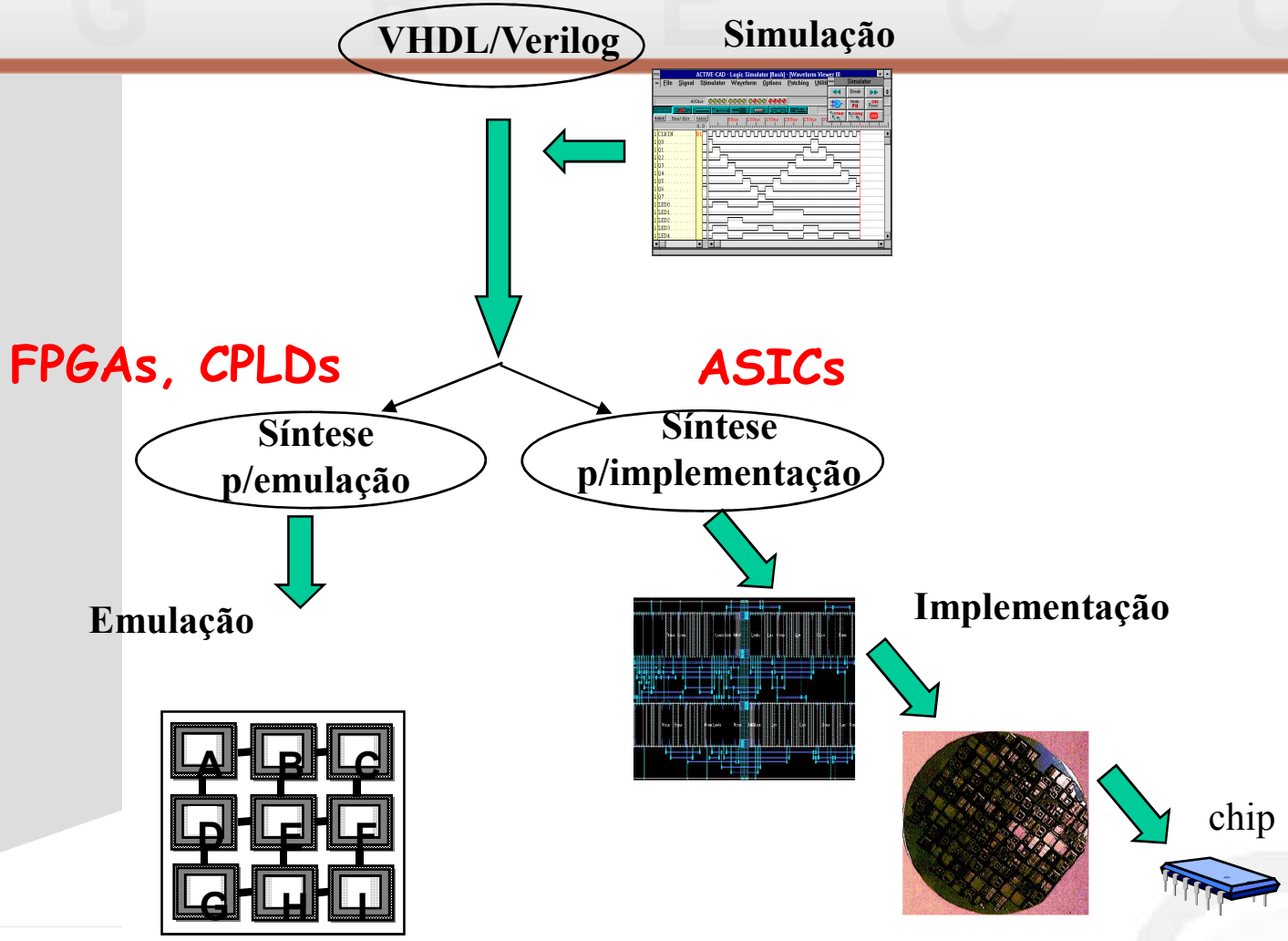
Prototipação de Sistemas Digitais

Metodologia de Projetos

Cristiano Araújo

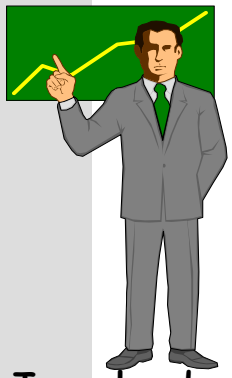
Fluxo de projeto Emulação/Implementação do componente de harwdare



- Mesma base de dados para emulação e implementação
- Testes durante a emulação seriam mais próximos possíveis da implementação final do projeto.

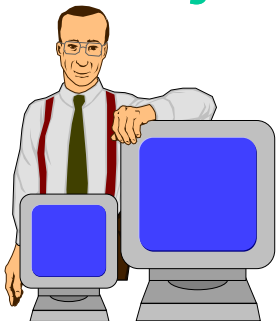
Comunicação: Cliente x projetista

Vendedor



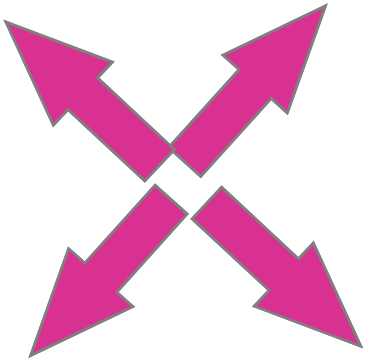
- Janela de mercado
- preço, etc.

Projetista

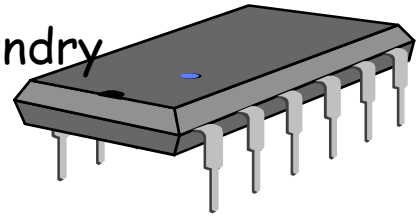


Ambiente de Projeto

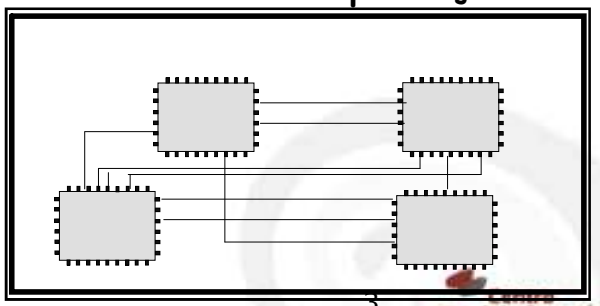
- Ferramentas de CAD
- Engenharia concorrente



- Tecnologia
- Silicon foundry



- Aplicação

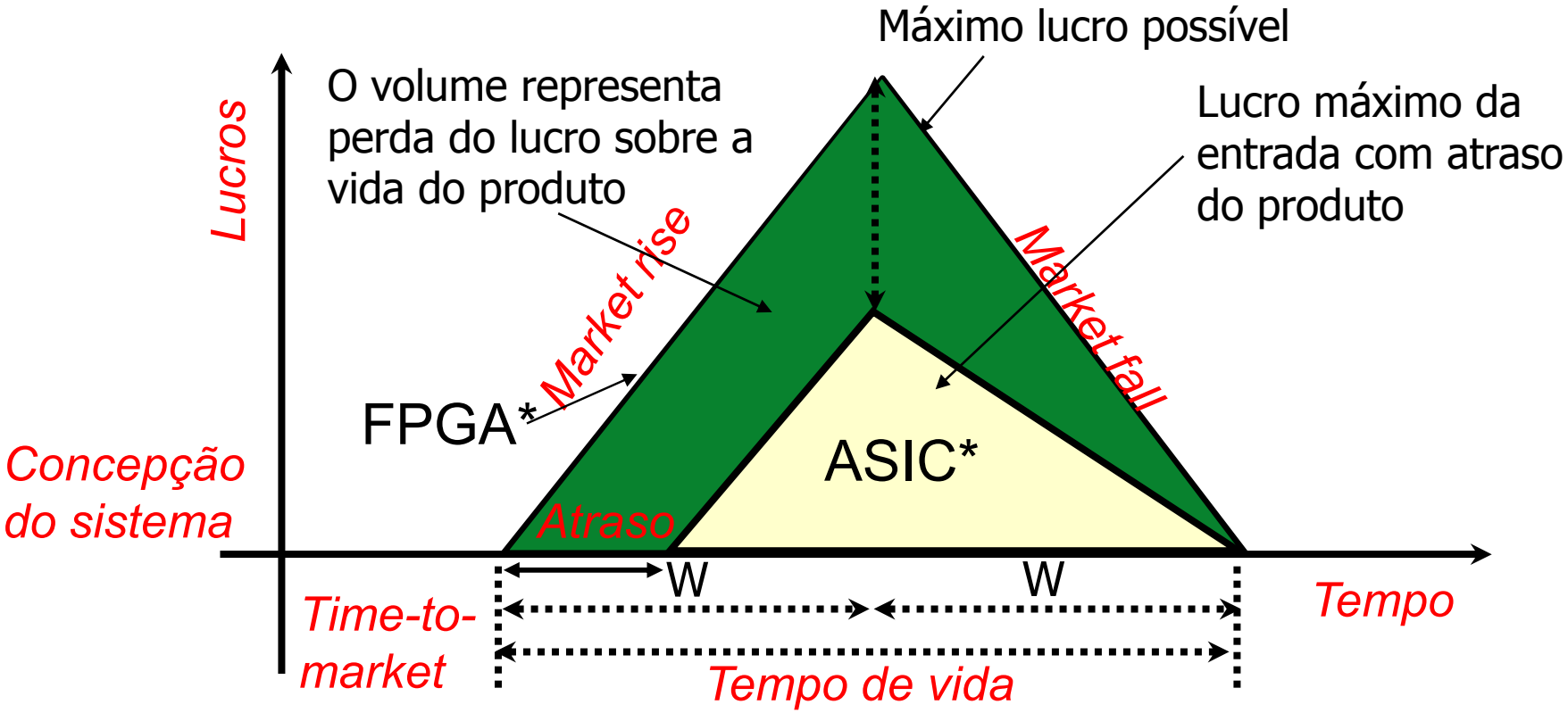


Dispositivo (chip) **Sistema**

Time-to-market

- Em um mercado competitivo, qualquer atraso incorpora perda da parte deste mercado:
  - Perda da janela de mercado
  - Atraso para lançamento em função do longo ciclo de desenvolvimento
  - O efeito da perda em lucro devido o atraso no lançamento do produto é maior que aquele custo de desenvolvimento

Time-to-market



O percentual de perda de lucro, do lucro possível, é dado pela área do maior retângulo menos a área do menor retângulo.

