displays LCD and that beyond of to be responsible for all LCD Display's functionality, also has the Driver function, or either, it is responsible for the characters printing in the display.

3. Stakeholder and User Descriptions

In this section, are identified the stakeholder and user of the LCD Controller and Driver, assigning responsibilities to them.

3.1 Stakeholders

Below, it lists the LCD's Controller and Driver project stakeholders with its descriptions and responsibilities.

Table 9: Project Stakeholders.

Name	Description	Responsibilities
BrazilIp	It is a collaborative effort of Brazilian institutions to create a distributed network of integrated circuit (IC) design centers capable of delivering OCP-IP compliant Intellectual Property (IP cores).	It gives support to the project development.
Developers	People who develops the project.	Developer the project.
Electronic engineers	People who manage the project.	They are responsible for successful planning and execution of the project.

3.2 Users

Below, it lists the LCD's Controller and Driver users with its descriptions and responsibilities.

Table 10: Product Users.

Name	Description	Responsibilities
BrazilIp's projects.	BrazilIp's projects that need a controller display LCD.	Validate the display LCD Controller and Driver testing its functionalities.

3.3 User Environment

The LCD Controller and Driver will be used to validate the IPProcess, a development process for Soft IP-core with prototyping in FPGA.

3.4 Alternatives and Competition

There are in the market various products similars to the LCD Controller and Driver that we plan to construct, the most important are: HD44780U – Dot Matrix Liquid Crystal Display Controller/driver, produced by Hitachi; PCF2116 family – LCD controller/drivers, produced by Philips; KS0066U – 16com/40 seg driver & controller for dot matrix lcd, produced by Samsung; and Dot matrix character LCD module produced by Optrex Corporation.

4. Product Overview

Below, it gives a product overview.

4.1 Product Capabilities

The Controller and Driver LCD controls all the functionality of a alphanumeric display LCD. It prints 32 characters in two lines, 16 in each, and possess several functions to manipulate the printed character.

4.2 Product Features

The LCD Controller and Driver's main features are:

- 5 x 8 and 5 x 10 dot matrix possible
- 2-line display of up characters per line
- Character generator ROM: 240 characters
- Wide range of instructions functions:

- Clear Display, Return Home, Entry Mode Set, Cursor or Display Shift, Function Set, Writing Characters, Display On/Off Control.

- Compatible with the industry standard Hitachi HD44780U LCD controller.

4.3 Licensing

The HD44780U is the LCD controller standard used by manufacturers, this standard is open source.

4.4 Deliverables and Installation

The final product will consist beyond of LCD Controller and Driver, a user guide, which have necessary product information to the user.

4.5 Assumption, Dependencies and Constraints

The Controller and Drive LCD prints in the maximum 32 characters.

5. Other Product Requirements

Below its provides a list of the general product requirements.

5.1 Applicable Standards

The standards that will be adopted on the product development are VSIA standard, the industry standard Hitachi HD44780 LCD controller; in relation to the characters printed by the display, the standard adopted will be the ASCII standard.

5.2 Hardware and Environmental Requirements

The hardware and environmental resources needed to develop the product are:

- An alphanumeric LCD Display, with two lines of 16 characters;
- 2 microcomputers Pentium IV 1.6 GHz or equivalent, with 256 MB of RAM, HD 20 GB.

5.3 Performance Requirements

The characters must be printed in the display at least sixty times per second.

6. References

[1] Datasheet Hitachi http://www.sparkfun.com/datasheets/LCD/HD44780.pdf ;

[2] Datasheet KS0066U http://pdf.alldatasheet.com/datasheet-pdf/view/37317/SAMSUNG/KS0066U.html ;

[3] Datasheet Optrex Corporation http://www.optrex.com/pdfs/Dmcman_1-10.pdf;

[4] Datasheet PCF2116 family LCD controller/driver: http://www.semiconductors.philips.com/acrobat_download/datasheets/PCF2116_FAM_4.pdf .

ipPROCESS

Requirements Specification

Lcd´s Controller and Driver

Project Fenix

BrazilIP

Version : 1.9

Revision History

Date	Version	Description	Author
21/05/2005	1.0	Definition of first version lcd´s controller.	Daniele Santos, Rodrigo Peixoto.
17/08/2005	1.1	Revision of lcd´s controller requirements.	Daniele Santos, Rodrigo Peixoto.
06/09/2005	1.2	Revision of lcd´s controller requirements.	Daniele Santos, Rodrigo Peixoto.
04/10/2005	1.3	Revision of lcd´s controller requirements.	Daniele Santos, Rodrigo Peixoto.
31/10/2005	1.4	Changes in the distribution of requirements in groups.	Daniele Santos.
01/11/2005	1.5	Includes the family of the LCD Control and Driver and it cuts the MPU interface of the not functional requirements and reedits some places in the document.	Daniele Santos, Rodrigo Peixoto.
01/18/2006	1.6	Changes in the document formatting.	Daniele Santos.
04/10/2006	1.7	Change in the Non Functional Requirement 3.	Daniele Santos.
05/16/2006	1.8	Changes in the description of the following functional requirements: Cursor or Display shift and Function Set.	Daniele Santos.
05/24/2006	1.9	Changes in the ip_functional_requirement and ip_non_functional_requirement styles.	Daniele Santos.

1. Introduction

This document has as objective to explain the functionality and peculiarities of a controller of alphanumeric display LCD, with the purpose to refine and to facilitate the development. The used LCD contains 32 (two lines of 16) characters of 5x8 or 5x7 bits (the last line of pixels of the character can be for the cursor). The spaces which do not contains pixels separate to the characters and the lines of the display.

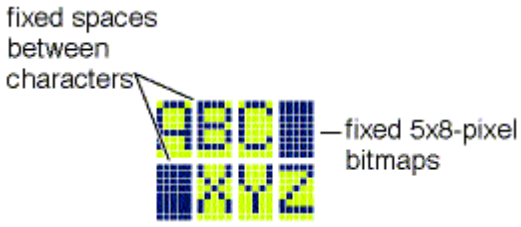

Figure 1: The mapping pixels.

1.1 Document Overview

Section 2 – Functional Requirements: this section lists all functional requirements of the project

Section 3 – Non Functional Requirements: this section lists all non functional requirements of the project

Section 4 – References: this section provides a complete list of all documents referenced elsewhere in this document.

1.2 Definitions, Acronyms and Abbreviations

This table describes some terms used in this document that appear some times in this text.

Table 11: Definitions, Acronyms and Abbreviations.

Term	Description
Functional requirements	Hardware's requirements that composes the controller, that describes action that the controller must be apt to execute.
Not functional requirements	Hardware's requirements that composes the controller, that describes attributes that controller must possess or restrictions under which it must operate.

1.3 Requirements Priority

To help managing the project scope and development priorities, the requirements presented in this document are categorized by their priorities that are:

- **Essential** – this type of requirement must be implemented to the system working.
- **Important** – without this requirement the system can work, but not in the expected way.
- **Desirable** – this type of requirement doesn't compromise the system working.

2. Functional Requirements

[FR 01] Clear Display

Returns the display to its original status if it was shifted. In other words, the display disappears and the cursor or blinking goes to the left edge of the display (in the first line).

Priority: Essential

[FR 02] Return Home

Returns the cursor to its original position if it was shifted. The cursor goes to the left edge of the display (in the first line).

Priority: Essential

[FR 03] Entry Mode Set

Sets cursor move direction and specifies display shift. These operations are performed during data write and read.

Priority: Important

[FR 04] Cursor or Display Shift

Cursor or display shift shifts the cursor position or display to the right or left without writing or reading display data. In a 2-line display, the cursor moves to the second line when it passes the 16th digit of the first line. The first and second line displays will shift at the same time. When the displayed data is shifted repeatedly each line moves horizontally. The second line display does no shift into the first line position.

Priority: Essential

[FR 05] Function Set

Sets the number of display lines, 1 or 2 lines, the character font, 5 x 8 or 5 x 10 matrixes of dots possible.

Priority: Important

[FR 06] Writing characters

Characters are printed in the display thus that them will be sent for controller. The characters are printed matters from the first line (case the display posses more than one line).

Priority: Essential

[FR 07] Display ON/OFF control

Sets entire display on/off, cursor on/off and blinking of cursor position character.

Priority: Essential

3. Non Functional Requirements

[NRF 01] Characters printing speed

The characters must be printed in the display at least sixty times per second.

[NRF 02] OCP-IP interface

The module have to respect the OPC-IP interface with the others modules.

[NRF 03] The standard

The LCD Controller and Driver obeys the industry standard Hitachi HD44780 LCD controller.

4. References

[4] Datasheet Hitachi http://www.sparkfun.com/datasheets/LCD/HD44780.pdf

[5] Datasheet KS0066U http://pdf.alldatasheet.com/datasheet-pdf/view/37317/SAMSUNG/KS0066U.html

[6] Datasheet Optrex Corporation http://www.optrex.com/pdfs/Dmcman_1-10.pdf

[7] Datasheet PCF2116 family LCD controller/driver: http://www.semiconductors.philips.com/acrobat_download/datasheets/PCF2116_FAM_4.pdf

ipPROCESS

Design Document

LCD Controller and Driver

Project Fenix

BrazilIP

Version : 4.6

Revision History

Date	Version	Description	Author
11/03/2005	1.0	Begins of the document, definition of the sections and add the class diagram figure.	Daniele Santos, Rodrigo Peixoto.
11/07/2005	1.1	Description of capsules	Daniele Santos, Rodrigo Peixoto.
11/09/2005	1.2	Changes in the pictures.	Daniele Santos.
11/09/2005	1.3	Changes in the Description of protocols.	Daniele Santos.
11/10/2005	1.4	Description of the state diagrams.	Daniele Santos, Rodrigo Peixoto.
11/11/2005	1.5	Changes in the Description of state diagrams.	Daniele Santos.
11/12/2005	1.6	Changes in the pictures and description of protocols.	Daniele Santos.
11/13/2005	1.7	Revision of capsule's descriptions and it inserts the structure's diagram pictures.	Daniele Santos.
11/14/2005	1.8	Changes some topics in this document.	Rodrigo Peixoto.
11/16/2005	1.9	Changes some topics in this document.	Rodrigo Peixoto.
11/17/2005	2.0	Changes in the Description of state diagrams.	Daniele Santos, Rodrigo Peixoto.
11/18/2005	2.1	Revision of capsule's descriptions.	Daniele Santos, Rodrigo Peixoto.
03/21/2006	2.2	Initial description of LCD_Controller_and_Driver capsule	Daniele Santos.
03/24/2006	2.3	Description of UserInterface subsystem.	Daniele Santos.
03/27/2006	2.4	Description of Character subsystem	Daniele Santos.
03/28/2006	2.5	Description of capsule's state diagram.	Daniele Santos.
03/30/2006	2.6	Description of some protocols.	Daniele Santos.
03/31/2006	2.7	Changes in the pictures and description of protocols.	Daniele Santos.
04/06/2006	2.8	Insertion of the new structure diagrams.	Daniele Santos.
04/07/2006	2.9	Description of state diagram of CommandInterface and DecoderCommand capsules.	Daniele Santos.
05/09/2006	3.0	Description of the structure diagram.	Daniele Santos.
05/12/2006	3.1	Initial description of the DisplayCommunication subsystem.	Daniele Santos.
05/16/2006	3.2	Description of the DisplayControl capsule.	Daniele Santos.
05/19/2006	3.3	Changes in the DisplaySubsystem structure.	Daniele Santos.
05/22/2006	3.4	Changes in the descriptions of the Display_char and Display_Cursor.	Daniele Santos.
05/30/2006	3.5	Change in the Cursor's state diagram.	Daniele Santos.
06/16/2006	3.6	Description of the modeSet, blink, clock. cursorOnOff and numberLines protocols.	Daniele Santos.
06/17/2006	3.7	Insertion of the reference to the modeSet, blink, clock, cursorOnOff and numberLines protocols.	Daniele Santos.
06/19/2006	3.8	Changes in the readMaps and Pdisplay_char protocols.	Daniele Santos.

06/30/2006	3.9	Description of the Appendix A.	Daniele Santos.
07/03/2006	4.0	Description of the protocol's state diagram of the Appendix A.	Daniele Santos.
07/10/2006	4.1	Description of the OCP_Slave's state diagram.	Daniele Santos.
07/20/2006	4.2	Change in the descriptions of the capsules and protocols of Controller and Character subsystems, exclusion of the Driver subsystem.	Daniele Santos.
07/29/2006	4.3	Change in the descriptions of the capsules and protocols of Cursor and Driver subsystems.	Daniele Santos.
07/30/2006	4.4	Change in the description of the Driver capsule.	Daniele Santos.
07/31/2006	4.5	Change in figure of the LCD's structure diagram.	Daniele Santos.
08/01/2006	4.6	Change in the description and the figure of the DataDisplayArray capsule.	Daniele Santos.

1. Introduction

This document provides the design structure references to the LCD Controller and Driver.

1.1 Document Overview

The following sections compose this document:

- Section 1 – LCD Controller and Driver: this section shows the LCD Controller and Driver, it is seen behind its internal packages, its capsules diagram and structure diagram;
- Section 2 – Controller subsystem: this section shows the Controller subsystem with its structure diagrams, state diagrams, capsules and protocols;
- Section 3 – Character subsystem: this section shows the Character subsystem with its structure diagrams, state diagrams, capsules and protocols;
- Section 4 – Cursor subsystem: this section shows the Cursor subsystem with its structure diagrams, state diagrams, capsules and protocols;
- Section 5 – Driver subsystem: this section shows the Driver subsystem with its structure diagrams, state diagrams, capsules and protocols;
- Section 7 – References: this section provides a complete list of all documents referenced elsewhere in this document;
- Appendix A: this section shows the necessary structure for the communication with the LCD Controller and Driver through the OCP-IP interface.

1.2 Definitions, Acronyms and Abbreviations

This table is used for specify any terms used some times in this text.

Table 12: Definitions, Acronyms and Abbreviations.

Term	Description
CAP	Capsule
PROT	Protocol
SD	State diagram
STD	Structure diagram

2. LCD Controller and Driver

The LCD Controller and Driver is divided in subsystems, with its capsules and protocols representing the necessary structure for the implementation of the use cases described in the LCD Controller and Driver Use Case Specification.

2.1 Internal Package Diagram

The LCD Controller and Driver is composed by four subsystems: Controller, Character, Cursor and Driver. The figure below shows the subsystems and their dependences.