O que é Metodologia de Projeto?

São seqüências de transformações que partem de uma descrição e/ou especificação inicial de um sistema digital para se chegar a uma descrição final, válida para o processo de fabricação deste sistema, no menor espaço de tempo possível.

Metodologia

Conjunto de regras

Procedimentos

Gerência

Metodologia de projetos

Conjunto de regras
- Seqüência de Etapas
- Forma de Apresentação dos Resultados
- Linguagem Utilizada

Procedimentos
- Especificação
- Projeto de alto nível
- Implementação
- Teste

Gerência
- Organização
- Integridade de Dados
- Atribuições de Atividades
- Cronograma
- Controle de Qualidade

Metodologia Top-Down (Fluxo de projeto ASIC)

Síntese de hardware

Projeto em HDL

Especificação do projeto

Descrição comportamental

Descrição RTL

Vetores de verificação

Verificação funcional RTL

não

sim

Síntese do projeto HDL

RTL -> Lógica

Restrições

Otimização lógica

Lógica para Tecnologia

Restrições

Otimização Timing/Área

Inserção de Scan Path & Geração de Vetor de teste

Netlist -Verificação Lógica e Timing

não

sim

Implementação do projeto

Floor Planning

Place & Route

Layout Físico

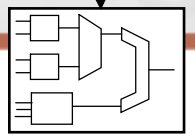
Layout Funcional e Timing

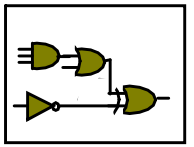
não

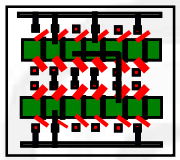
sim

Produção do Chip

```
Process(d,clk)
Begin
    if clk='1' then
      Q<=d;
    end if;
end Process;
```

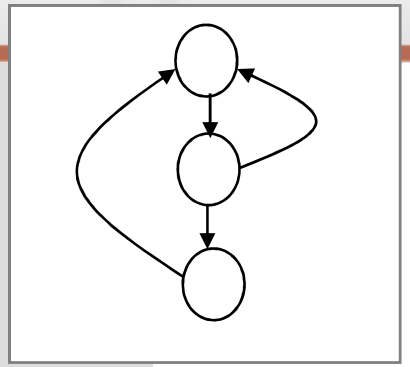




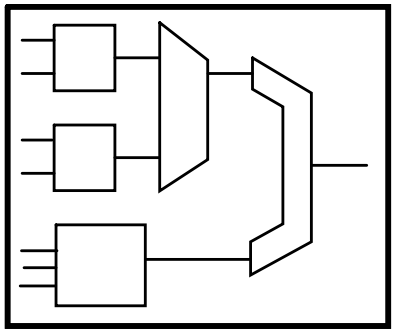


Níveis de abstração

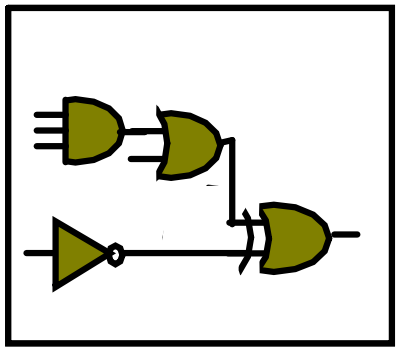
Comportamental



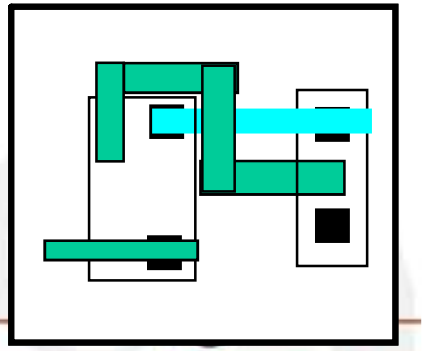
Transf. Registro



Estrutural



Layout Físico



Níveis de abstração para descrição de um projeto

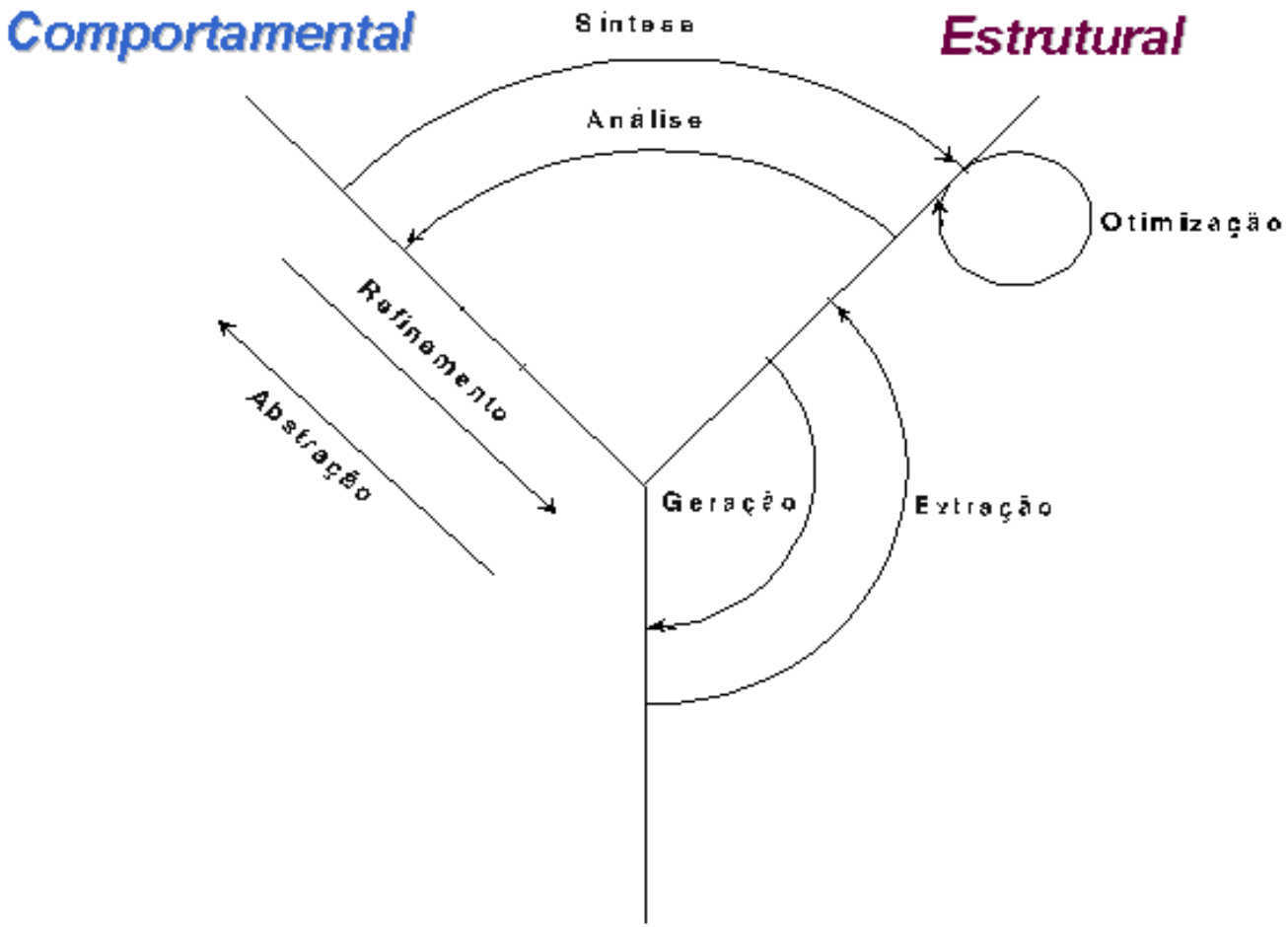
- **Descrição Comportamental**
  - O sistema é descrito de forma comportamental, ou seja o modelamento comportamental que aponta algoritmicamente a funcionalidade requerida sem referenciar a estrutura do modelo.
- **Descrição RTL ou Data Flow**
  - Nível de abstração seguinte abaixo da descrição comportamental. É caracterizado pela definição de um sistema em termos de registradores, chaves e unidades funcionais. Uma descrição RTL define uma arquitetura e incorpora a noção de relógio.
- **Descrição estrutural (Lógico)**
  - Este nível é o mais baixo nível não físico para representação de um projeto. As descrições mostram a implementação lógica da função descrevendo a arquitetura para implementar o algoritmo comportamental.
- **Descrição Física**
  - Neste nível é definida a geometria de implementação do circuito. Este nível depende da tecnologia.

Níveis de descrição de sistemas eletrônicos
Diagrama de Gajski

- **Domínios**
  - Comportamental
    - O comportamento do sistema é especificado, idealmente sem qualquer referência ao caminho que este comportamento vai ser implementado.
  - Estrutural
    - Trata o sistema como como uma hierarquia de elementos funcionais e suas interconexões.
  - Físico
    - A estrutura do projeto é mapeada no espaço, sem qualquer referência a funcionalidade do circuito.