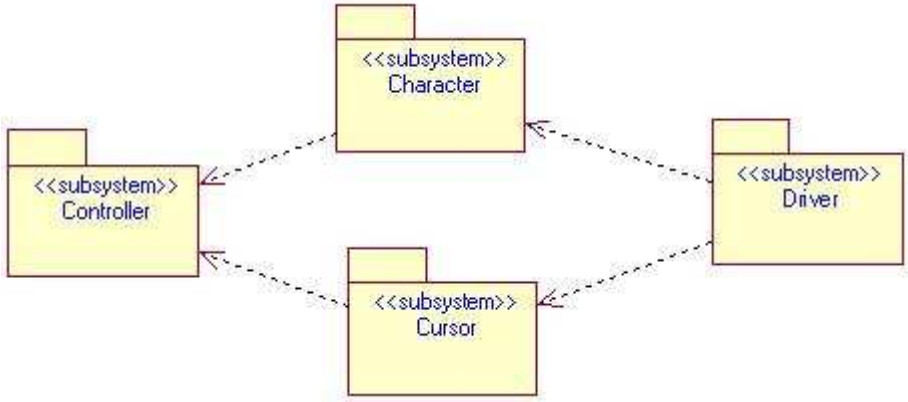

Figure 1: Internal Package Diagram.

2.2 Structure Diagram

LCD Controller and Driver's structure diagram

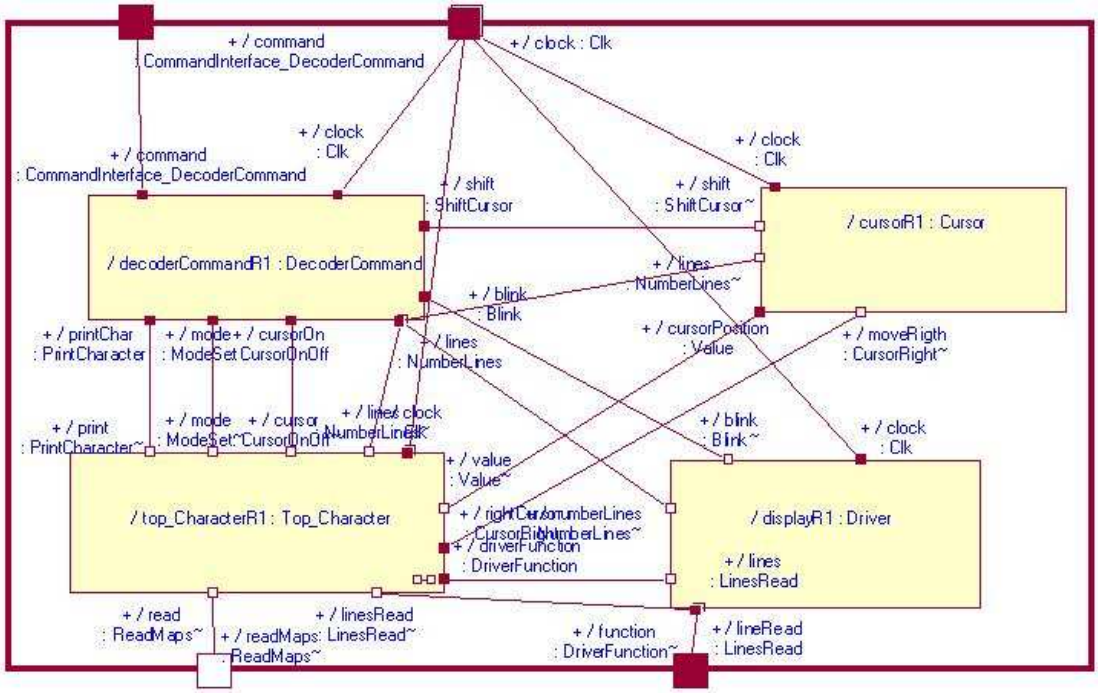

Figure 2: LCD Controller and Driver's structure diagram.

2.3 Capsules diagrams

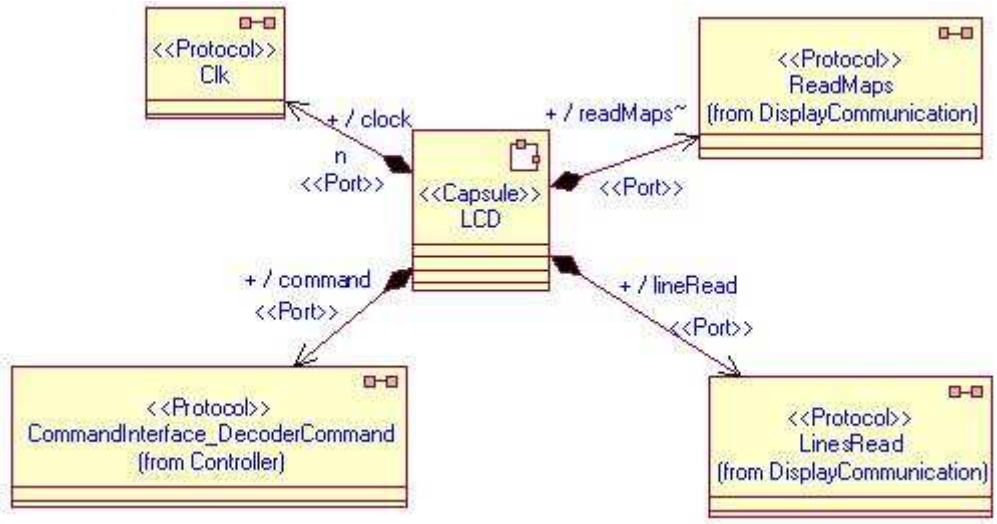

Figure 3: LCD Controller and Driver's capsules diagram.

[CAP 001] LCD

This capsule represents the LCD Controller and Driver.

Figure 4: LCD capsule.

2.4 Protocols

- **CommandInterface_DecoderCommand:** This protocol will be explained later in this document, in [PROT 002];
- **LinesRead:** This protocol will be explained later in this document, in [PROT 0015];
- **ReadMaps:** This protocol will be explained later in this document, in [PROT 0012].

[PROT 001] Clk

This protocol represents the interaction between the LCD Controller and Driver and the user in relation to the sending of the clock and reset signals.

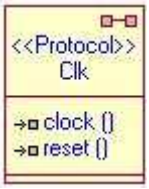

Figure 5: Clk protocol.

In signals:

- **clock:** The system clock.
- **reset:** The system reset signal.

3. Controller subsystem

This subsystem represents the necessary control of the LCD Controller and Driver to decoder the command.

3.1 Capsules diagrams

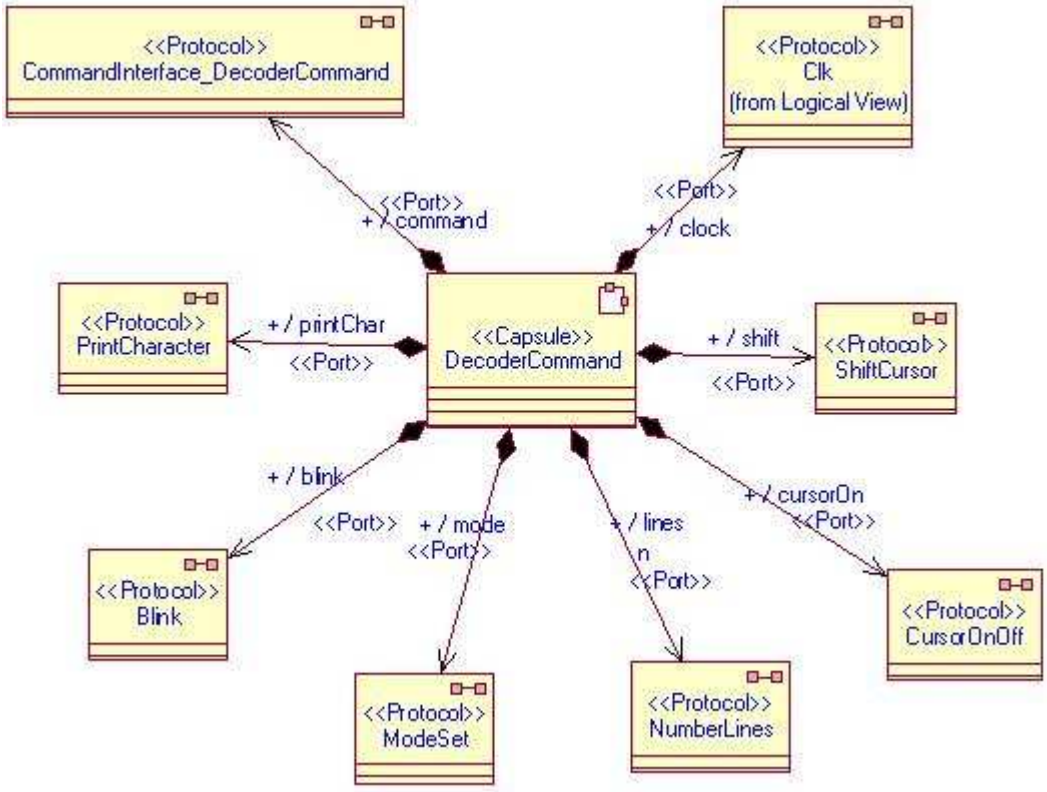

Figure 6: Controller subsystem's capsules diagram.

[CAP 002] DecoderCommand

This capsule is responsible for the decoding of the user's command.

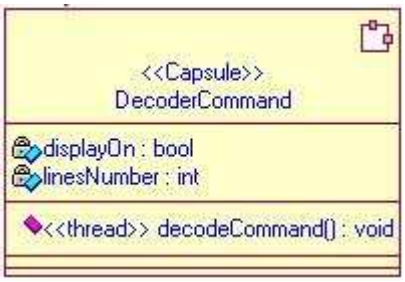

Figure 7: DecoderCommand capsule.

Attributes:

- **displayOn:** It indicates if the LCD Controller and Driver is enabled to function.
- **linesNumber:** It indicates the amount of the character's lines that can be printed on the display.

Operations:

- **decodeCommand:** It decodes the user's command**.**

[SD 001] DecoderCommand's state diagram

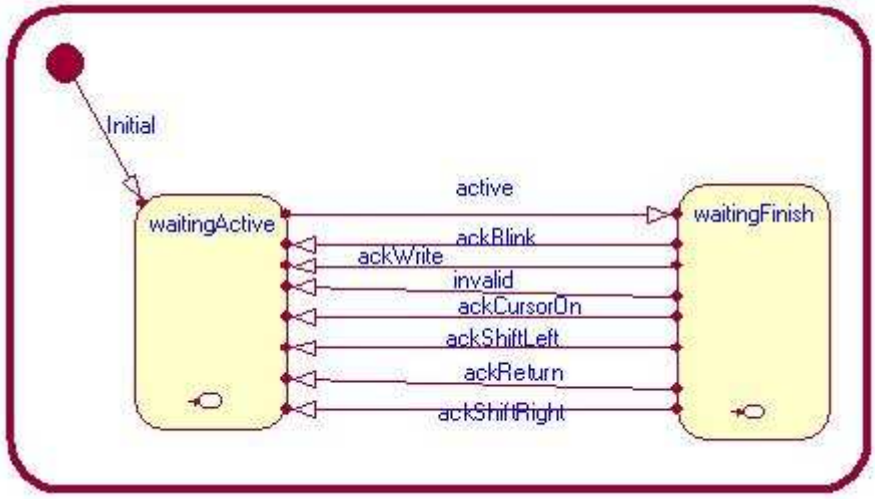

Figure 8: DecoderCommand's state diagram.

The initial state of this capsule is the waitingActive state, it stays in this state until it to receive the active signal; the active signal indicates that command sent by the user can be read.

After the receiving of the active signal, the DecoderCommand capsule goes to waitingFinish state. In the waitingFinish state, the command sent by the user is read and it decoded, the valid commands are the following: **Clear Display** - represented by the code 0x01, **Return Home** - represented by the code 0x02, **Cursor Shift Right -** represented by the code 0x09, **Cursor Shift Left -** represented by the code 0x0D, **Display OFF -** represented by the code 0x08; **Display ON/Cursor OFF** – represented by the code 0x0c, **Display ON/Cursor ON** – represented by the code 0x0e, **Occidental Mode** – represented by the code 0x06, **Oriental Mode,** represented by the code 0x04, **Blink Command,** represented by the code 0x0f and the **Print Char On Display,** represented by the codes that represent a character, the codes that represent a valid character are the codes between 0x10 and 0xff (see the Special Requirements topic of the LCD Controller and Driver Use Case Specification).

After that command is decoded, it is executed and the DecoderCommand capsule continues in the waitingFinish state, this capsule stays in this state waiting the confirmation of the command execution, this confirmation is done through the sending of one the following signals: ackBlink, ackWrite, invalid, ackCursorOn, ackShiftLeft, ackReturn and ackShiftRigth, each one confirming the finish of determined activities.

With the receiving of the confirmation of the command execution, the DecoderCommand sends the ackActive signal for the user, indicating that last command read by the LCD Controller and Driver was executed, after the sending of this signal, the DecoderCommand capsule comes back to initial state.

3.2 Protocols

- **Clk:** This protocol already was explained later in this document, in the [PROT 001].

[PROT 002] CommandInterface_DecoderCommand

This protocol represents the interaction between the user and the DecoderCommand capsule. This interaction is done through the sending of the command to the LCD Controller and Driver and the confirmation of the execution of the last command read by the LCD Controller and Driver.

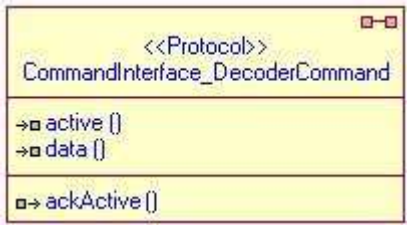

Figure 9: CommandInterface_DecoderCommand protocol.

In signals:

- **active:** It indicates that command sent by the user can be read.
- **data:** Command sent for the LCD Controller and Driver by the user, its type is sc_uint<8>.

Out signals:

- **ackActive:** It indicates for the user the ending of the execution of the last command read by the LCD Controller and Driver.