Diagrama de Gajski (Y)

Diagrama de Gajski
(Transição no diagrama Y definindo passos do projeto)



Síntese de hardware
Fluxo de projeto de ASICs

- Entrada da especificação
- Análise
- Otimização tecnológica
- Verificação de projeto
- Layout

Metodologia Top-Down (Fluxo de projeto ASIC)

Síntese de hardware

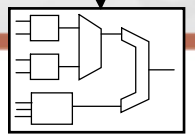
Projeto em HDL

Especificação do projeto

Descrição comportamental

Descrição RTL

```
Process(d,clk)
Begin
    if clk='1' then
      Q<=d;
    end if;
end Process;
```



Vetores de verificação

Verificação funcional RTL

não

sim

Síntese do projeto HDL

RTL -> Lógica

Restrições

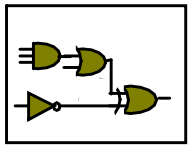
Otimização lógica

Lógica para Tecnologia

Restrições

Otimização Timing/Área

Inserção de Scan Path & Geração de Vetor de teste



Netlist -Verificação Lógica e Timing

não

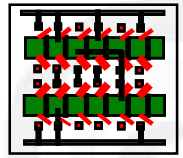
sim

Implementação do projeto

Floor Planning

Place & Route

Layout Físico



Layout Funcional e Timing

não

sim

Produção do Chip

Níveis do projeto

HDL View

```
process(CLK)
  begin
    if (CLK = '1') and (not CLK'stable) then
      s_counter_output <= s_counter_input and not s_reset;
      s_ref_ctr_out <= s_load;
      end if;
end process;
```

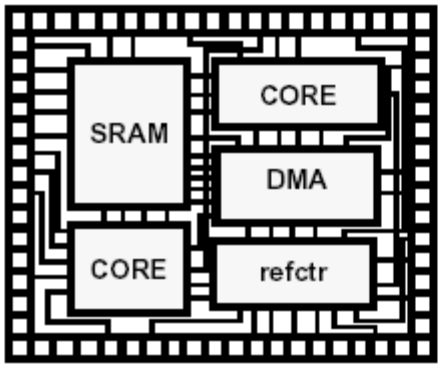
Independente da tecnologia

Netlist View

```
begin
  U68: INVERT_A port map (Z => s_load, A => n265);
  U87 : NOR3_4 port map (Z => n275, A => COUNTb(3),
  B => COUNTb(4), C => COUNTb(0));
  U88 : NOR3_4 port map (Z => n275, A => COUNTb(3),
  B => COUNTb(4), C => COUNTb(0));
  s_ref_ctr_out_reg : D_F_LPH0001_4 port map (L2 =>
  s_ref_ctr_out, D => s_load, E => CLKb);
end SYN_refctr_rtl;
```

Dependente da tecnologia

Physical View



Dependente da tecnologia

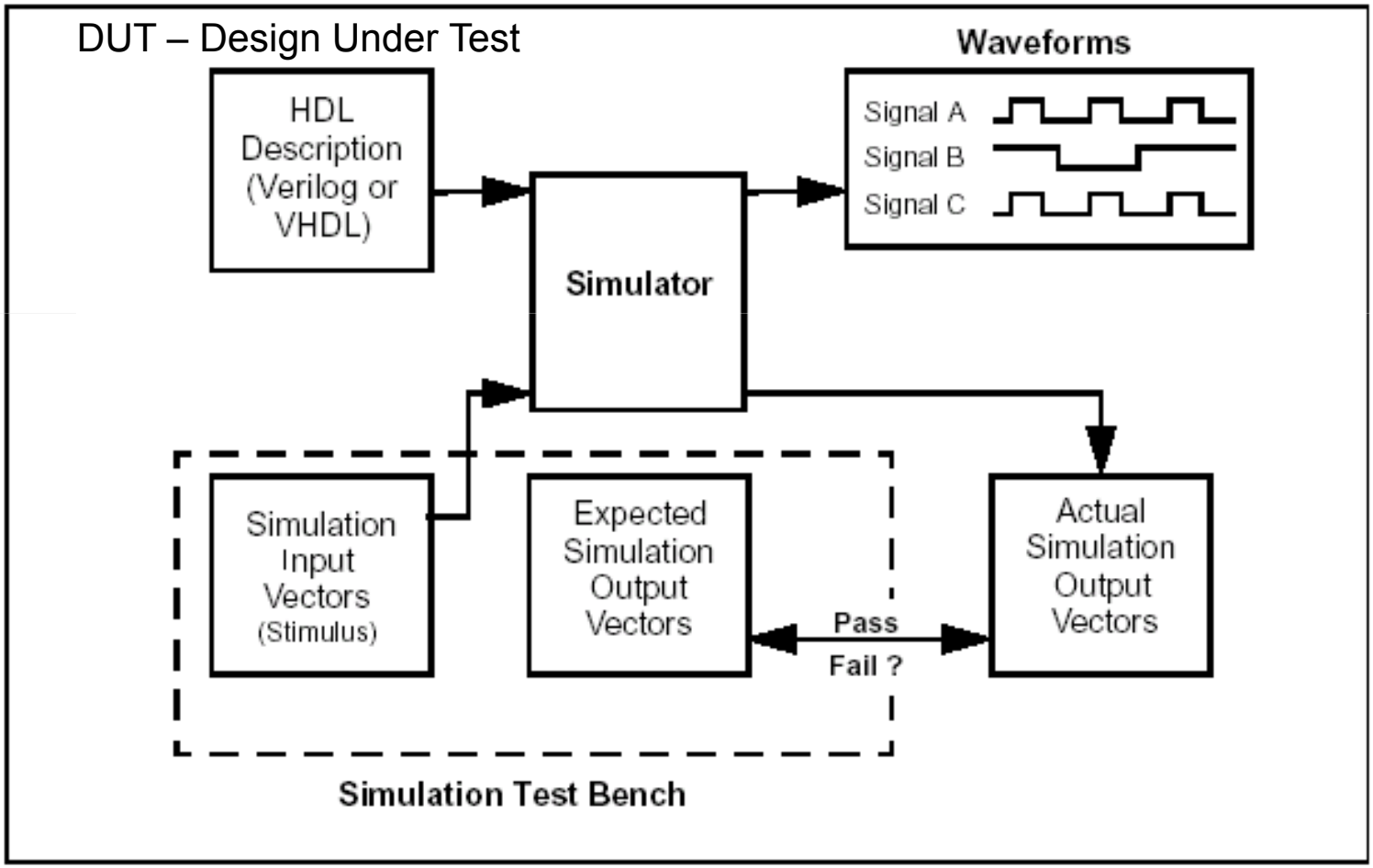
Exemplo – projeto

Entrada VHDL (contador)
(independente da tecnologia)

```
entity refctr is
    port (COUNT: in   std_ulogic_vector(5 downto 0);
          CLK: in std_ulogic;
          RESET: out std_ulogic);
          ....

architecture refctr_rtl of refctr is
    signal s_ref_ctr_out   : std_ulogic;
    signal s_load          : std_ulogic;
    s_next_ctr_val         : std_ulogic_vector(5 downto 0);
    s_counter_input        : std_ulogic_vector(5 downto 0);
    s_counter_output       : std_ulogic_vector(5 downto 0);
    s_reset                : std_ulogic_vector(5 downto 0);
begin
    s_reset(0) <= RESET;
          ....
process(CLK)
       begin
          if (CLK = '1') and (not CLK'stable) then
             s_counter_output <= s_counter_input and not s_reset;
             s_ref_ctr_out <= s_load;
          end if;
        end process;
          ....
end refctr_rtl;
```

IBM

Simulação - funcional



Metodologia Top-Down (Fluxo de projeto ASIC)

Síntese de hardware

Projeto em HDL

Especificação do projeto

Descrição comportamental

Descrição RTL

Vetores de verificação

Verificação funcional RTL

não

sim

Síntese do projeto HDL

RTL -> Lógica

Restrições

Otimização lógica

Lógica para Tecnologia

Restrições

Otimização Timing/Área

Inserção de Scan Path & Geração de Vetor de teste

Netlist -Verificação Lógica e Timing

não

sim

Implementação do projeto

Floor Planning

Place & Route

Layout Físico

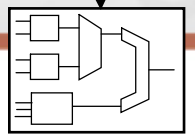
Layout Funcional e Timing

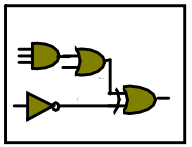
não

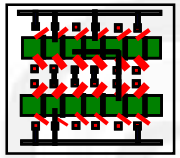
sim

Produção do Chip

```
Process(d,clk)
Begin
    if clk='1' then
      Q<=d;
    end if;
end Process;
```