[SD 002] CommandInterface_DecoderCommand's state diagram

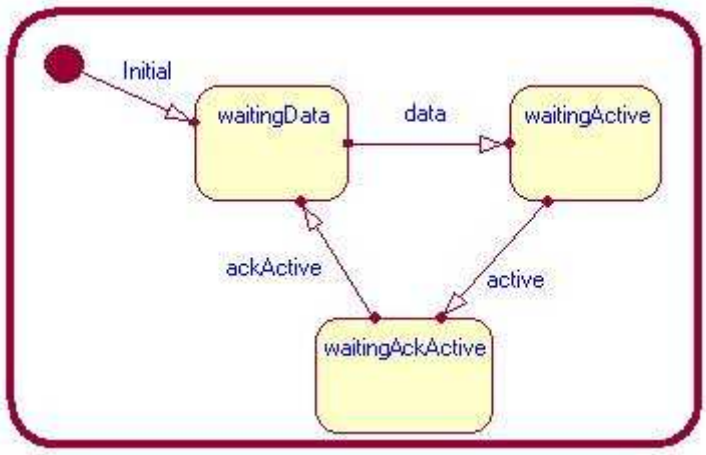

Figure 3: CommandInterface_DecoderCommand state diagram.

The initial state is the waitingData state, the CommandInterface_DecoderCommand protocol stays in this state until the data signal is received; after data signal is received, the protocol goes to waitingActive state; the protocol stays in the waitingActive state until the active signal is received; the next state is waitingAckActive; the protocol stays in waitingAckActive until the ackActive signal is sent; after the sending of ackActive signal, the protocol comes back to initial state.

[PROT 003] PrintCharacter

This protocol represents the interaction between the DecodeCommand capsule and the PrintChar capsule. The purpose of this interaction is the storage of the character's mapping relative to the codes (that it represents a character) sent by the user in the mapCharacters memory from the DataDisplayArray capsule.

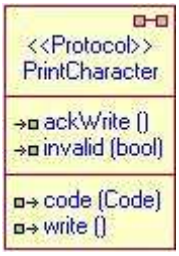

Figure 11: PrintCharacter protocol.

In signals:

- **ackWrite:** It indicates that the mapping of the character reference to the code received from the DecoderCommand capsule was stored in the mapCharacters memory from the DataDisplayArray capsule, the DataDisplayArray capsule will be explained later in this document.
- **invalid:** It indicates that code sent by code signal (explained below) doesn't represent a character (see the Special Requirements topic of the LCD Controller and Driver Use Case Specification).

Out signals:

- **write:** It sends the write's command to printChar capsule; this command indicates that the mapping of the character reference to the code sent in the signal code,

which is explained below, must be stored in the mapCharacters memory from the DataDisplayArray capsule in the cursor position.

- **code:** The character's code that will be printed by the display.

[SD 003] PrintCharacter's state diagram

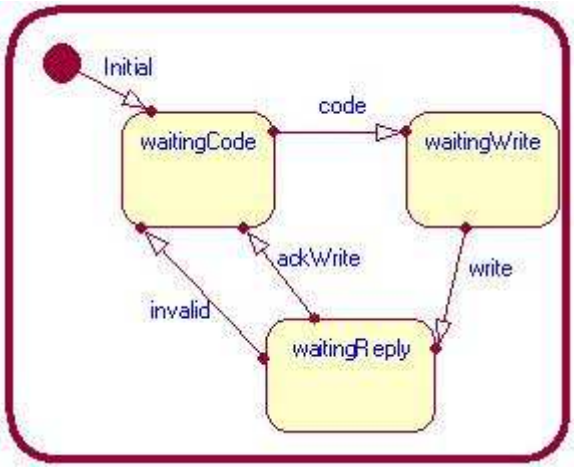

Figure 4: PrintCharacter's state diagram.

The initial state of the PrintCharacter protocol is the waitingCode state; in this state, the protocol waits the sending of the code signal, this signal contains the code reference to the character that will be printed on the display; after the code signal is sent, the protocol goes to waitingWrite state, where the protocol waits the write signal, indicating that mapping of the character reference to the code sent by the code signal must be stored in the mapCharacters memory from the DataDisplayArray capsule in the cursor position. After the receiving of the write signal, the protocol goes to waitingAckWrite, it stays in this state until to receive the ackWrite signal or the invalid signal from the PrintChar capsule, the invalid signal indicates that the code sent by the code signal doen't represent a character, the ackWrite signal indicates that the mapping of the character reference to the code received in the code signal was stored in the mapCharacters memory from the DataDisplayArray capsule, after the sending of the ackWrite signal or invalid signal, the protocol comes back to initial state.

[PROT 004] ShiftCursor

This protocol represents the interaction between the DecoderCommand capsule and the Cursor capsule in relation to the sending of the commands for to shift the cursor.

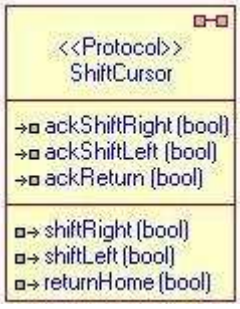

Figure 13: ShiftCursor protocol.

In signals:

- **ackShiftRight:** It confirms the displacement of the cursor for the right direction.
- **ackShiftLeft:** It confirms the displacement of the cursor for the left direction.
- **ackReturn:** It confirms the returnHome command execution.

Out signals:

- **shiftRight:** It sends the command to the cursor to shift for the right direction.
- **shiftLeft:** It sends the command to the cursor to shift for the left direction.
- **returnHome:** It sends the returnHome command to the Cursor capsule; if its value will be true, the command is sent.

[SD 004] ShiftCursor's state diagram

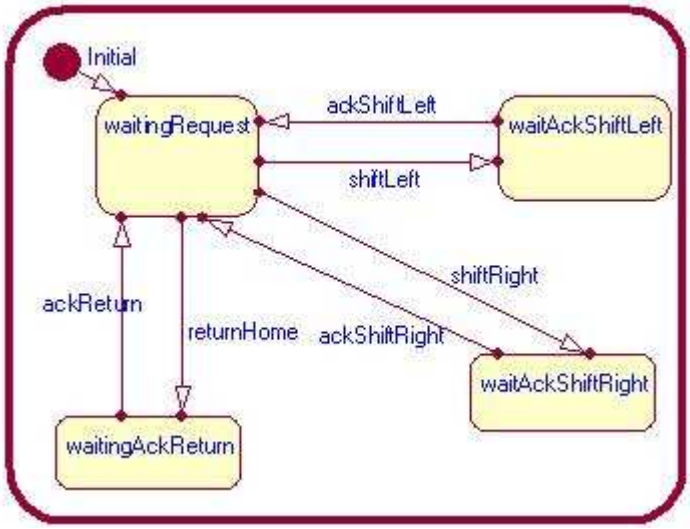

Figure 14: ShiftCursor's state diagram.

The initial state of the ShiftCursor protocol is the waitRequest state, the protocol stays in this state until one of the following signals will be sent: shiftLeft, shiftRight or returnHome signal. If the protocol is in the initial state and the shiftLeft signal will be sent, the protocol goes to waitAckShiftLeft state, the protocol stays in this state until the ackShiftLeft signal will be received, after the ackShiftLeft signal is received, the protocol comes back to initial state; if the ShiftCursor protocol is in the initial state and the shiftRight signal is sent, the protocol goes to waitAckShiftRight state, the protocol stays in this state until the AckShiftRight signal will be received, after the ackShiftRight signal is received, the protocol comes back to initial state; if the ShiftCursor protocol is in the initial state and the returnHome signal is sent, the protocol goes to waitingAckReturn state, the protocol stays in this state until the ackReturn signal will be received, after the ackReturn signal is received, the protocol comes back to initial state.

[PROT 005] Blink

This protocol represents the interaction between the DecoderCommand capsule and the DataDisplayArray capsule in relation of the blink of the cursor position character.

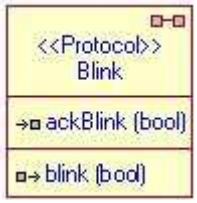

Figure 15: Blink protocol.

In signals:

- **ackBlink:** It indicates the receiving of the blink signal.

Out signals:

- **blink:** It indicates the request of the cursor position character's blink.

[SD 005] **Blink's state diagram**

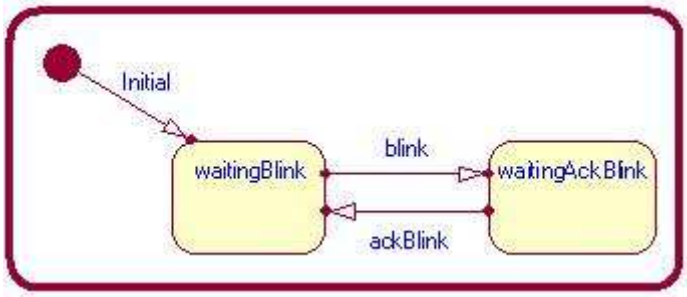

Figure 16: Blink's state diagram.

The initial state of the Blink protocol is the waitingBlink state, the Blink protocol stays in this state until the blink signal will be sent, when the blink signal is sent, indicating the request of cursor position character's blink, the protocol goes to waitingAckBlink state; the protocol waits in the waitingAckBlink state until the ackBlink signal is sent, after the ackBlink signal is sent the protocol comes back to initial state.

[PROT 006] **ModeSet**

This protocol represents the interaction between the DecoderCommand capsule and the PrintChar capsule in relation to the order of memory's position that will be stored the character's mapping.

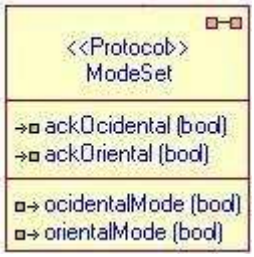

Figure 17: ModeSet protocol.

In signals:

- **ackOcidental:** It confirms the receiving of the ocidentalMode signal.

- **AckOriental:** It confirms the receiving of the orientalMode signal.

Out signals:

- **ocidentalMode:** It indicates that characters will be stored in the memory from the memory first position to last position.
- **orientalMode:** It indicates that characters will be stored in the memory from the memory last position to first position.

[SD 006] ModeSet's state diagram

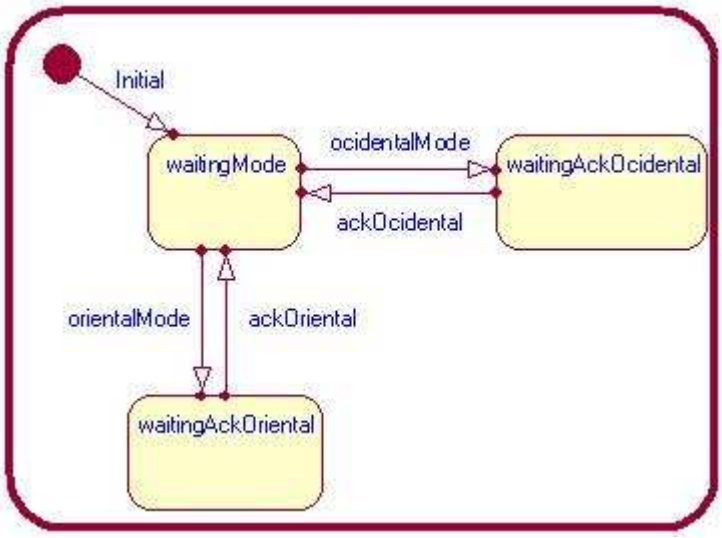

Figure 18: ModeSet's state diagram.

The initial state of the modeSet protocol is the waitingMode state, where the protocols stays until the ocidentalMode signal or orientalMode signal will be sent; if the protocol is in the initial state and the ocidentalMode signal will be sent, the protocol goes to waitingAckOcidental state where it stays until the ackOcidental signal is sent, when the ackOcidental signal is sent, the protocol comes back to initial state. If the protocol is in the initial state and the orientalMode signal is sent, the protocol goes to waitingAckOriental state where it stays until the ackOriental signal is sent, when the ackOriental signal is sent, the protocol comes back to initial state.

[PROT 007] NumberLines

This protocol represents the interaction between the DecoderCommand capsule and the PrintChar capsule, Cursor capsule and Driver capsule in relation of the amount of the character's lines that it can be printed on the display.

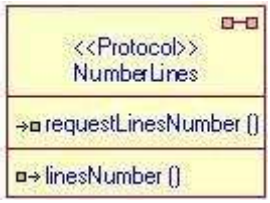

Figure 19: NumberLines protocol.

In signals:

- **requestLinesNumber:** It indicates the request of the amount of the character's lines that it can be printed on the display.

Out signals:

- **linesNumber:** It indicates the amount of the character's lines that it can be printed on the display.

[SD 007] NumberLine's state diagram

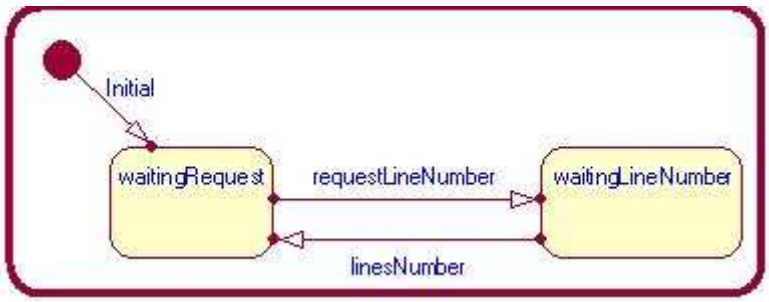

Figure 20: NumberLine's state diagram.

The initial state of the NumberLine protocol is the waitRequest state, it stays in this state until the requestLinesNumber signal is received; when the requestLinesNumber is received, the protocol goes to waitLineNumber state where it stays until the linesNumber signal is sent, after the linesNumber signal is sent, the protocol comes back to initial state.

[PROT 008] CursorOnOff

This protocol represents the interaction between the DecoderCommand capsule and the DataDisplayArray capsule in the relation to the cursor's printing.

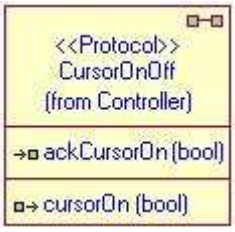

Figure 21: CursorOnOff protocol.

In signals:

- **ackCursorOn:** It indicates the receiving of the cursorOn signal.

Out signals:

- **cursorOn:** It indicates if the cursor will be printed on the display.

[SD 008] CursorOnOff's state diagram

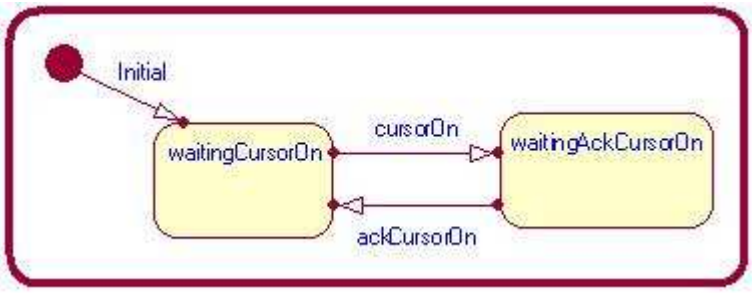

Figure 22: CursorOnOff's state diagram.

The initial state of the CursorOnOff protocol is the waitingCursorOn state, where the protocol stays until the cursorOn signal is sent; after the cursorOn signal is sent, the protocol goes to waitingAckCursorOn state, it stays in this state until the ackCursorOn signal is received, after the ackCursorOn is received, the protocol comes back to initial state.

4. Character subsystem

This subsystem represents the LCD Controller and Driver's functionalities related to the character's manipulation that will be printed on the display.

4.1 Structure diagram

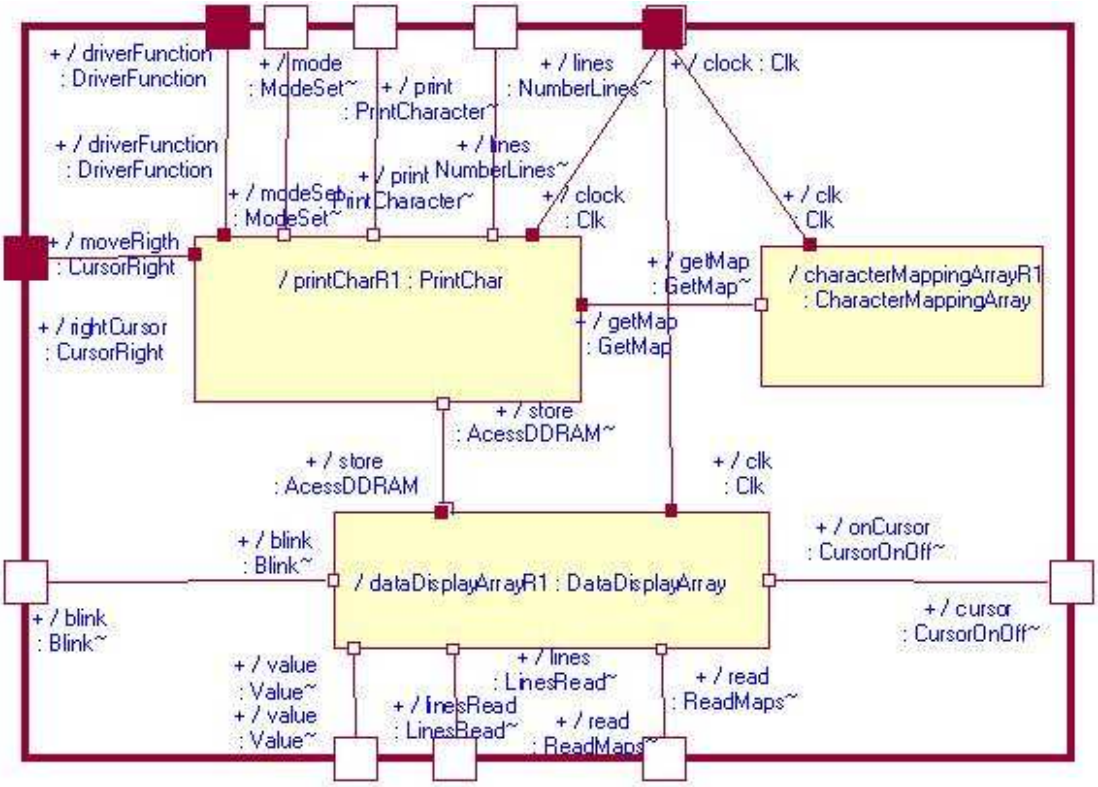

Figure 23: Top_Character structure diagram.

4.2 Capsules diagrams

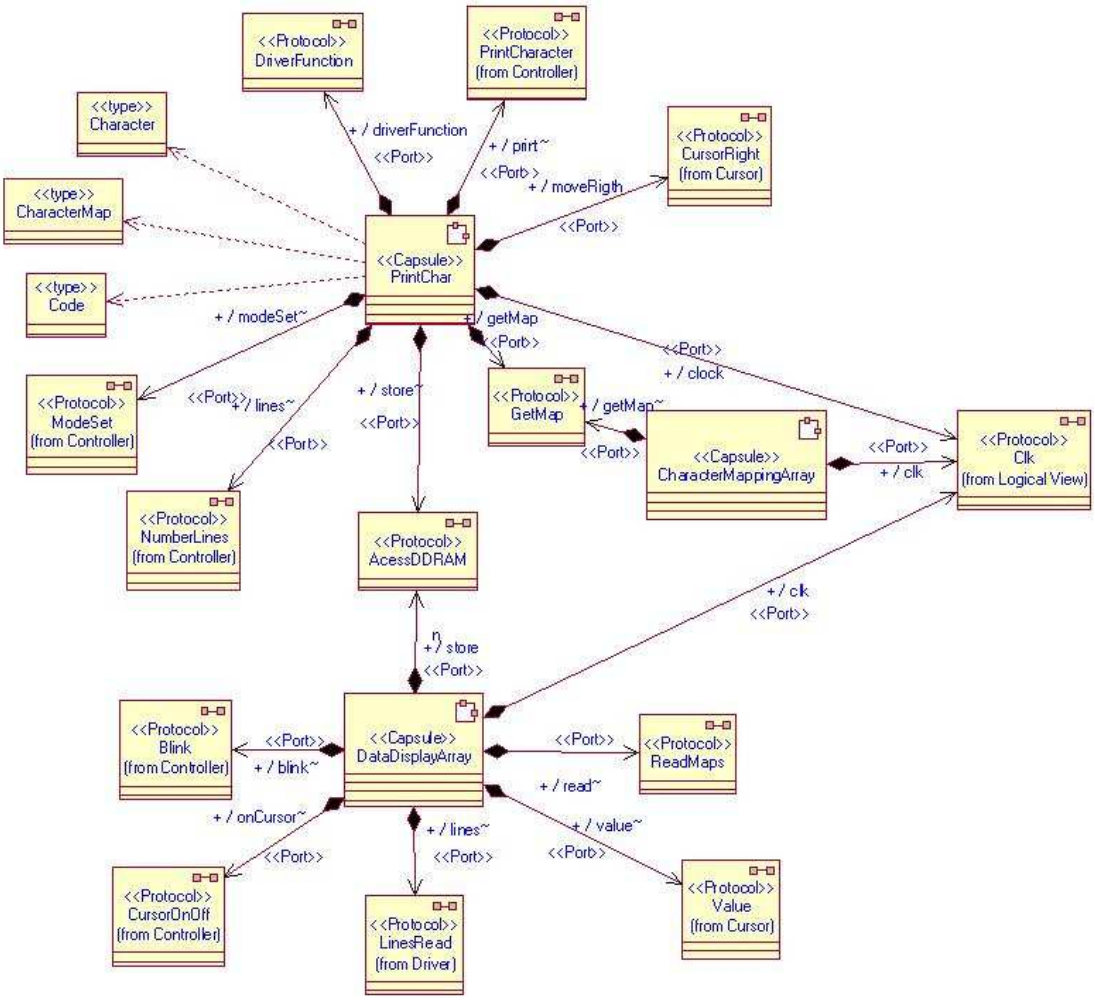

Figure 24: Character's capsule diagram.

[CAP 003] PrintChar

This capsule is responsible by the necessary process for to store in the mapCharacters memory from the DataDisplayArray capsule, that will be explained later, the mapping of the characters that will be printed on the display.

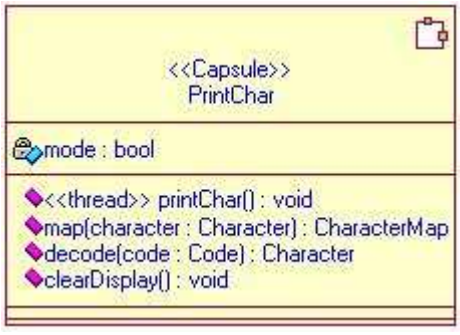

Figure 25: PrintChar capsule.

Attributes:

- **mode:** It indicates the order of memory's position that will be stored the character's mapping; if the mode is true, the character's mapping will be stored in the mapCharacters memory from the first position for last position; if the mode is false, the character's mapping will be stored in the mapCharacters memory from the last position for first position.

Operations:

- **printChar:** It stores in the mapCharacters memory the character's mapping that will be printed on the display.
- **map:** It returns the character's mapping associate to the character received as parameter.
- **decode:** It returns the character associate to the code received as parameter.
- **clearDisplay:** It cleans the display.

[SD 009] PrintChar's state diagram

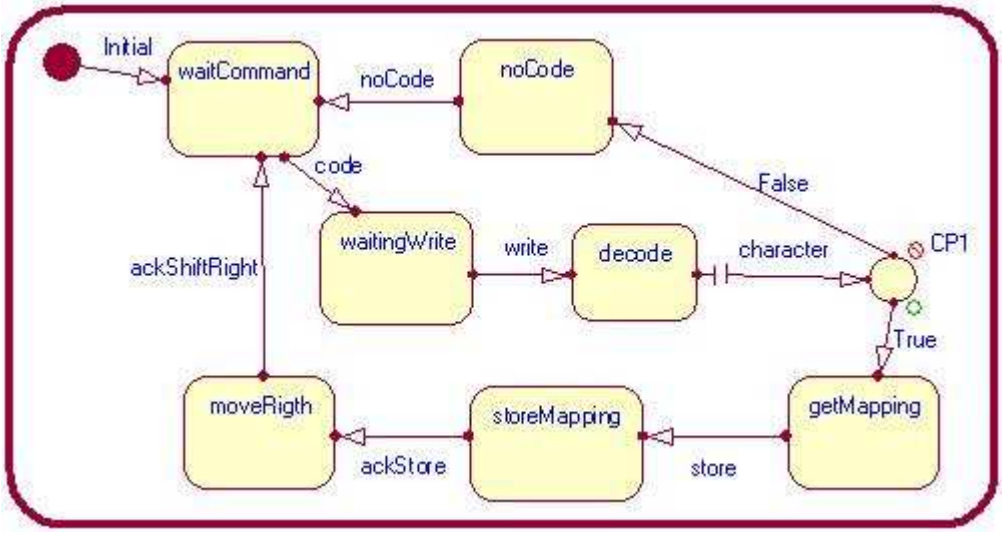

Figure 26: PrintChar's state diagram.

The initial state is the waitCommand state where the PrintChar capsule waits the code signal from the DecoderCommand capsule to initiate its operation; when the PrintChar capsule receives the code signal, it goes to waitingWrite state; in the waitingWrite state, the PrintChar capsule stays waiting to receive the write signal from DecoderCommand capsule; after the receiving of the write signal, the PrintChar capsule reads the code from the code signal and it goes to decode state where the code is decoded; if the code doesn't represent a character, the PrintChar sends the noCode signal for the DecoderCommand capsule and it goes to initial state; if the code represent a character (see the Special Requirements topic of the LCD Controller and Driver Use Case Specification for know which codes represent a character), the capsule goes to getMapping state where the PrintChar capsule obtains the character's mapping from the CharacterMappingArray capsule relative to the code through of the sending the code signal that indicates the code of the character that its mapping is

searched, after the mapping is obtained, this mapping is stored in the mapCharacters from the DataDisplayArray capsule in the cursor position, this is done through the sending of the character signal for the DataDisplayArray capsule, after the mapping is stored, the ackStore signal is received from the DataDisplayArray capsule and the

PrintChar capsule goes to MoveRight state; in the MoveRight state, the PrintChar capsule sends the right signal for the Cursor capsule for the move the cursor for the right direction, after the receiving the ackShiftRight signal from the Cursor capsule, indicating that the cursor was shifted in the right direction, the PrintChar capsule comes back to initial state.

[CAP 004] CharacterMappingArray

This capsule is responsible by the storage of the character's mapping that can be printed on the display by the LCD Controller and Driver, (see the Special Requirements topic of the LCD Controller and Driver Use Case Specification).

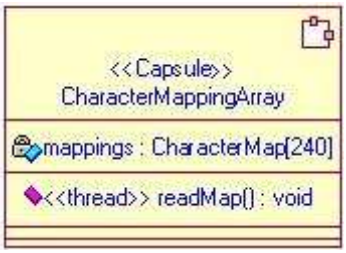

Figure 27: CharacterMappingArray capsule.

Attributes:

- **mappings:** The storage of the character's mappings.

Operations:

- **readMap:** It reads the character's mapping stored in the mappings, explained above, according with character's code.

[SD 010] CharacterMappingArray's state diagram

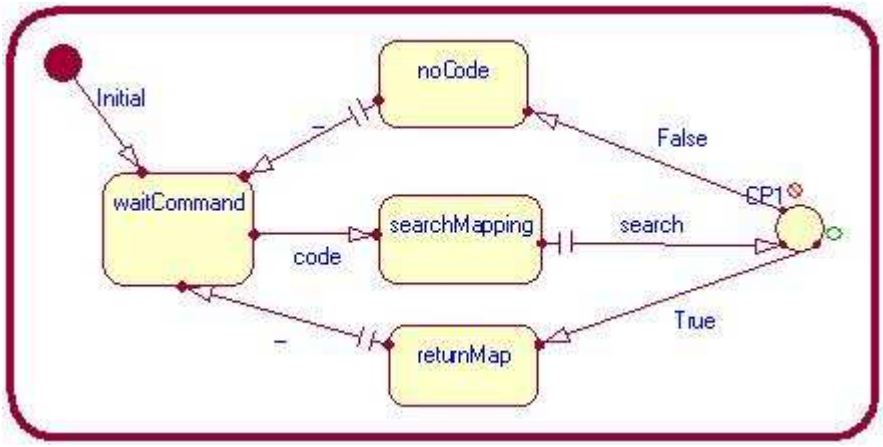

Figure 28: CharacterMappingArray's state diagram.

The initial state of the CharacterMappingArray is the waitCommand state where the CharacterMappingArray capsule waits a request of the PrintChar capsule to initiate its

operation; when the signal code is received, the CharacterMappingArray goes to searchMapping state, where the character's mapping relative to the code is searched; if the character's mapping will not be found, the CharacterMappingArray sends the noCode

signal and it comes back to waitCommand state; if the character mapping will be found, the next state is returnMap state where the character mapping is returned and the capsule comes back to initial state.