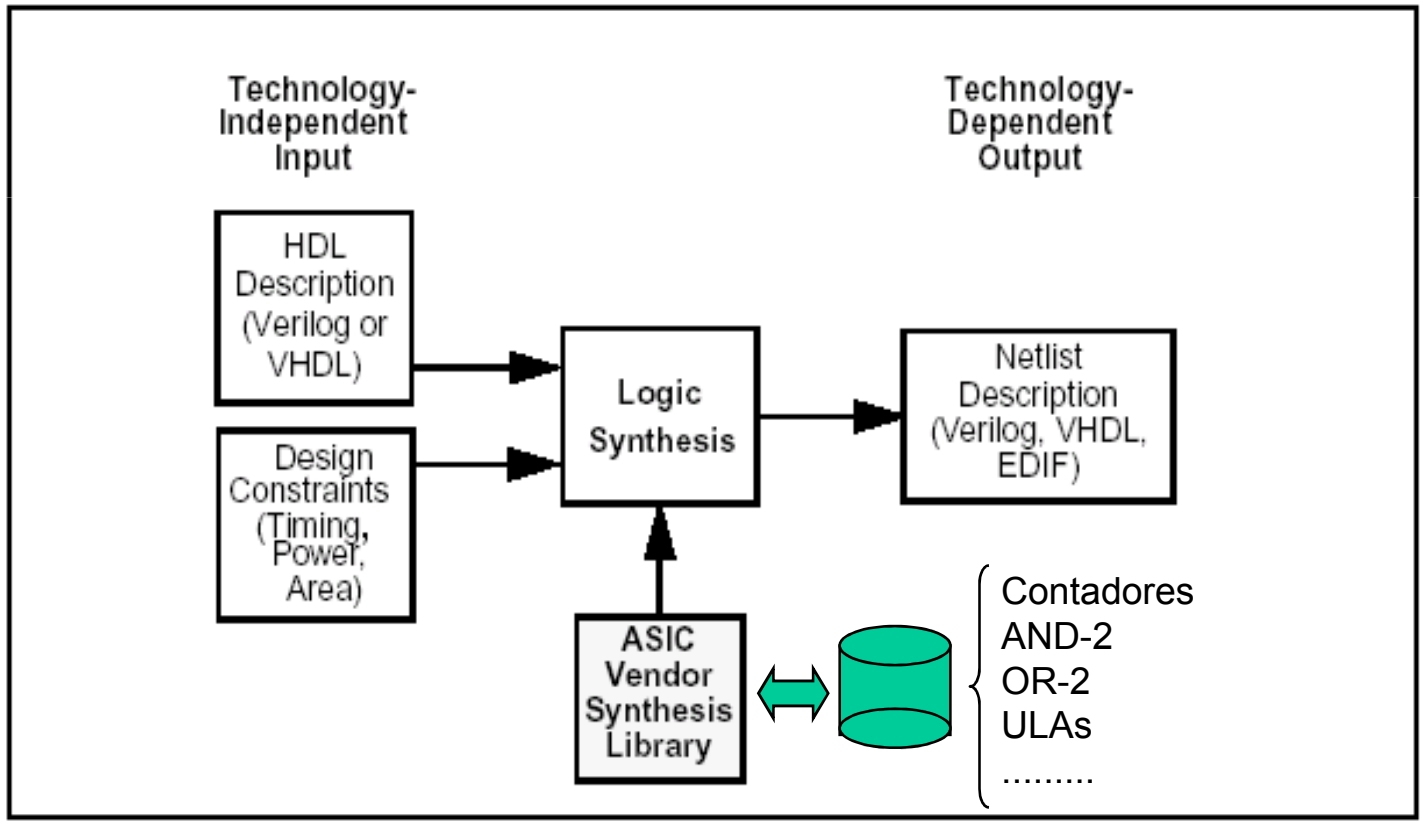






Síntese Lógica

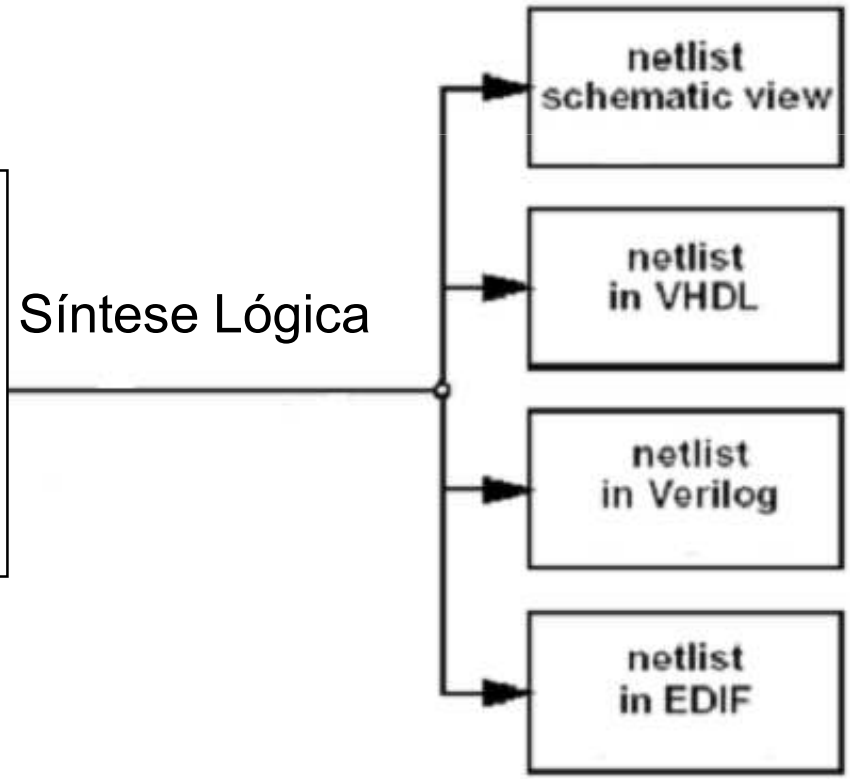
- Síntese lógica tem como entrada uma descrição HDL-RTL, independente de tecnologia e, mapea o projeto em uma biblioteca de circuitos lógicos (células) fornecidos pelo fabricante.



Síntese Lógica – Possíveis saídas

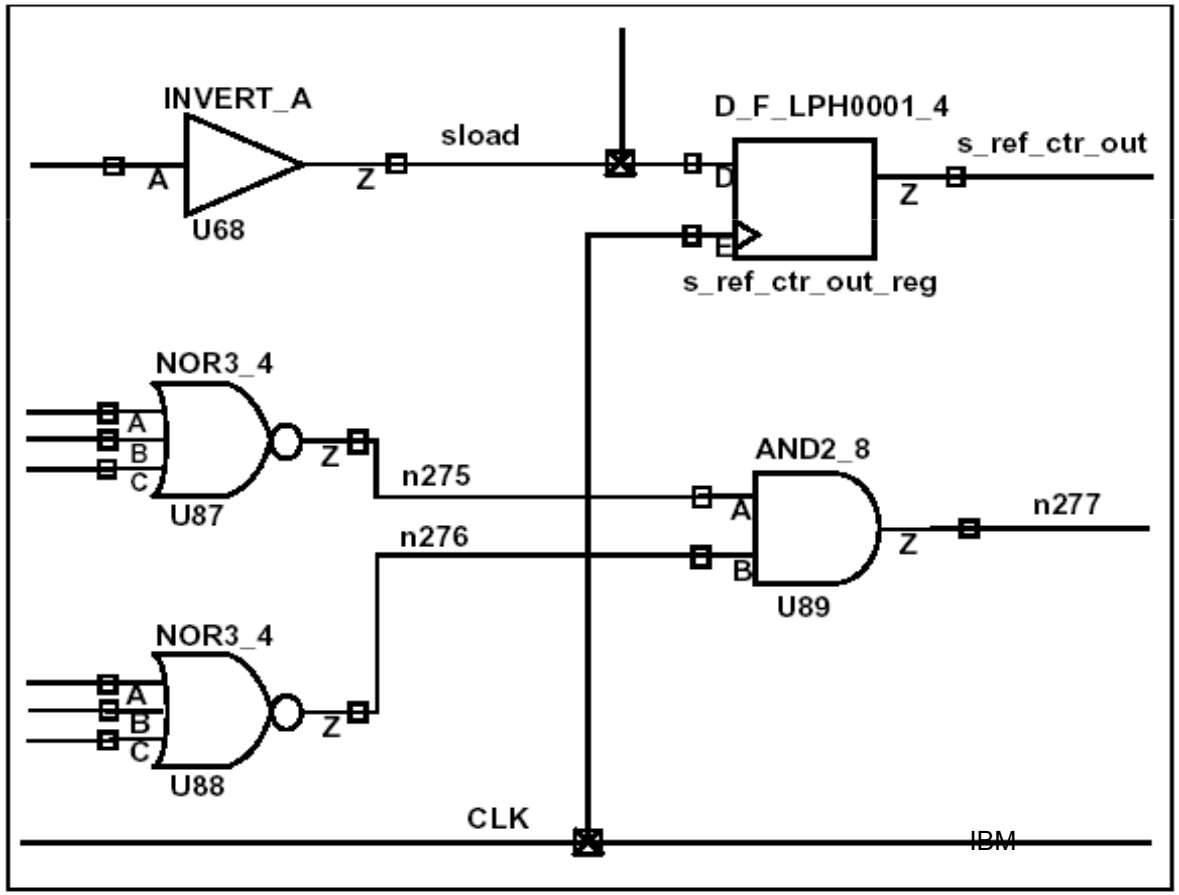
- Como saída, a síntese lógica gera uma netlist que pode ser escrita em diferentes linguagens.

```
entity bombom01 is
port( reset, clk, c, d : in bit;
.............................
architecture arc_bombons of bombom01
is
.............................
begin
 process (clk)
   begin
    if (reset = '1') then
      estado_atual <= tem_zero;
      libera_bombom <= '0';
   elsif (clk = '1' and clk'event) then
```



Visão- Esquemática

- Os circuitos podem ser representados graficamente por um esquema baseado
  nas células básicas da biblioteca do fabricante.