[CAP 005] DataDisplayArray

This capsule is responsible by storage of the character mapping that will be printed by the Display.

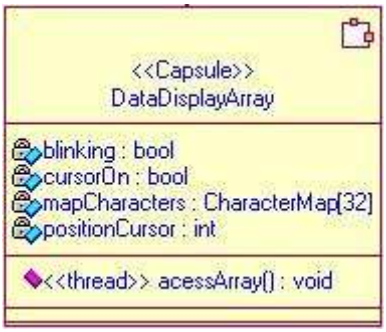

Figure 29: DataDisplayArray capsule.

Attributes:

- **blinking:** It indicates if the cursor position character must blink; if the blinking is true, the cursor position character blinks, contrary case, it doesn't blink.
- **cursorOn:** It indicates if the cursor will be printed on the display; if the cursorOn is true, the cursor will be printed on the display, contrary case, it won't be printed on the display.
- **mapCharacters:** The storage of the character's mappings that will be printed on the display.
- **positionCursor:** It indicates the cursor's position.

Operations:

- **acessArray:** Depending of the signal received, the character signal or activeDisplay signal, the acessArray reads the content of mapCharacters attribute and it writes the character's mapping in the cursor position.

[SD 011] DataDisplayArray's state diagram

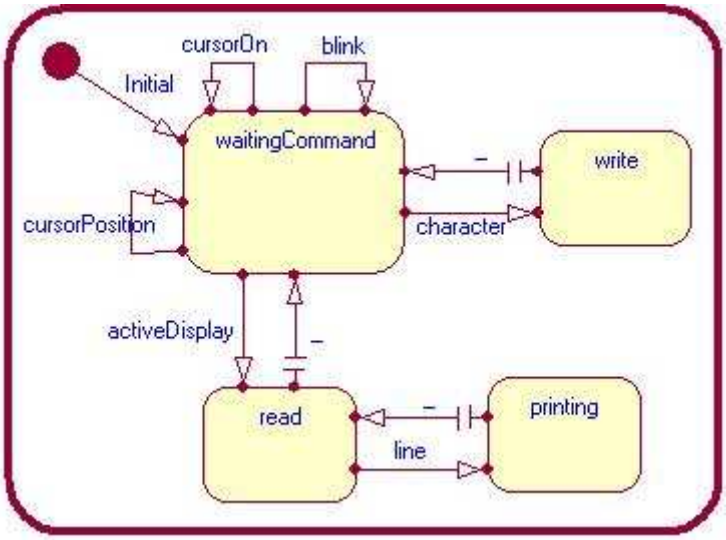

Figure 30: DataDisplayArray's state diagram.

The initial state of the DataDisplayArray capsule is the waitingCommand state; if the DataDisplayArray capsule is in the initial state and it receives the cursorPosition signal, its positionCursor attributes receives the value of the cursorPosition signal, this signal indicates the cursor position, the DataDisplayArray capsule will continue in the initial state; if the DataDisplayArray capsule is in the initial state and it receives the cursorOn signal, its cursorOn attribute receives the value of the cursorOn signal, this value will indicate if the cursor will be printed on the display, the DataDisplayArray capsule will continue in the initial state; if the DataDisplayArray capsule is in the initial state and it receives the blink signal, its blink attribute receives the value of the blink signal, this signal will indicate if the cursor position character must to blink, the DataDisplayArray capsule will continue in the initial state.

If the DataDisplayArray capsule is in the initial state and it receives the character signal, it goes to write state; in this state, the DataDisplayArray stores in the cursor position the character's mapping sent by the character signal; after the mapping is stored, the DataDisplayArray sends the ackStore signal for the PrintChar capsule, confirming the mapping's storage; after the DataDisplayArray capsule comes back to initial state.

If the DataDisplayArray capsule is in the initial state and it receives the activeDisplay signal, it goes to read state; in this state, the DataDisplayArray waits the line signal that it indicates which line of the character's mapping will be printed on the display. After the line signal is received, DataDisplayArray capsule goes to printing state; in this state, the content of the mapCharacters attribute is read and the line of all mapping stored in the mapCharacters reference to the value of the line signal will be printed on the display; after the DataDisplayArray capsule comes back to read state and it stays waiting the line signal for the next line of the mapping is printed; after the all lines were printed, the DataDisplayArray capsule comes back to initial state.

[TYPE 001] Character

This type represents the character.

Figure 31: Character type.

[TYPE 002] CharacterMap

This type represents the mappings reference to the character.

Figure 32: CharacterMap type.

[TYPE 003] Code

This type represents the code associate to the character.

Figure 33: Code type.

4.3 Protocols

- **Clk:** This protocol already was explained in this document, in [PROT 001].
- **Blink:** This protocol represents the conjugated of the Blink [PROT 005] that already it was explained previously in this document.
- **CursorOnOff:** This protocol represents the conjugated of the CursorOnOff protocol [PROT 008] that it already was explained in this document.
- **CursorRight:** This protocol will be explained in this document, in [PROT 0013].
- **Value:** This protocol represents the conjugated of the Value protocol [PROT 0014] that will be explained later in this document.
- **LinesRead:** This protocol will be explained in this document, in [PROT 0015].
- **ModeSet:** This protocol represents the conjugated of the ModeSet protocol [PROT 006] that it already was explained in this document.
- **NumberLines:** This protocol represents the conjugated of the NumberLines protocol [PROT 007] that it was explained in this document.
- **PrintCharacter:** This protocol represents the conjugated of the PrintCharacter protocol [PROT 003] that it was already explained in this document.

[PROT 009] GetMap

This protocol represents the interaction between the PrintChar capsule and the CharacterMappingArray capsule in relation to obtain the character's mapping according the character's code.

Figure 34: GetMap protocol.

In signals:

- **map:** The character's mapping relative to the code (explained later in this document) received.
- **noCode:** It indicates that the code sent by the code signal (explained below) is invalid, i.e., there aren't a character's mapping relative to the code.

Out signals:

- **code:** The character's code that should be printed by the display.

[SD 012] GetMap's state diagram

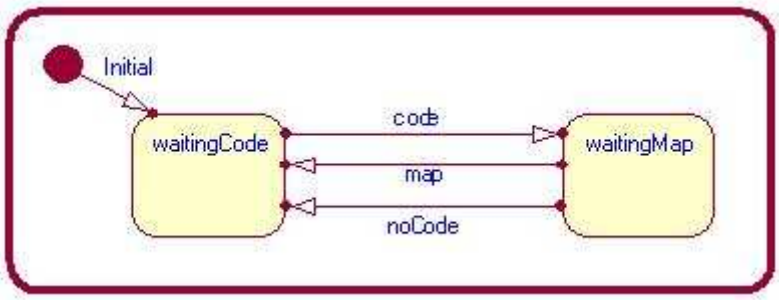

Figure 35: GetMap's state diagram.

The initial state of the GetMap protocol is the waitingCode state, the protocol stays in this state until the code signal will be sent; after code signal is sent, the protocol goes to waitingMap state, where the protocol stays until the map signal or the unCode signal will be received. After that map signal or the unCode signal is received, the protocol comes back to initial state.

[PROT 0010] AcessDDRAM

This protocol represents the access to the DDRAM memory.

Figure 36: AcessDDRAM protocol.

In signals:

- **character:** The character's mapping that will be stored in mapCharacters from DataDisplayArray capsule.

Out signals:

- **ackStore:** It indicates that the character's mapping sent by the character signal was stored in the cursor's position.

[SD 013] AcessDDRAM's state diagram

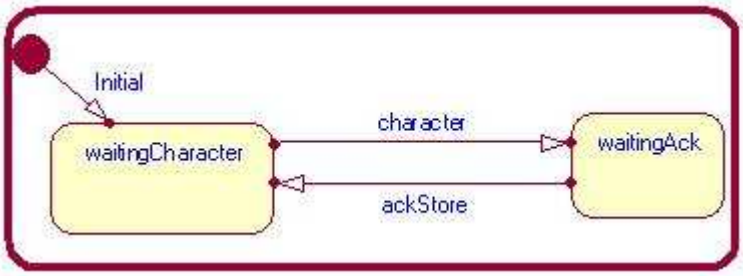

Figure 37: AcessDDRAM's state diagram.

The initial state of the AcessDDRAM protocol is waitingCharacter state, the protocol stays in this state until the character signal to be received; after the character signal is received, the AcessDDRAM goes to waitingAck state, the AcessDDRAM stays in the waitingAck state until the ackStore signal to be sent; after the ackStore signal is sent, the AcessDDRAM comes back to initial state.

[PROT 0011] DriverFunction

This capsule is responsible for sending the command to the Driver capsule for stop or start the reading of the mapCharacters from the DataDisplayArray capsule. The reading of the mapCharacters must stop when a writing will be occurring in the mapCharacters.

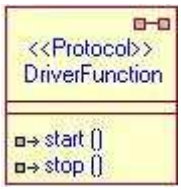

Figure 38: DriverFunction protocol.

Out signals:

- **start:** It indicates for the Driver capsule that the mappingCharacters from the DataDisplayArray capsule can be read.
- **stop:** It Indicates for the Driver capsule that it can't read the mappingCharacters because the mappingCharacters is been write by the PrintChar capsule.

[SD 014] DriverFunction's state diagram

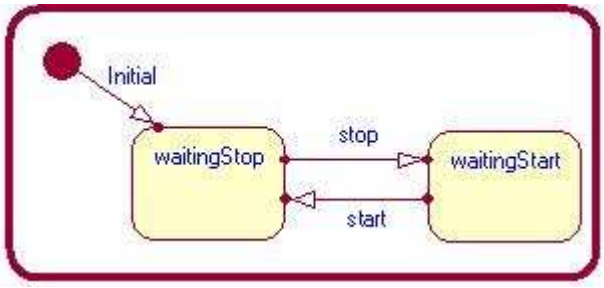

Figure 39: DriverFunction's state diagram.

The initial state is the waitingStop state; the protocol stays in this state until the stop signal will be sent; after the stop signal is sent, the protocol goes to waitingStart state, the protocol stays in the waitingStart state until the start signal will be sent; after that start signal is sent, the protocol comes back to initial state.

[PROT 0012] ReadMaps

This protocol represents the interaction between the DataDisplayArray and the display in relation to the printing of the characters stored in the DataDisplayArray capsule.

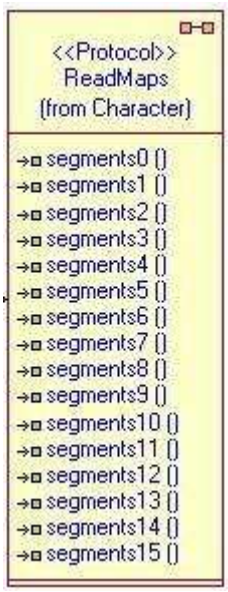

Figure 40: ReadMaps protocol.

Out signals:

- **Segment 0 ...15 (0 to 15):** the signals are responsible to send the characters mapping from DataDisplayArray capsule for the display, these signals are sc_uint<5>.

[SD 015] ReadMap's state diagram

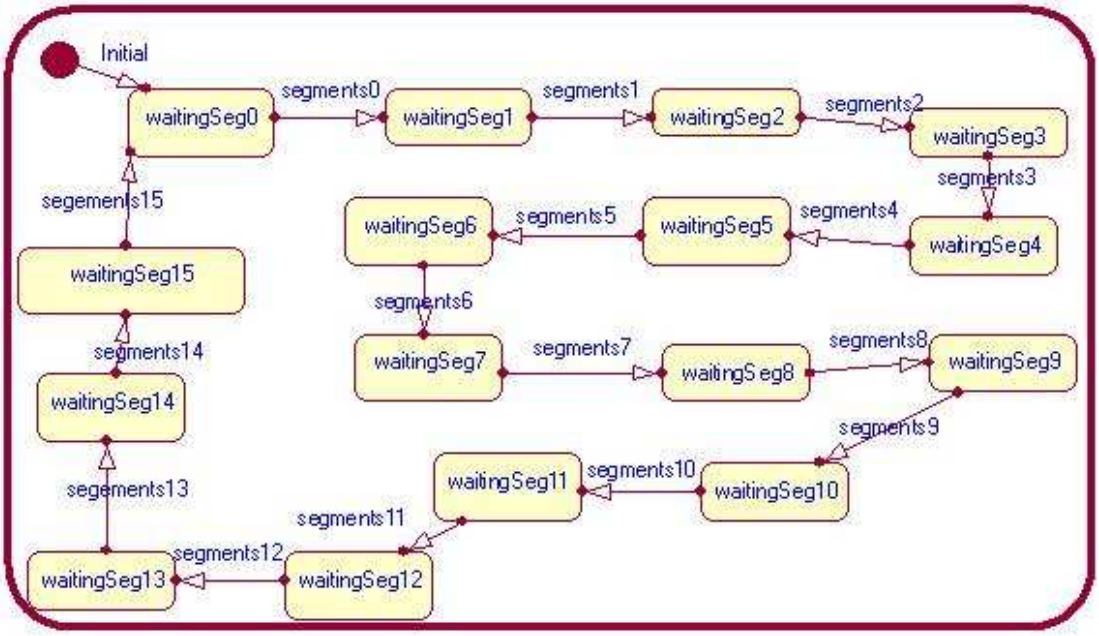

Figure 41: ReadMap's state diagram.

The initial state of the ReadMaps protocol is the waitingSeg0 state; the ReadMaps stays in the initial state until the segments0 signal to be sent; the order that signals are sent is the following: first, the segments0 signal; second, the segments1 signal; third, the segments3 signal and so on until the segments15 is sent; after the segments5 is sent, the ReadMaps protocol comes back to initial state.

5. Cursor subsystem

This subsystem represents the LCD Controller and Driver's functionalities related with cursor's manipulation.

5.1 Capsules diagrams

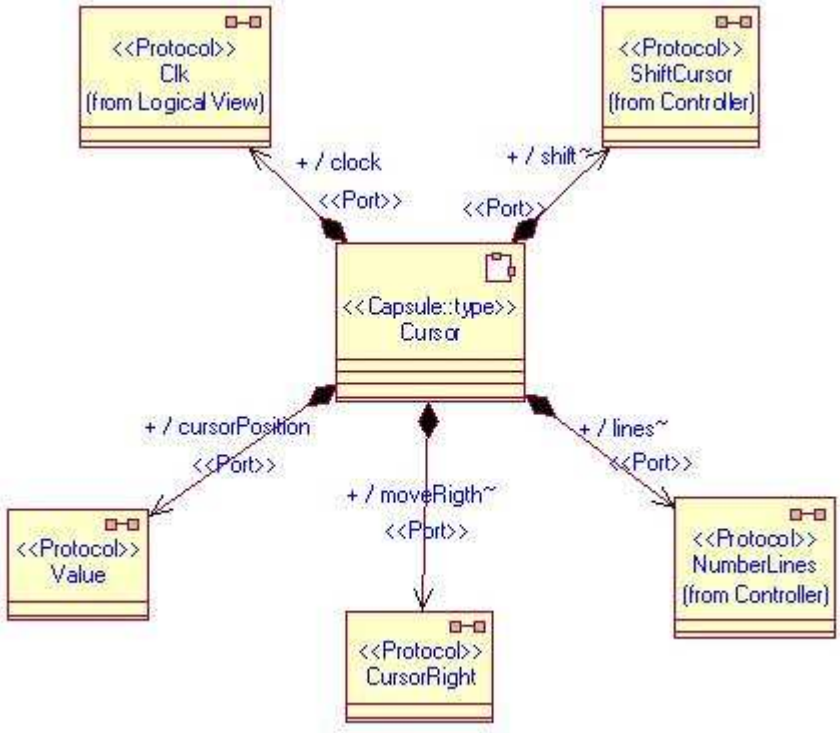

Figure 42: Cursor's capsule diagram.

[CAP 006] Cursor

This capsule is responsible by the activities related to the cursor's position and its displacement.

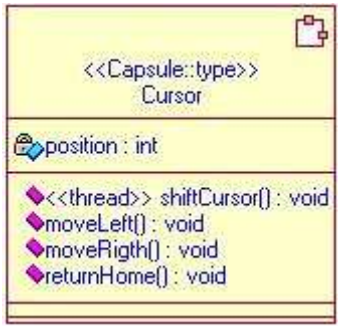

Figure 43: Cursor capsule.

Attributes:

- **position:** It indicates the cursor's position.

Operations:

- **shiftCursor:** It shifts the cursor in accordance with desired direction.
- **moveLeft:** It shifts the cursor to left direction.
- **moveRigth:** It shifts the cursor to right direction.
- **returnHome:** It returns the cursor to the display's 1th digit of the first line.

[SD 016] Cursor's state diagram

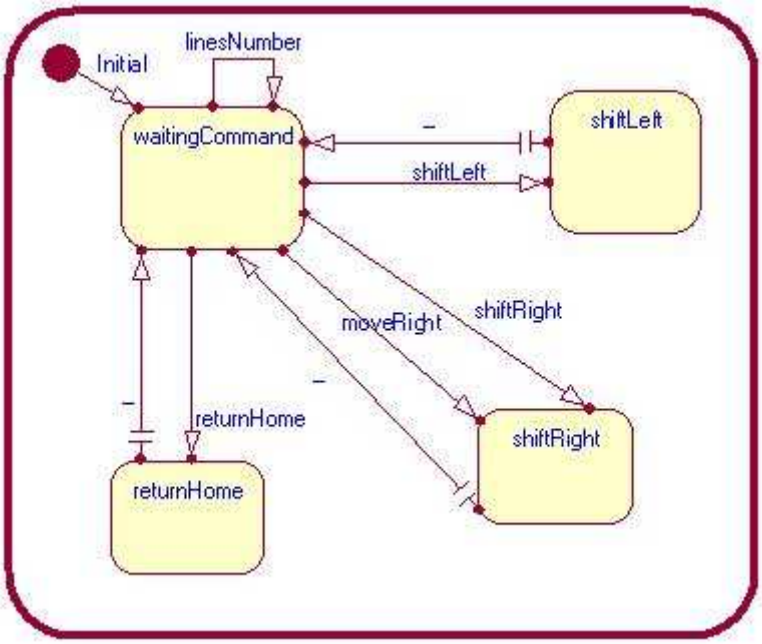

Figure 44: Cursor's state diagram.

The initial state of the Cursor capsule is the waitingCommand state; if the Cursor capsule is in the initial state and it receives the linesNumber signal from the DecoderCommand capsule, it continues in the same state. If the Cursor capsule is in the initial state and it receives the shiftLeft signal from the DecoderCommand capsule, the capsule goes to shiftLeft state where the cursor is shifted for the left direction, after the execution this operation, the Cursor capsule sends the ackShiftLeft signal for the DecoderCommand capsule and it goes to initial state. If the Cursor capsule is in the initial state and it receives the moveRight or shiftRight signals from the PrintChar and DecoderCommand capsule respectively, it goes to shiftRight state where the cursor is shifted for the right direction, after the execution of this operation, the Cursor capsule sends the ackShiftRight signal for the PrintChar capsule or DecoderCommand capsule and it comes back to initial state. If the Cursor capsule is in the initial state and it receives the returnHome signal from the DecoderCommand capsule, the Cursor capsule goes to returnHome state where the cursor is shift for the display's 1th digit of the first line, after the execution this operation, the Cursor capsule sends the ackReturn signal for the DecoderCommand capsule and it comes back to initial state.

5.2 Protocols

- **Clk:** This protocol already was explained in this document, in the [PROT 001].
- **NumberLines:** This protocol represents the conjugated of the NumberLines protocol [PROT 007] that it was explained before in this document.
- **ShiftCursor:** This protocol represents the conjugated of the ShiftCursor protocol [PROT 004] that it was explained before in this document.

[PROT 0013] CursorRight

This protocol represents the interaction between the PrintChar capsule and the Cursor capsule in relation to the sending of the command for the shift the cursor for the right direction.

Figure 45: CursorRight protocol.

In signals:

- **ackShifRigth:** Signal that indicates that the command to shift the cursor to right direction was executed.

Out signals:

- **right:** Signal that indicates the request for to shift the cursor to right.

[SD 017] CursorRight's state diagram

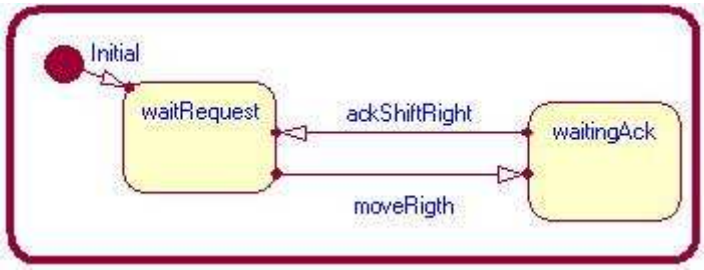

Figure 46: CursorRight's state diagram.

The initial state of the CursorRight protocol is the waitRequest state, the protocol stays in this state until it to receive the moveRight signal; after the receiving of the moveRight signal, the protocol goes to waitingAck state; the protocol stays in the waitingAck until it to send the ackShiftRight signal, after the ackShiftRight signal is sent, the protocol comes back to initial state.

[PROT 0014] Value

This protocol represents the interaction between the Cursor capsule and the DataDisplayArray capsule in relation to the sending of the cursor's position.

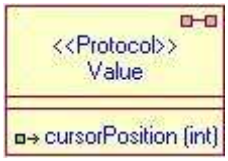

Figure 47: Value protocol.

Out signals:

- **cursorPosition:** It sends the cursor's position.

[SD 018] **Value's state diagram**

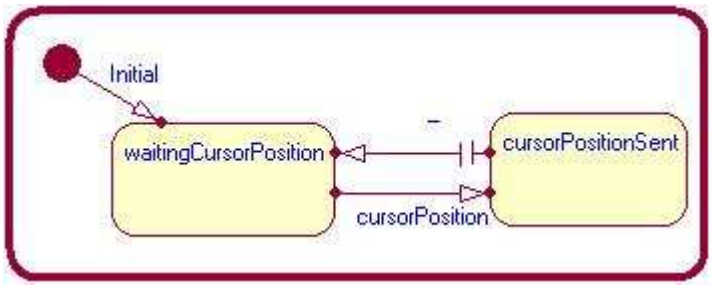

Figure 48: Value's state diagram.

The initial state of the Value protocol is the waitingCursorPosition state, the Value protocol stays in the initial state until the cursorPosition signal is sent; after that Position signal is sent, the protocol goes to cursorPositionSent state and it comes back to initial state after.

6. Driver subsystem

This subsystem represents the LCD Controller and Driver's functionalities related with the character's printing on the display.

6.1 Capsules diagrams

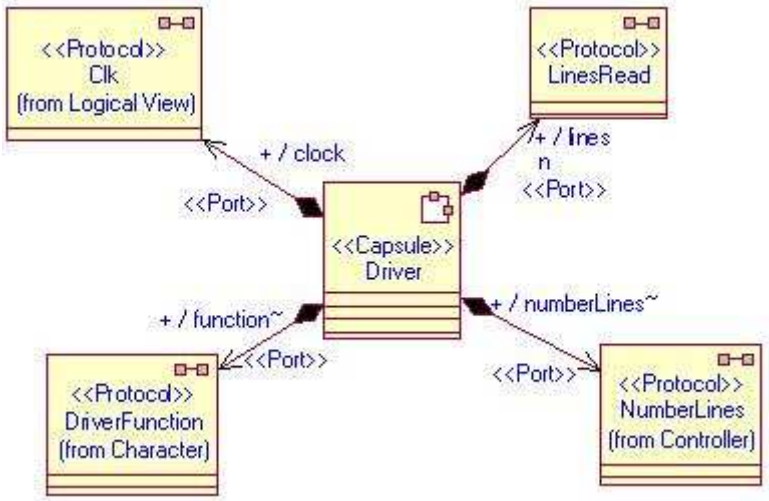

Figure 49: Driver's capsules diagram.

[CAP 007] **Driver**

This capsule is responsible for the control to the reading of the mapCharacters from the DataDisplayArray for the display.

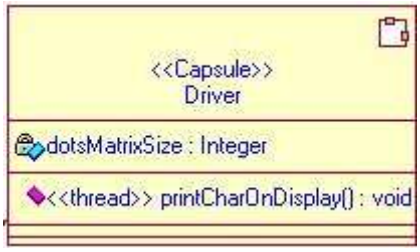

Figure 50: Driver Capsule.

Attributes:

- **dotsMatrixSize:** It indicates the character font that will be printed on the display; if it is 8(eight), the character is represented by 5x8 dots matrix; if it is 10(eight), the character is represented by 5x10 dots matrix.

Operations:

- **printCharOnDisplay:** It controls the reading of the DataDisplayArray capsule and the printing of its characters on the display. The cursor must be printed in the 2 Hz frequency and the characters must be printed in the display at least sixty times per second.

[SD 019] **Driver state diagram**

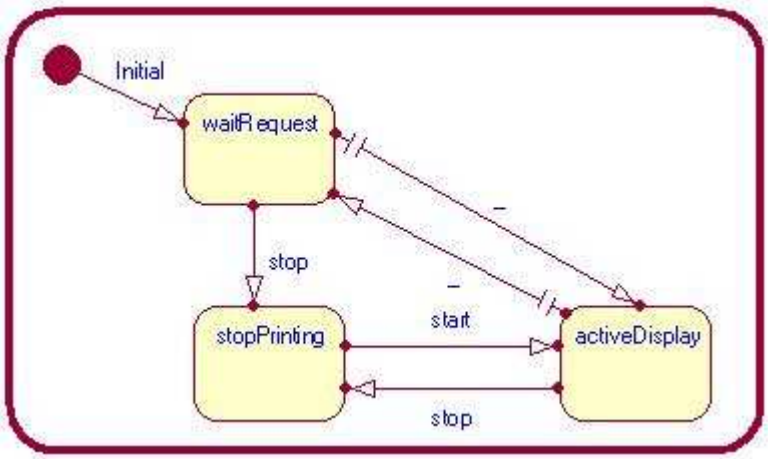

Figure 51: Driver's state diagram.

The initial state of the Driver capsule is the waitRequest state. The Driver capsule stays in the initial state until it sends the activeDisplay signal to the user, indicating that the characters will be printed on the display, after, it sends to the user the line signal, that indicates which line of the characters are printed on the display, if the display has one line, the display will print 8 (lines) that represent the characters printed in this line; if the display has two lines, the display will print sixteen (16) lines that represent the characters printed in the two lines; after that all lines of the characters were printed, the Display capsule comes back to initial state.

If the Display capsule is in the initial state and it receives the stop signal, the Display goes to stopPrinting state where it doesn't printed character and it stays waiting to receive the start signal from the PrintChar capsule for to start print the characters on the display; when it receives the start signal, it goes to activeDisplay state where it sends the activeDisplay and line signals for the user; after all characters were printed on the display, the Display capsule comes back to initial state.

The characters must be printed in the display at least sixty times per second. The cursor is printed on the display according its position, stored in the DataDisplayArray capsule; its position is verified and it is printed using the line and the segment signals appropriate, for example, if the cursor position is between the first and tenth-sixth position, the cursor will be printed in the seventh line and the segment7 signal contains the cursor's mapping. For blink the cursor position character, if the blinking attribute of the DataDisplayArray capsule is true, the cursor position character must blink in the same frequency of the cursor, 2 HZ.

6.2 Protocols

- **Clk:** This protocol already was explained later in this document, in the [PROT 001].
- **NumberLines:** This protocol represents the conjugated of NumberLines [PROT 007] that already it was explained previously in this document.
- **DriverFunction:** This protocol represents the conjugated of the DriverFunction protocol [PROT 0011] that already it was explained previously in this document.

[PROT 0015] LinesRead

This protocol represents the interaction between the Driver capsule and DataDisplayArray capsule in relation to the reading's control of the mapCharacters from the DataDisplayArray capsule.

Figure 52: LinesRead protocol.

Out signals:

- **activeDisplay:** It indicates that the character's mapping stored in the mapCharacters from the DataDisplayArray capsule are been read and printed on the display.
- **line:** It indicates which line of the Character's mapping is been printed on the display.

[SD 020] LinesRead's state diagram

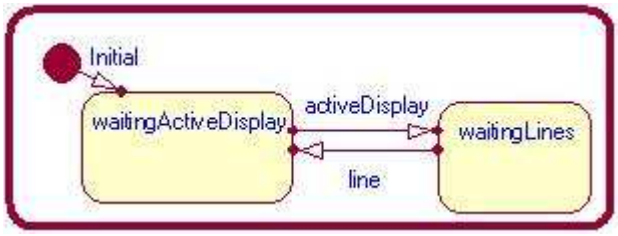

Figure 53: LinesRead's state diagram.