Visão – Geração da Netlist

VHDL

```
entity refctr is
    ...
architecture SYN_refctr_rtl of refctr is
    ...
component INVERT_A
 port(Z : out std_logic; A : in std_logic);
end component;
component NOR3_4
 port(Z : out std_logic; A, B, C : in std_logic);
end component;
component AND2_8
 port(Z : out std_logic; A, B : in std_logic);
end component;
component D_F_LPH0001_4
 port(L2 : out std_logic; D, E : in std_logic);
end component;

    ...
begin
 U68 : INVERT_A port map (Z => s_load, A => n265);
 U87 : NOR3_4 port map (Z => n275, A => COUNT(3),
      B => COUNT(4), C => COUNT(0));
 U88 : NOR3_4 port map (Z => n276, A => COUNTb(5),
      B => COUNT(2), C => COUNT(1));
 s_ref_ctr_out_reg : D_F_LPH0001_4 port map (L2 =>
      s_ref_ctr_out, D => s_load, E => CLK);
end SYN_refctr_rtl;
```

Verilog

```
module refctr (COUNT, CLK, RESET, REF);
  ...
 INVERT_A U68 (.Z(s_load), .A(n265) );
 NOR_4 U87 (.Z(n275), .A(COUNT[3]),
  .B(COUNT[4]), .C(COUNT[0]) );
 NOR3_4 U88 (.Z(n276), .A(COUNT[5]),
  .B(COUNT[2]),.C(COUNT[1]) );

 AND2_8 U89 (.Z(n277),.A(n275),
  .B(n276) );
 D_F_LPH0001_4 s_ref_ctr_out_reg(
  .L2(s_ref_ctr_out), .D(s_load), .E(CLK));
  ...
endmodule;
```

Visão – Netlist EDIF

EDIF

```
(cell refctr (cellType GENERIC)
   ...
(contents
   (instance U68
    (viewRef Netlist_representation
     (cellRef INVERT_A(libraryRef IBMCMOS5S_SC))
    )
)
 (instance U87
  (viewRef Netlist_representation
   (cellRef NOR3_4(libraryRef IBMCMOS5S_SC))
  )
 )

(instance U88
  (viewRef Netlist_representation
   (cellRef NOR3_4(libraryRef IBMCMOS5S_SC))
  )
)
(instance U89
  (viewRef Netlist_representation
   (cellRef AND2_8 (libraryRef IBMCMOS5S_SC))
  )
)
(instance s_ref_ctr_out_reg
  (viewRef Netlist_representation
   (cellRef D_F_LPH0001_4
    (libraryRef IBMCMOS5S_SC))
  )
)
```

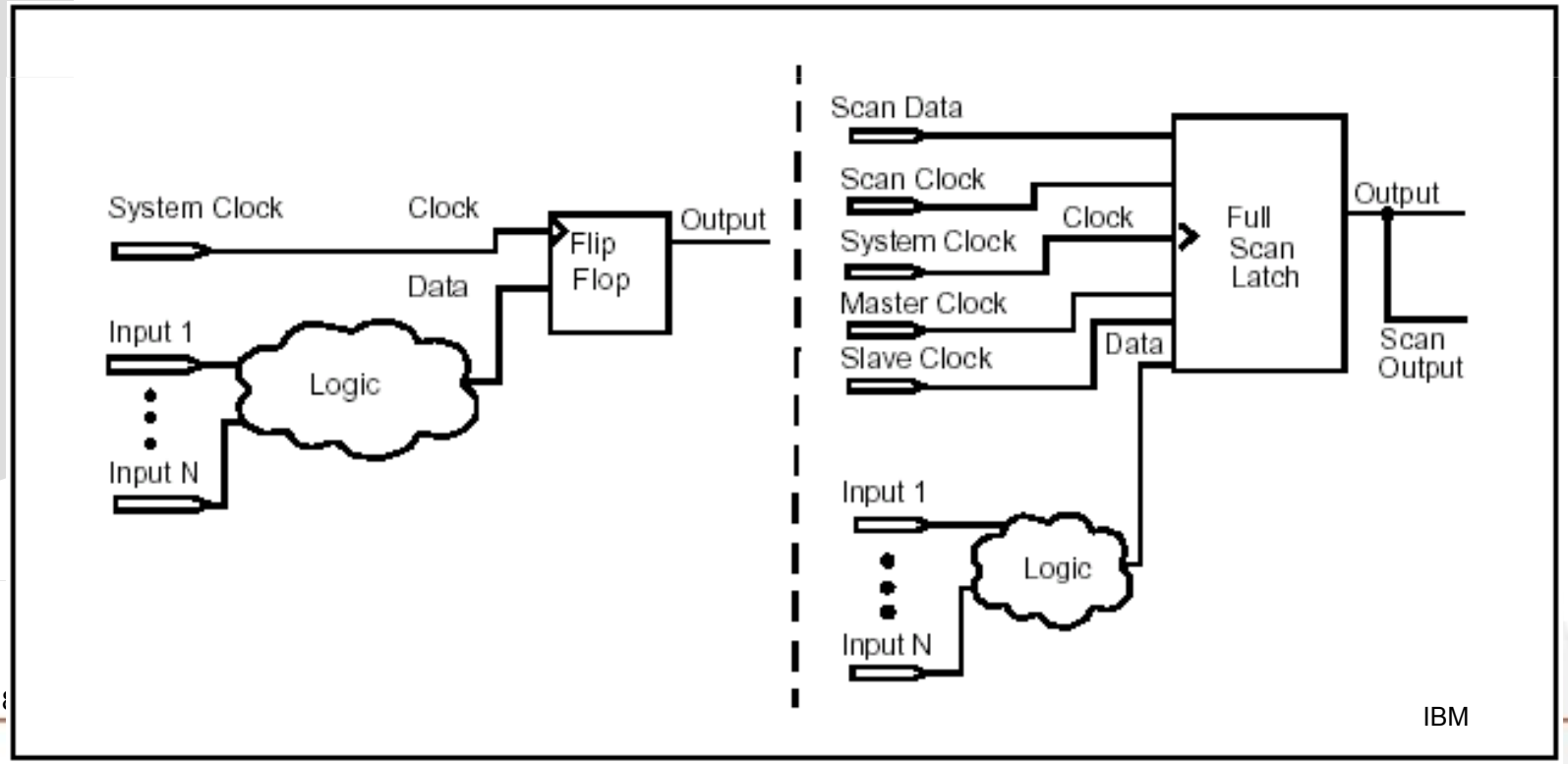
EDIF (continued)

```
(net s_load
  (joined (portRef A (instanceRef U74))
   (portRef D (instanceRef s_ref_ctr_out_reg))
   (portRef Z (instanceRef U68))
  )
)
(net CLK
  (joined (portRef CLK) (portRef E (instanceRef
    (s_ref_ctr_out_reg))
     (portRef E (instanceRef s_counter_output_.....
       ...
(net s_ref_ctr_out
   (joined (portRef D0 (instanceRef U90))
    (portRef L2 (instanceRef s_ref_ctr_out_reg))
   )
)
(net 275
   (joined (portRef A (instanceRef U89)) (portRef Z
    (instanceRef U87)))
)
(net 276
   (joined (portRef B (instanceRef U89)) (portRef Z
    (instanceRef U88)))
)
(net 277
   (joined (portRef SD instanceRef u90)) (portRef Z
    (instanceRef U89)))
   )
)))))
```

EDIF – Electronic Design Interchange Format

Inserção de teste

- DFT – Design-For-Test
  - Inserir estruturas, depois da síntese lógica, para permitir completo teste de manufatura do dispositivo.
  - Flip-Flops normais são substituídos por elementos especiais que podem ser varridos (scannable)



Planejamento de relógio interno

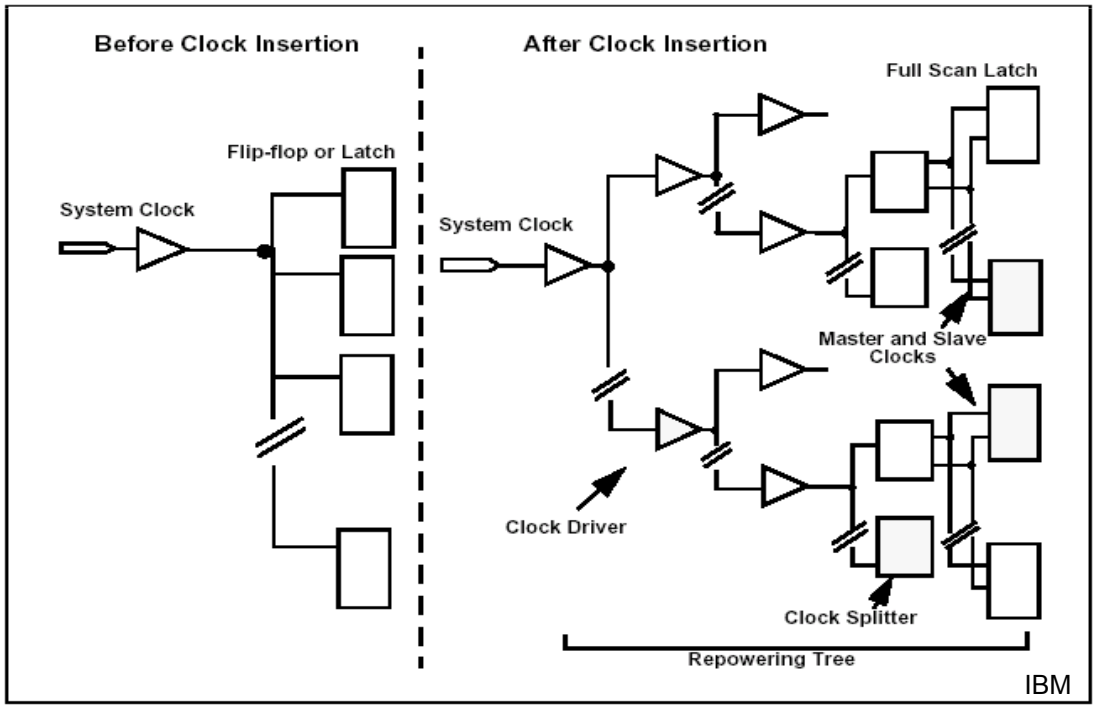
- A última fase da otimização tecnológica é a inserção da rede de relógio interno:
- Todo ASIC possui pelo menos um relógio(clock); circuitos complexos podem possuir mais que um relógio.

Parâmetros a serem considerados na rede interna de relógio:

- Área de silício;
- Atraso através da rede de relógio (clock) para circuitos que usam relógio (**latency**);
- A variação no tempo de chegada do relógio em vários elementos que usam o relógio (**skew**);
- A potência gerada pela rede de relógio como switches.

Planejamento de relógio interno

- Empresas como a IBM usam árvore para relógio (clock tree) para propagar um sinal de relógio a centenas e milhares de circuitos lógicos que recebem um sinal de relógio.
- Antes da inserção da ávore, um projeto é dito ter relógios ideais, significando que todos os circuitos lógicos estão recebendo um dado sinal de clock em paralelo de um simples driver.