The initial state is the waitingActiveDisplay, the protocol stays in this state until the activeDisplay is sent; after the activeDisplay is sent, the protocol goes to waitingLines state; the protocol stays in the waitingLines protocol until all lines (if the display has 1(one) line, 8(eight) lines will be printed on the display; if the display has 2 (two) lines

on the display, 16(sixteen) lines will be printed on the display) of the character's mapping is printed on the display.

References

[1] LCD Controller and Driver glossary document, version 1.3;

[2] LCD Controller and Driver Use Case Specification, version 2.3;

[3] LCD Controller and Driver Analysis Document, version 2.7;

[4] IPProcess introduction, http://www.cin.ufpe.br/~aaca/ipprocess/ipprocess_introduction.ppt.

Appendix A

This section shows the necessary structure for the communication between the user and LCD Controller and Driver through the OCP-IP interface.

- **Structure Diagram**

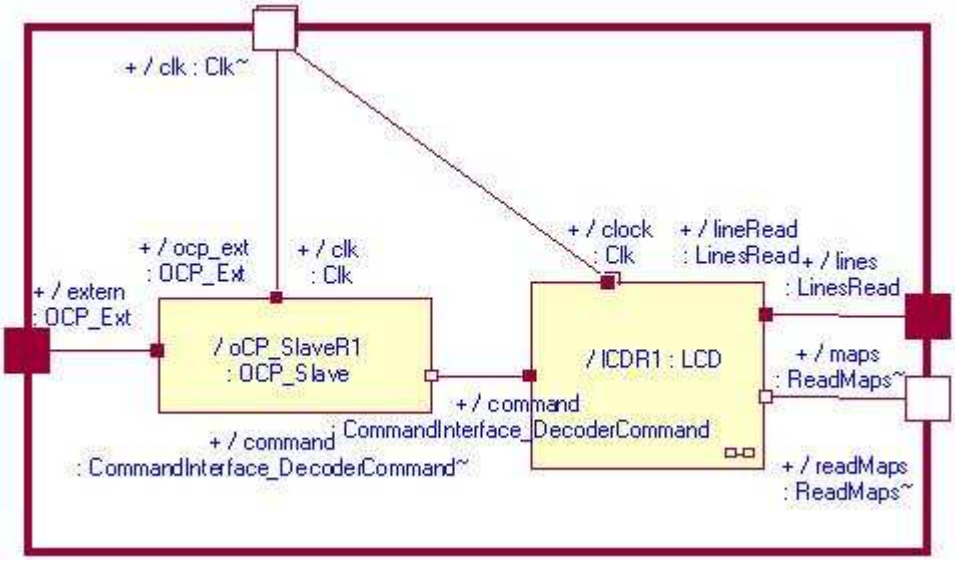

Figure 54: System's structure diagram.

- **Capsules Diagram**

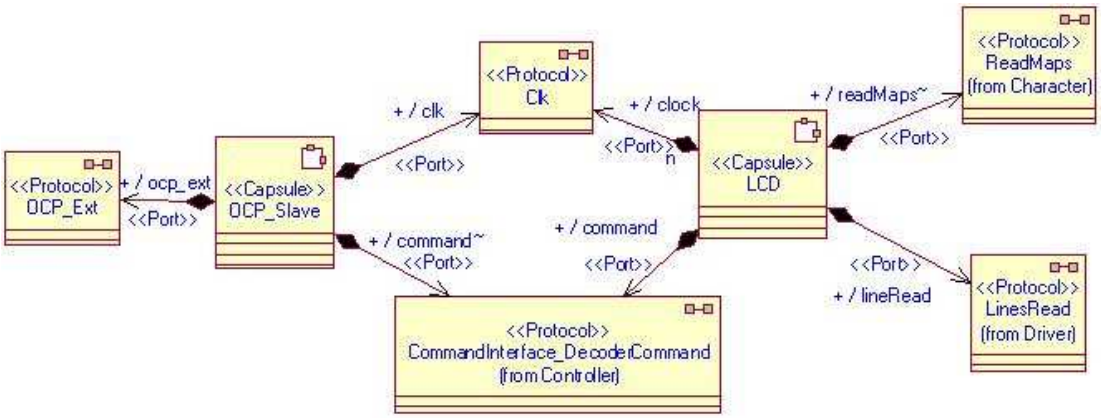

Figure 55: System's capsules diagram.

[CAP 008] OCP_Slave

This capsule represents the interface OCP-IP instance.

Figure 56: OCP_Slave capsule.

[SD 020] OCP_Slave's state diagram

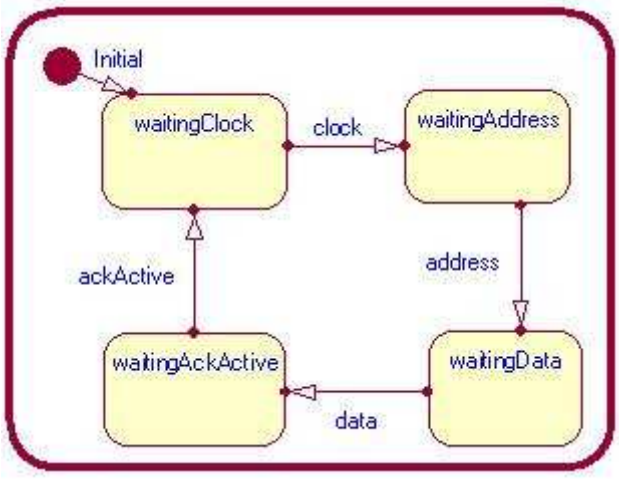

Figure 57: OCP_Slave's state diagram.

The initial state is waitingClock state where the OCP_Slave capsule waits to receive the clock signal; when it receives this signal, it goes to waitingAddress state, it stays in this state until it receives the address signal from user; the next state is waitingData, in this state the OCP_Slave capsule stays waiting the data from the user, this data will be the command that it will sent for the LCD Controller and Driver. After the sending of the data signal for the LCD Controller and Driver, the OCP_Slave goes to waitingAckActive state, it stays in this state until it to receive the ackActive signal, this signal indicates

that the execution of the last command sent by the LCD Controller and Driver finished and other command can be sent to the LCD Controller and Driver; after the receiving of ackActive signal the protocol comes back to initial state.

- **Protocols**

[PROT 0016] **OCP_Ext**

This protocol represents the interaction between the OCP_Slave capsule and the user in relation to commands sending.

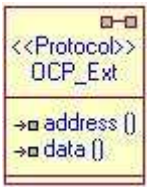

Figure 58: OCP_Ext protocol.

In signals:

- **address:** The transfer address.
- **data:** The command that will be sent for the controller.

[SD 021] **OCP_Ext's state diagram**

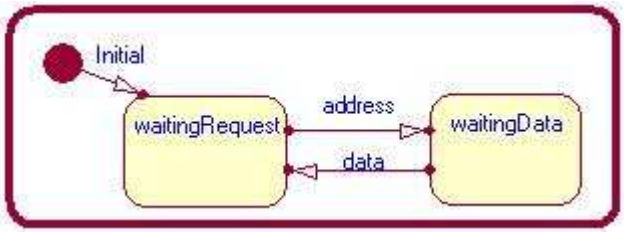

Figure 59: OCP_Ext's state diagram.

The initial state of the OCP_Ext protocol is the waitRequest state, the OCP_Ext stays in this state until it to receive the address signal; after the address signal is received, the OCP_Ext protocol goes to waitingData state; the OCP_Ext protocol stays in this state until it to receive the data signal; after the data signal is sent, the protocol comes back to initial state.

[PROT 0017] **Clk**

This protocol represents the interaction between OCP_Slave capsule and the user in relation to sending of the clock signal.

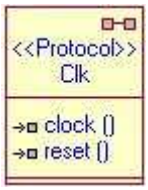

Figure 60: Clk protocol.

In signals:

- **clock:** It represents the system clock.
- **reset:** It represents the system reset signal.

[PROT 0018] CommandInterface_DecoderCommand

This protocol represents the interaction between the LCD Controller and Driver and OCP_Slave capsule in relation to commands sending.

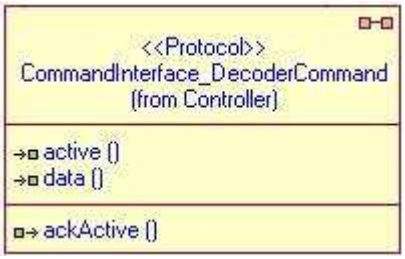

Figure 61: CommandInterface_DecoderCommand protocol.

In signals:

- **active:** It indicates that command sent by the user can be read.
- **data:** Command sent for the LCD Controller and Driver by the user, its type is sc_uint<8>.

Out signals:

- **ackActive:** It indicates the ending of the execution of the last command read by the LCD Controller and Driver.

[SD 022] CommandInterface_DecoderCommand's state diagram

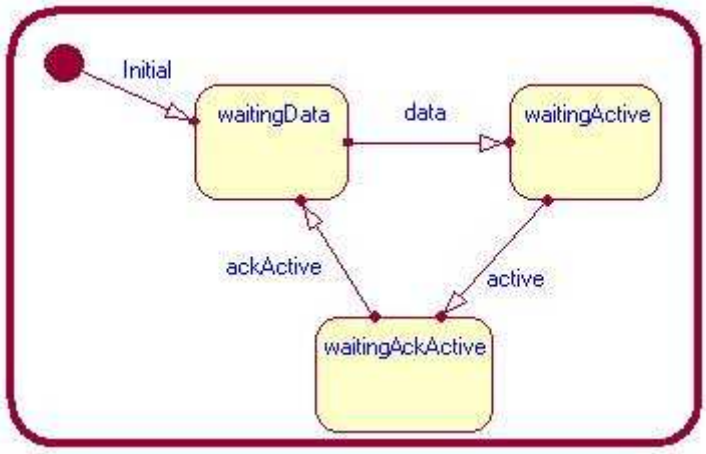

Figure 62: CommandInterface_DecoderCommand's state diagram.

The initial state is the waitingData state, the CommandInterface_DecoderCommand protocol stays in this state until the data signal is sent; after data signal is sent, the protocol goes to waitingActive state; the protocol stays in the waitingActive state until the active signal is sent; the next state is waitingAckActive; the protocol stays in waitingAckActive until the ackActive signal is sent; after the sending of ackActive signal, the protocol comes back to initial state.

ipPROCESS

Verification Plan

LCD Controller and Driver

Project Fênix

BrazilIP

Version : 1.5

Revision History

Date	Version	Description	Author
11/19/2005	1.0	Description of target verification items and verification approach.	Daniele Santos.
12/05/2005	1.1	Revision of figure1.	Daniele Santos.
02/07/2006	1.2	Exclusion of table of definitions, acronyms and abbreviations.	Daniele Santos.
04/19/2006	1.3	Changes in the verification approach topic.	Daniele Santos.
05/16/2006	1.4	Changes in the description of the Test Designer and Test System Administrator.	Daniele Santos.
05/24/2006	1.5	Description of the display specification.	Daniele Santos.

1. Introduction

The purpose of the Verification Plan is to gather all of the information necessary to plan and control the verification effort for a given IP-core. It describes the approach to verify the modules, and is the top-level plan generated and used by managers to direct the verify effort.

This Verification Plan for the LCD Controller and Driver supports the following objectives:

- Tests the LCD Controller and Driver's functionalities;
- Tests the main and secondary flows of LCD Controller and Driver's use cases;
- Identifies the necessary no-human resources to the execution of the test;
- Identifies the necessary professionals for the execution of the test;
- Minimizes redudant effort.

1.1 Scope

This Verification Plan is addressed to the LCD Controller and Driver module, with the objective to test its use cases.

1.2 Document Structure

- Section 2 – Target Verification Items: Provide a high level list of the major target verification items.
- Section 3 – Verification Approach: presents the recommended strategy for designing and implementing the required tests.
- Section 4 – Environmental Needs: presents the non-human resources required for the Verification Plan.
- Section 5 - Responsibilities, Staffing, and Training Needs: presents the required resources to address the verification effort outlined in the Verification Plan.
- Section 6 – References: this section provides a complete list of all documents referenced elsewhere in this document.

2. Target Verification Items

The listing below identifies those verification items that have been identified as targets for testing. This list represents what items will be verified.

- LCD Controller and Driver module.

3. Verification Approach

The verification technique that will be used in the LCD Controller and Driver consists in to generate a set of inputs for the input ports of the DUV and the input ports of the Reference Model and to compare the output responses of the DUV to those of a reference designer.

Stim: It generates and sends to the DUV and to reference model useful input's sequences that will verify that the DUV complies with the specification.

Check: It checks the responses of the design to those of the reference model to verify that design is working correctly.

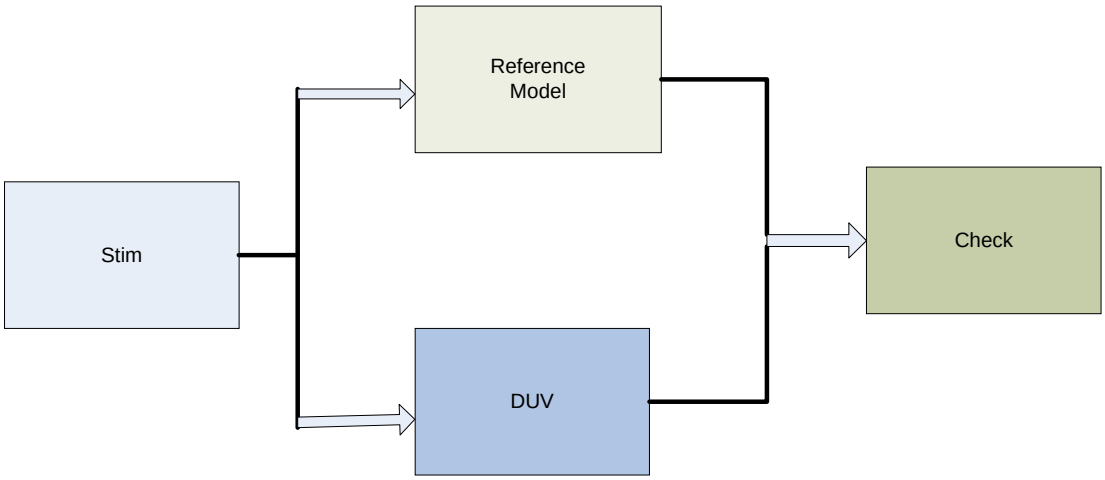

Figure 5: Self-checking testbench.