Metodologia Top-Down (Fluxo de projeto ASIC)

Síntese de hardware

Projeto em HDL

Especificação do projeto

Descrição comportamental

Descrição RTL

Vetores de verificação

Verificação funcional RTL

não

sim

Síntese do projeto HDL

RTL -> Lógica

Restrições

Otimização lógica

Lógica para Tecnologia

Restrições

Otimização Timing/Área

Inserção de Scan Path & Geração de Vetor de teste

Netlist -Verificação Lógica e Timing

não

sim

Implementação do projeto

Floor Planning

Place & Route

Layout Físico

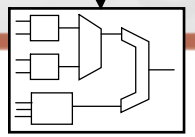
Layout Funcional e Timing

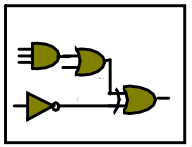
não

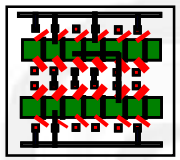
sim

Produção do Chip

```
Process(d,clk)
Begin
    if clk='1' then
      Q<=d;
    end if;
end Process;
```



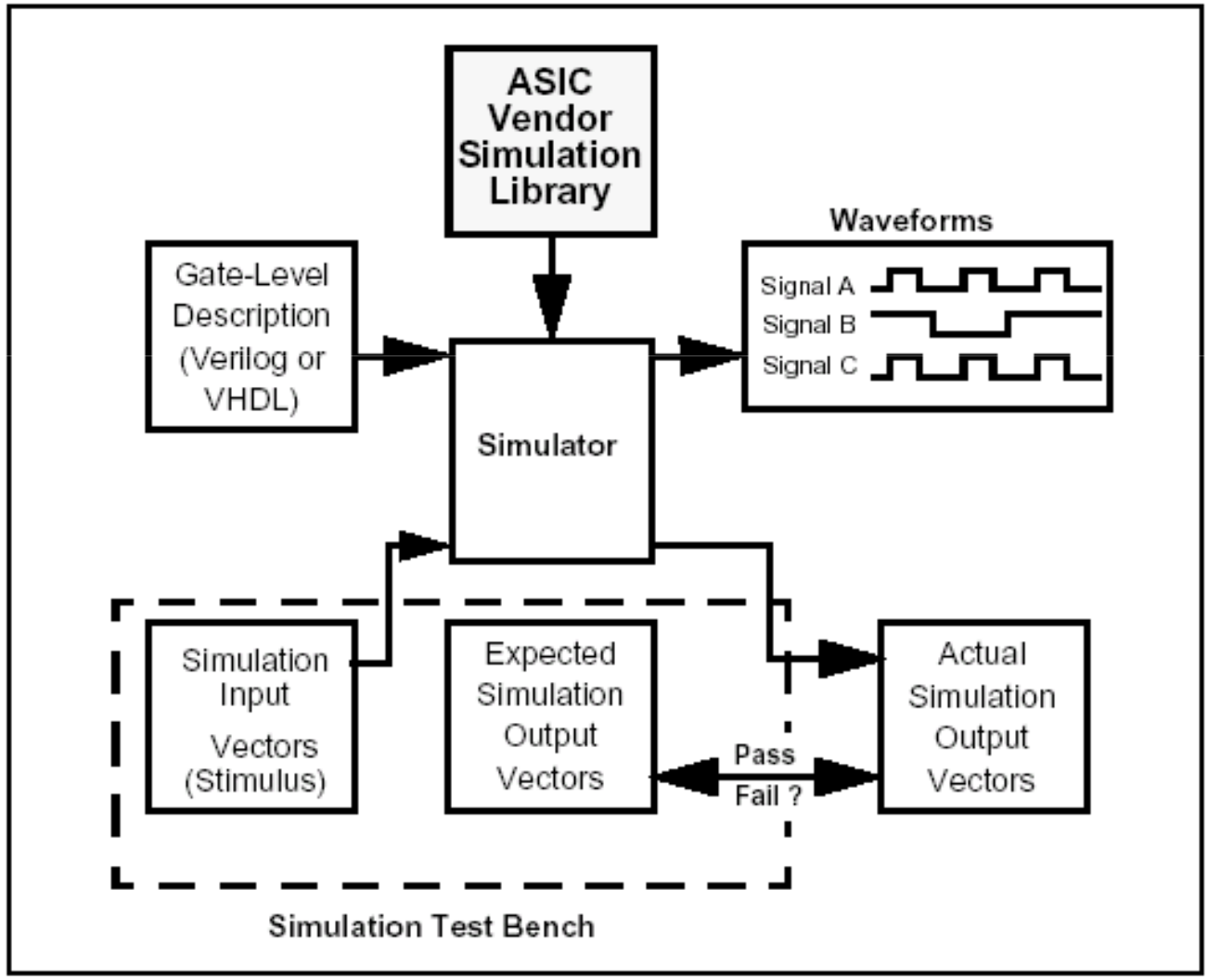




Verificação de projeto

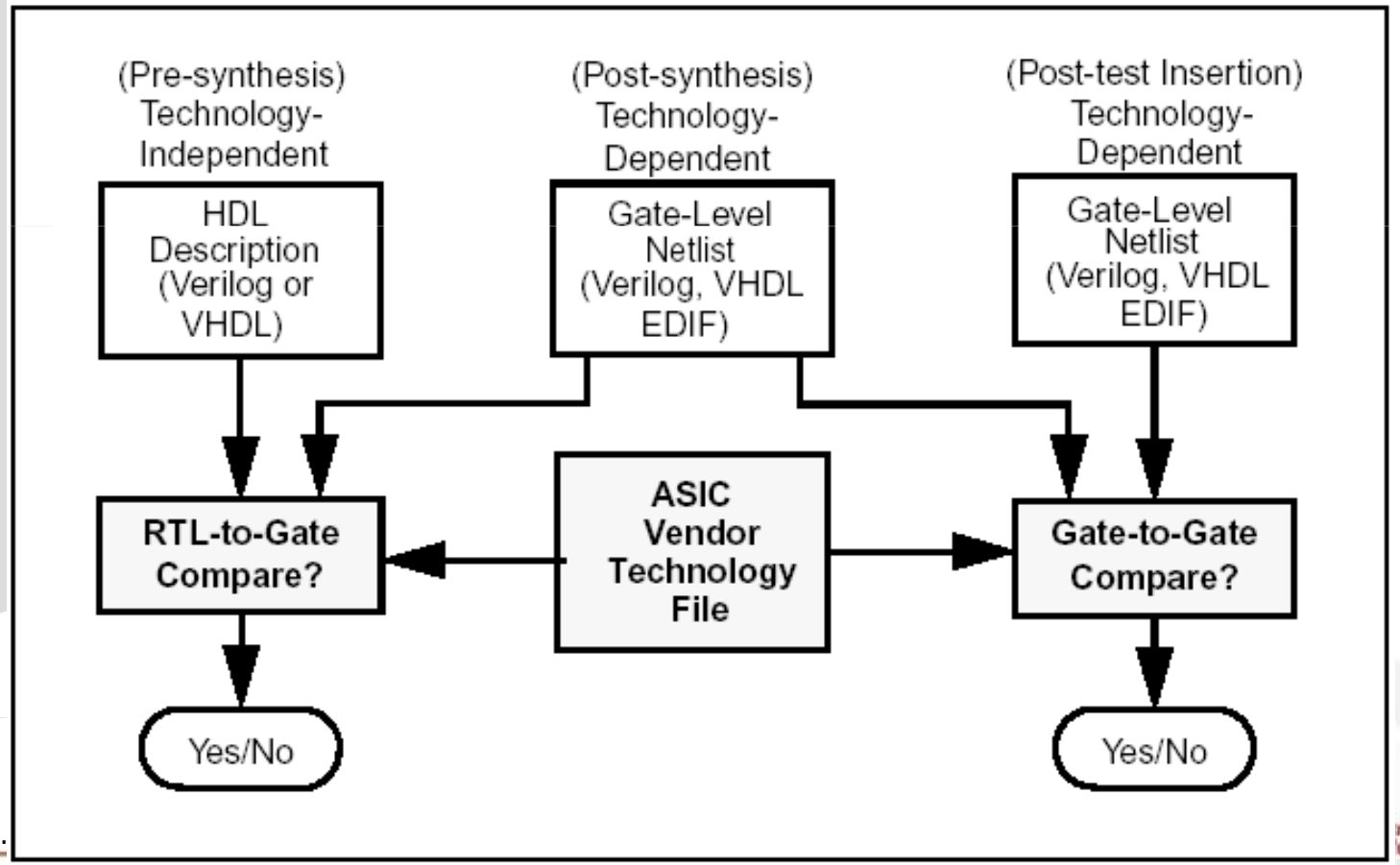
- Nesta fase o projetista tem a oportunidade de verificar se as restrições de projeto estão sendo satisfeitas após a síntese do circuito:
  - Funcionalidade do projeto está correta;
  - Restrições físicas preliminares em termos de performance, testabilidade potência, tamanho, características elétricas.
- Os circuitos podem ser verificados em vários níveis:
  - **Verificação funcional**
    - Verifica-se apenas a funcionalidade do circuito antes da síntese usando-se simulação. Em geral são gerados vetores de testes (estímulos), os quais são aplicados como vetores de entrada ao circuito em teste. Método não muito eficiente para circuitos de médio porte (100000 gates), devido ao tempo excessivo para se verificar grandes circuitos.
  - **Verificação formal**
    - Através de um modelo equivalente, é possível se chegar ao mesmo resultado de uma simulação. Neste caso, o circuito é quebrado em expressões lógicas Booleanas equivalentes e matematicamente avaliado através de comparações exaustivas. Neste método não há a necessidade de vetores de entrada ou saída. Indicado para circuitos com um grande número de gates.