4. Environmental Needs

4.1 Base System Hardware

The following table sets forth the system resources for the test effort presented in this *Verification Plan*.

Table 13: Base System Hardware.

System Resources		
Resource	**Quantity**	**Name or Type**
LCD Display	1	LCD Display 16x2 Alphanumeric.
Processor 1.5 GHz	1	No preferred type.

4.2 Base Software Elements in the Verification Environment

The following base software elements are required in the verification environment for this Verification Plan.

Table 14: Base Software Elements in the Verification Environment

Software Name	Version	Type and other Notes
Eclipse	3.0	C/C++ Development tools
SystemC, C++ library	2.0.1	
gcc compiler	3.3.6	

5. Responsibilities, Staffing, and Training Needs

5.1 People and Roles

This table shows the staffing assumptions for the verification effort.

Table 15: Human Resources.

Human Resources		
Role	**Minimum Resources Recommended**	**Specific Responsibilities or Comments**
Tester	Knowledge in programming language C++, with in emphasis in SystemC library; knowledge in operational system Linux.	Implements and executes the verification. Responsibilities include: • implement verification components • execute verification • log results • analyze and recover from verification failures • document incidents
Test Designer	Knowledge in tests process.	Identifies and defines the verification approach. Responsibilities include: • define the target verification items • define the levels of verification • define test cases, for functional verification
Test System Administrator	Knowledge in tests process.	Ensures verification environment and assets are managed and maintained. Responsibilities include: administer verification management system install and support access to, and recovery of, verification environment configurations and labs

5.2 Staffing and Training Needs

This section outlines how to approach staffing and training the verification roles for the project.

- Training in programming languagem C++, with emphasis in the C++ library SystemC;
- Training in LINUX operational system.

6. References

[1] KEATING M. Reuse Methodology Manual, Third Edition.

[2]LCD Controller and Driver's Glossary document.

ipPROCESS

Functional Verification Test Cases

LCD Controller and Driver

Project Fênix

BrazilIP

Version : 2.3

Revision History

Date	Version	Description	Author
11/17/2005	1.0	Decision of the topics that if apply for the test case used.	Daniele Santos.
11/19/2005	1.1	Description of Compliance test case and Corner test case.	Daniele Santos.
11/27/2005	1.2	Description of Random test case and Corner test case.	Daniele Santos.
01/14/2006	1.3	Description of the input and output of test cases.	Daniele Santos.
01/27/2006	1.4	Description of the input and output of test cases.	Daniele Santos.
02/07/2006	1.5	Table's legends and table index.	Daniele Santos.
04/10/2006	1.6	Insertion of new tables.	Daniele Santos.
04/11/2006	1.7	Description of the test cases of the following use cases: Clear Display, Return Home and Cursor or Display shift. Changes in the test cases of the Print char on display.	Daniele Santos.
04/17/2006	1.8	Description of test case files.	Daniele Santos.
04/28/2006	1.9	Description of the RA002 test file	Daniele Santos.
05/18/2006	2.0	Changes in the test case's description and description of the appendix A.	Daniele Santos.
05/19/2006	2.1	Legend in the test files table.	Daniele Santos.
05/24/2006	2.2	Change in the CA03 test file	Daniele Santos.
08/02/2006	2.3	Change in the test files.	Daniele Santos.

1. Introduction

The purpose of the Test Cases is to gather all of the information necessary to plan and define the functional verification for LCD Controller and Driver. It describes the input/output data set and the coverage criteria for the functional verification.

1.1 Scope

These test cases are addressed to the LCD Controller and Driver module, with the objective to test its functionalities.

1.2 Document Structure

This subsection describes what the rest of this document contains and explains how the document is organized.

- Section 2 – Test Cases: this section contains the set of input/output data that will be used as the stimuli in the test bench.
- Section 3 – Coverage: this section defines the coverage criteria.
- Section 4 - References: this section provides a complete list of all documents referenced elsewhere in this document.
- Appendix A - Test files: it shows the test files used in the LCD Controller and Driver's verification.

2. Test Cases

The listing below identifies and defines the test cases (stimulus) that have been identified for the LCD Controller and Driver. This list represents items that will be used in its test bench.

2.1 Compliance Test Cases

Table 16: Print char on display's compliance test.

Test Case Name	[CA 001] Print char on display test
Description:	The objective of this test case is to test the main flow of *Print char on display* use case, for this, codes that represent valid characters are generated in the stimulus module to verifies if the LCD Controller and Driver prints them correctly.
Related use case:	[UC01] Print char on display
Pre condition:	• The LCD Controller and Driver must be on. • The codes sent to the LCD Controller and Driver must represent valid characters.
Pos condition:	• Characters references to the code sent by user must be printed in the display and their mappings are stored in the DDRAM memory.

Test Procedure:	• Starts the LCD Controller and Driver; • Give the reset command; • Send a code of 8 (eight) bits that represents a character to the LCD Controller and Driver input; • Verifies if the LCD Controller and Driver printed the correct character in the display, comparing the LCD Controller and Driver outputs with the generated output by reference model. • Repeats the steps described above until all the valid codes have been tested.
INPUT: verification/testcases/CA001/compliance.in	OUTPUT: verification/testcases/CA001/compliance.out

Table 17: Clear Display's compliance test.

Test Case Name	**[CA02] Clear Display**
Description:	The objective of this test case is to test the main flow of *Clear Display* use case, for this, thirty-two codes that represent valid characters are generated in the stimulus module and send for the LCD Controller and Driver, after the Clear Display command is sent and it verifies if the display is clean.
Related use case:	[UC02] Clear Display
Pre condition:	• The LCD Controller and Driver must be on; • Characters must be printed on the display, i.e., mapping of the characters must be stored in the DDRAM memory.
Pos condition:	• There aren't characters in the display and the thirty-two positions of the RAM memory store the space character.
Test Procedure:	• Starts the LCD Controller and Driver; • Send thirty-two codes of the valid characters for the LCD Controller and Driver; • Send the Clear Display command for the LCD Controller and Driver; • Verify if the display is clean.
INPUT: verification/testcases/CA002/compliance.in	OUTPUT: verification/testcases/CA002/compliance.out

Table 18: Return Home's compliance test.

Test Case Name	[CA03] Return Home	
Description:	The objective of this test case is to test the main flow of *Return Home* use case, for this, thirty-two codes that represent characters are sent for the LCD Controller and Driver, after, the *Return Home* command is sent and it verified if the cursor is in the first position of the first line.	
Related use case:	[UC03] Return Home	
Pre condition:	• The LCD Controller and Driver must be on; • The cursor was shifted.	
Pos condition:	• The cursor is in the first position in the first line.	
Test Procedure:	• Starts the LCD Controller and Driver; • Send thirty-two codes that represent valid characters to the LCD Controller and Driver; • Send the Return Home command to the LCD Controller and Driver; • Verify if the cursor is in the first position of the first line.	
INPUT: verification/testcases/CA003/compliance.in		OUTPUT: verification/testcases/CA003/compliance.out

Table 19: Cursor shift's compliance test.

Test Case Name	[CA04] Cursor shift
Description:	The objective of this test case is to test the main flow of C*ursor shift* use case, for this, Cursor Shift commands are sent for the LCD Controller and Driver and it verifies if the cursor is in the correct position.
Related use case:	[UC04] Cursor or Display shift
Pre condition:	• The LCD Controller and Driver must be on.
Pos condition:	• The position of the cursor changes according with the shift command received.
Test Procedure:	• Starts the LCD Controller and Driver; • Sends thirty-one shift cursor command to right direction; • Sends thirty-one shift cursor command to left direction; • Sends thirty-one codes that represent characters for the LCD Controller and Driver; • Sends thirty-one shift cursor command to left direction;

	• Sends thirty-one shift cursor command to right direction; • Verify if the cursor shifted correctly.
INPUT: verification/testcases/CA004/compliance.in	OUTPUT: verification/testcases/CA004/compliance.in

2.2 Corner Test Cases

Table 20: Print char on display's corner test.

Test Case Name	**[CO01] Print char on display test**
Description:	The objective of this test case is to test the secondary flows of *Print char on display* use case, for this, thirty -three codes representing valid characters are sent for the LCD Controller and Driver, to verify if the characters and the cursor are printed correctly.
Related use case	[UC01] Print char on display
Pre condition:	• The LCD Controller and Driver must be on. • There aren't characters printed on the display.
Pos condition:	• Thirty-two mapping of the characters must be stored in the DDRAM memory and the cursor's position is the last position of the display.
Test Procedure:	• Starts the LCD Controller and Driver; • Give the reset command; • Send sixteen code of 8 (eight) bits that represent a character to the LCD Controller and Driver input; • Verifies if the cursor is in the first position of the second line. • Send more seventeen code of 8 (eight) bits that represent a character to the LCD Controller and Driver input; • Verifies if the thirty-two characters references to the first thirty-two codes sent for the display are printed on the display and if the cursor is in the last position of the display.
INPUT: verification/testcases/CO001/corner.in	OUTPUT: verification/testcases/CO001/corner.out

Table 21: Clear Display's corner test.

Test Case Name	**[CO02] Clear Display test**	
Description:	The objective of this test case is to test the secondary flows of *Clear Display* use case, for this, the instruction Clear Display will execute when the display is clean, i.e., when nothing character is in the display.	
Related use case:	[UC02] Clear Display	
Pre condition:	• The LCD Controller and Driver must be on. • There aren't characters printed on the display, i.e., there aren't characters stored in the DDRAM memory.	
Pos condition:	• The state of the memory and of the display doesn't change.	
Test Procedure:	• Starts the LCD Controller and Driver; • Give the reset command; • Send the Clear Display command; • Verifies if the display is clean.	
INPUT: verification/testcases/CO002/corner.in		OUTPUT: verification/testcases/CO002/corner.out

Table 22: Return Home's corner test.

Test Case Name	**[CO03] Return Home test**	
Description:	The objective of this test case is to test the secondary flows of *Return Home* use case, for this, the instruction Return Home will execute when the position of the cursor is in the display's 1st digit of the first line.	
Related use case:	[UC03] Return Home	
Pre condition:	• The LCD Controller and Driver must be on. • The cursor is in the display's 1st digit of the first line.	
Pos condition:	• The position of the cursor is in the display's 1st digit of the first line.	
Test Procedure:	• Starts the LCD Controller and Driver; • The cursor must be in the first position of the first line of the display, for this, codes representing valid characters aren't sent for the LCD Controller and Driver. • Send the Return Home command; • Verifies if the cursor continues in the first position of the first line.	
INPUT: verification/testcases/CO003/corner.in		OUTPUT: verification/testcases/CO003/corner.out

Table 23: Cursor shift's corner test.

Test Case Name	[CO04] Cursor Shift test	
Description:	The objective of this test case is to test the secondary flows of *Cursor Shift* use case, for this, the instruction Cursor Shift will execute when the position cursor is in the 1st digit in the first line, the position of the cursor is in the last digit in the first line, the position cursor is in the 1st digit in the second line, and the position cursor is in the last digit in the second line.	
Related use case:	[UC04] Cursor or Display shift	
Pre condition:	• The LCD Controller and Driver must be on; • The cursor is in the display's 1st digit of the first line.	
Pos condition:	• The cursor is shifted according the command received.	
Test Procedure:	• Starts the LCD Controller and Driver; • Send a command *shift cursor*, shifting the cursor to the left direction; • Verifies if the cursor continues in the first position of the first line; • Send more sixteen commands *shift cursor,* shifting the cursor for the right direction; • Verifies if the cursor is in the first position of the second line; • Send the command *shift cursor,* shifting the cursor for the left direction; • Verifies if the cursor is in the last position of the first line; • Send the command *shift cursor,* shifting the cursor for the right direction; • Verifies if the cursor is in the first position of the second line; • Send sixteen commands *shift cursor*, shifting the cursor to right direction; • Verifies if the cursor is in the last position of the second line.	
INPUT: verification/testcases/CO004/corner.in		OUTPUT: verification/testcases/CO004/corner.out

2.3 Random Test Cases

Table 24: Print char on display's random test.

Test Case Name	[RA01] Print char on display test
Description:	Codes that represent and don't represent characters are generating randomly in the stimulus module to verify if the LCD Controller and Driver prints them correctly.

Related use case:	[UC01] Print char on display
Pre condition:	• The LCD Controller and Driver must be on. • There aren't characters printed on the display, i.e., there aren't mappings of the characters stored in the DDRAM memory.
Pos condition:	• Mapping of the characters references to the valid codes (codes that represent characters) sent to the LCD Controller and Driver are stored in the DDRAM memory.
Test Procedure:	• Starts the LCD Controller and Driver; • Give the reset command; • Generates randomly codes of 8 (eight) bits that represent and don't represent a valid character and sends them to the LCD's Controller and Driver input. • Verify if only the characters references to the valid codes sent by the user was printed on the display.
INPUT: verification/testcases/RA001/random.in	OUTPUT: verification/testcases/RA001/random.out

Table 25: Random test of the commands.

Test Case Name	**[RA02] Commands test**
Description:	Codes representing characters or one of the following commands: Clear Display, Return Home and Cursor or Display shift are generate randomly in the stimulus module to verifies if the LCD Controller and Driver execute them correctly.
Related use case:	[UC01] Print char on display, [UC02] Clear Display, [UC03] Return Home and [UC04] Cursor or Display shift.
Pre condition:	• The LCD Controller and Driver must be on. • The LCD Controller and Driver must receive the reset command by the user.
Test Procedure:	• Starts the LCD Controller and Driver; • Give the reset command; • Generates randomly codes of 8 (eight) bits that represent a valid character or one of the following instructions: clear display, return home, cursor shift, it sends them to the LCD's Controller and Driver input; • Verify if the instructions received by the LCD Controller and Driver were executed correctly.
INPUT: verification/testcases/RA002/random.in	OUTPUT: verification/testcases/RA002/random.out

3. Coverage Criteria

Below, it explains the coverage criteria's used in the realization of test cases cited in this document.

3.1 Functional Coverage

The functional coverage criteria was tests the following use cases: Print Char on display, Clear Display, Return Home and Cursor or Display shift for complete, observing if their main flows and secondary flows are realized correctly, verifying if their functionalities are correct.

3.2 Code Coverage

It wasn't made the measurement of the code coverage carried through for the tests describes in this document.

4. References

[8] LCD Controller and Driver Use Case specification, version 2.1;

[9] KEATING M. Reuse Methodology Manual, Third Edition.