Verificação Funcional
Simulação – Nível de gate (Netlist)



Verificação Formal

- Este tipo de verificação tem o mesmo propósito da simulação a nível de gates, o qual é assegurar que a funcionalidade do projeto não tenha sido modificada ou corrompida durante o processo de síntese.



Verificação de projeto

- **Verificação temporal**
  - O objetivo desta fase é determinar se o projeto, uma vez mapeado em una tecnologia específica, atende os requisitos temporais previamente estabelecidos pelo usuário.
- **Potência**
  - Verificar a dissipação de potência de operação real do circuito nas condições normais de funcionamento.
- **Verificação Pré-layout**
  - Fase utilizada para se verificar, antes da geração final do layout do ciruito, aspectos relativos a conexão interna e externa de pinos (se estão todos adequadamente ligados), pinos especiais para teste, etc.

Verificação de projeto

- **Verificação da testabilidade (pós manufatura)**
  - Esta fase permite que o projetista detecte possíveis falhas de fabricação do chip, mediante o uso de uma série de testes funcionais similares a simulação a nível de gates. Padrões são aplicados aos testadores de circuitos (manufacturing testers). Estes testadores analisam características elétricas do silício verificando se determinada parte do circuito funciona adequadamente ou não.
    1. Aqueles componentes que produziram os valores adequados (esperados) são ditos que passaram nos testes e portanto são componentes bons;
    2. aqueles que não passaram nos teste são ditos que falharam.
  - Exemplo (IBM):
    - utiliza full-scan design-for-test (DTF).
    - Fornece ao cliente um conjunto de ferramentas para geração de testbench.

Metodologia Top-Down (Fluxo de projeto ASIC)

Síntese de hardware

Projeto em HDL

Especificação do projeto

Descrição comportamental

Descrição RTL

Vetores de verificação

Verificação funcional RTL

não

sim

Síntese do projeto HDL

RTL -> Lógica

Restrições

Otimização lógica

Lógica para Tecnologia

Restrições

Otimização Timing/Área

Inserção de Scan Path & Geração de Vetor de teste

Netlist -Verificação Lógica e Timing

não

sim

Implementação do projeto

Floor Planning

Place & Route

Layout Físico

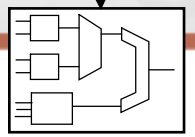
Layout Funcional e Timing

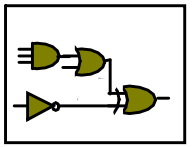
não

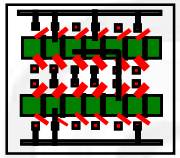
sim

Produção do Chip

```
Process(d,clk)
Begin
    if clk='1' then
      Q<=d;
    end if;
end Process;
```



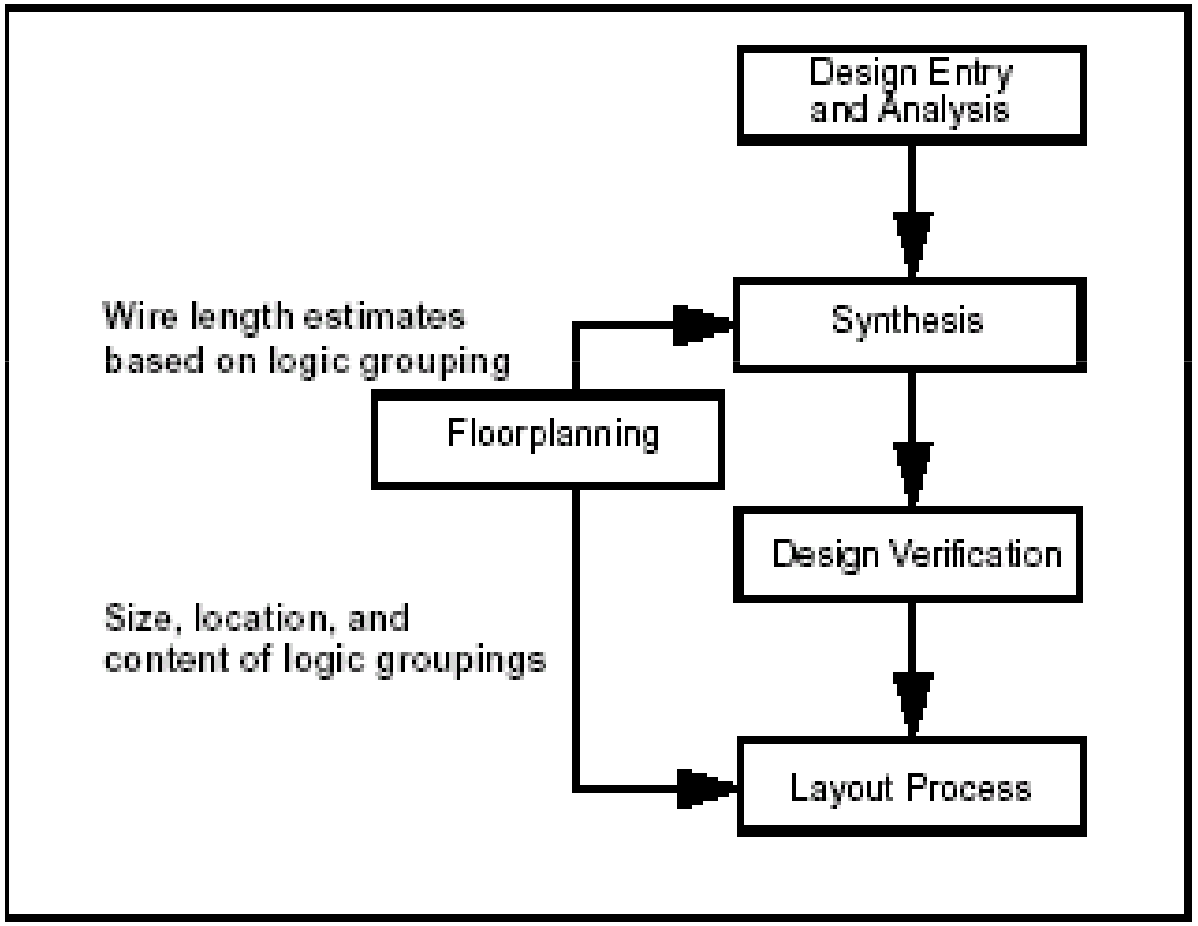




Floorplanning

- Está é a fase na qual os blocos lógicos são colocados adequadamente dentro do espaço (*loosing placement*) de sílicio reservado para a implementação do circuito (*die*) observando performance e rotabiliadadae. Nesta fase deve-se levar em conta parâmetros como:
  - Área
  - Tamanho das redes
  - Localização de pinos
  - …
- O floorplanning permite ao projetista otimizar:
  - a síntese lógica (escolha de melhores células), em função da velocidade, tamanho, etc.;
  - ter uma visão geral da malha de redes do circuito;
  - e gerar um modelo inicial otimizado para o processo de layout final do circuito.

Floorplanning



Metodologia Top-Down (Fluxo de projeto ASIC)

Síntese de hardware

Projeto em HDL

Especificação do projeto

Descrição comportamental

Descrição RTL

Vetores de verificação

Verificação funcional RTL

não

sim

Síntese do projeto HDL

RTL -> Lógica

Restrições

Otimização lógica

Lógica para Tecnologia

Restrições

Otimização Timing/Área

Inserção de Scan Path & Geração de Vetor de teste

Netlist -Verificação Lógica e Timing

não

sim

Implementação do projeto

Floor Planning

Place & Route

Layout Físico

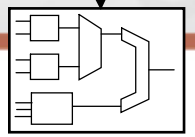
Layout Funcional e Timing

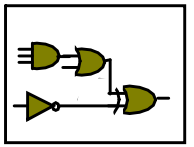
não

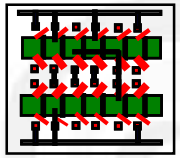
sim

Produção do Chip

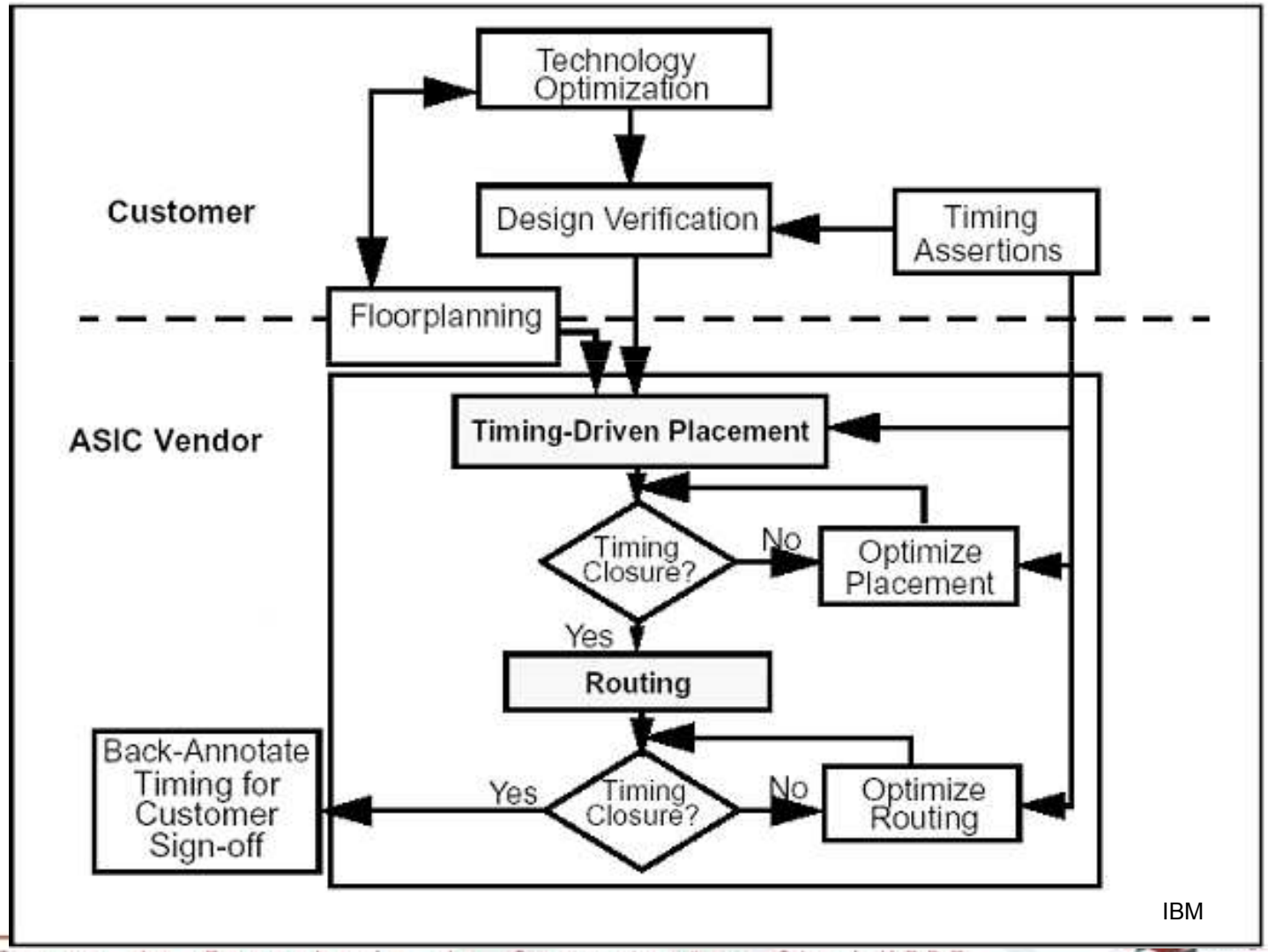
```
Process(d,clk)
Begin
    if clk='1' then
      Q<=d;
    end if;
end Process;
```





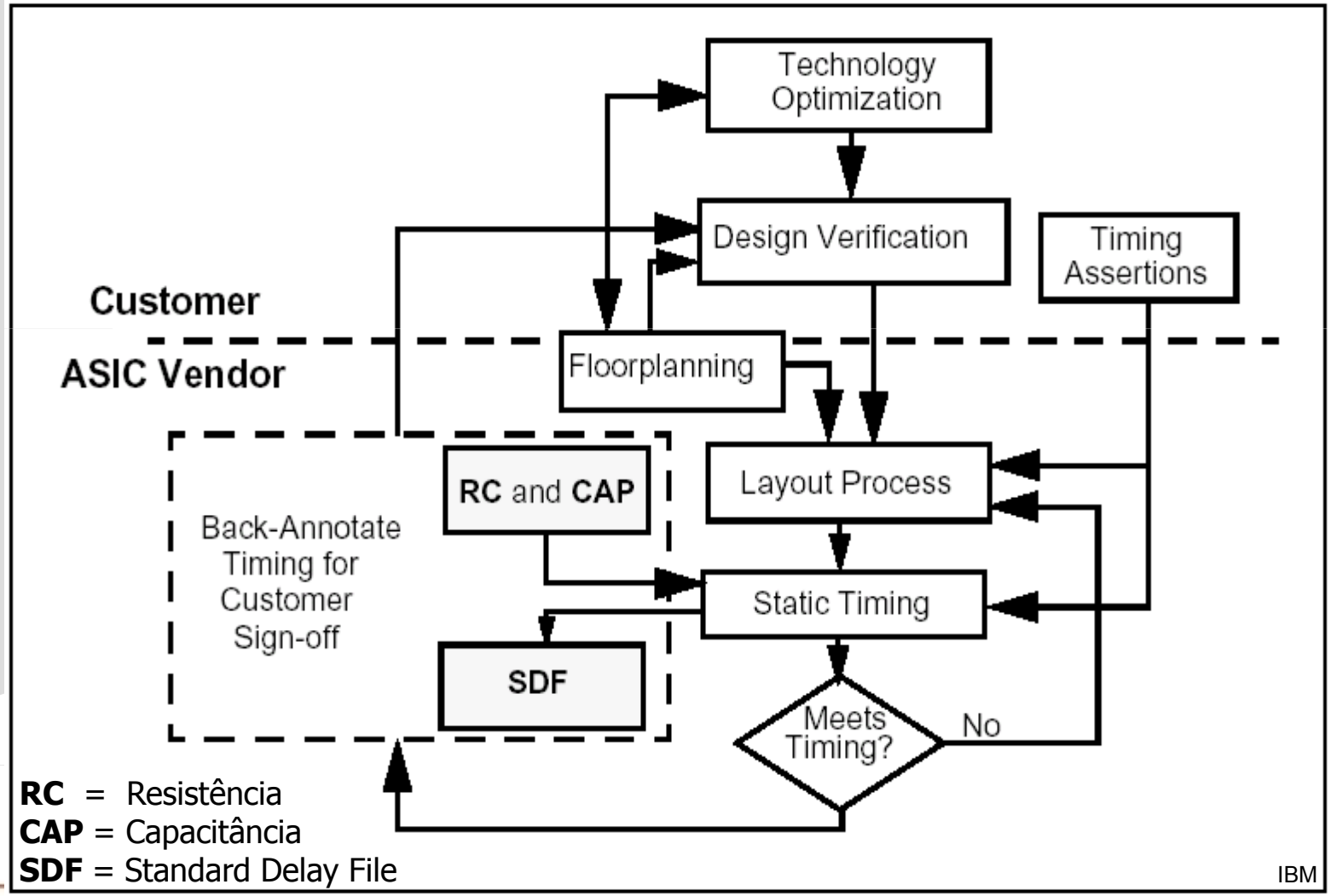


Geração do layout (IBM)



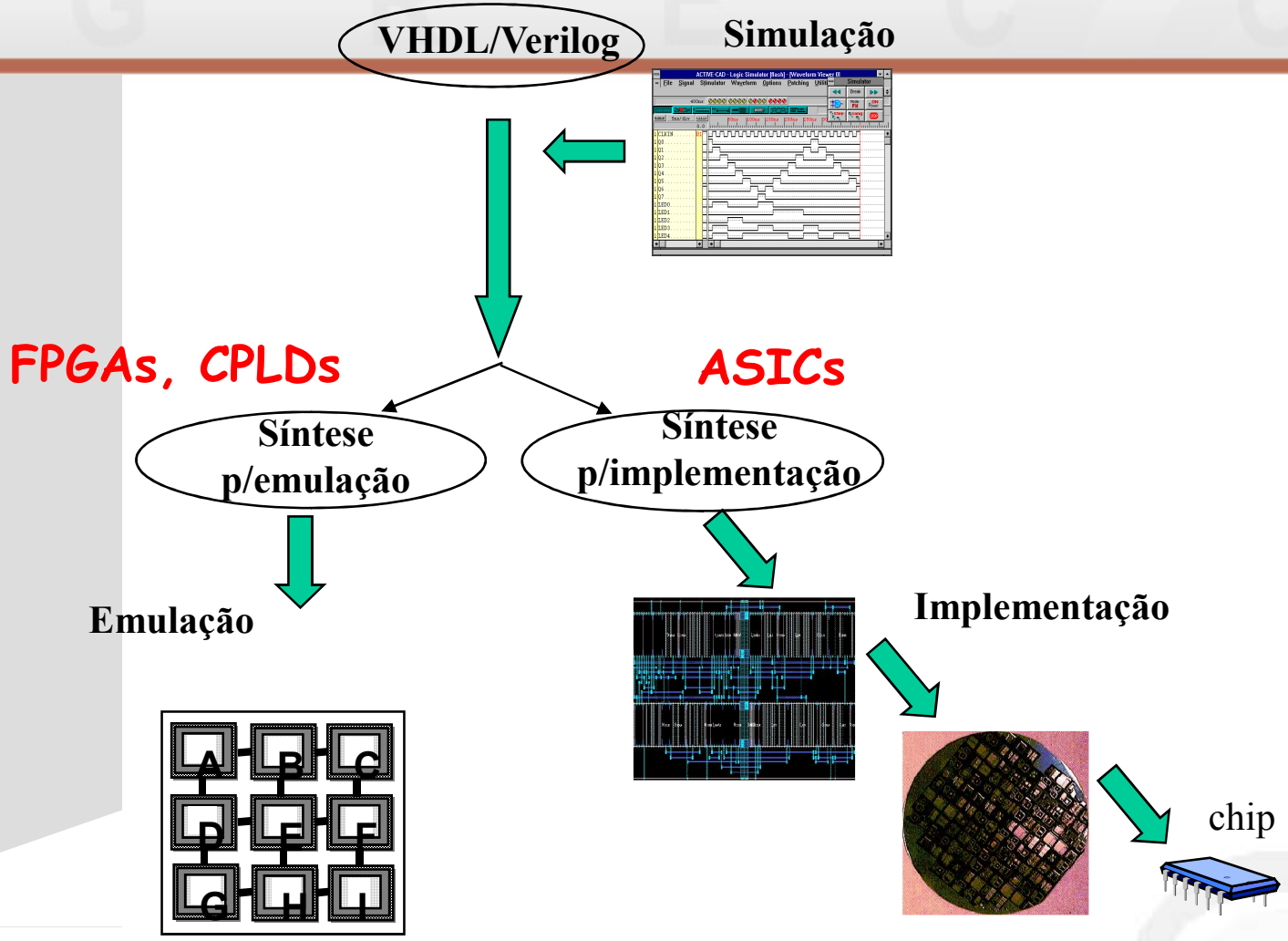
IBM

Layout – Back - Annotation



RC = Resistência
CAP = Capacitância
SDF = Standard Delay File

Fluxo de projeto Emulação/Implementação do componente de harwdare



- Mesma base de dados para emulação e implementação
- Testes durante a emulação seriam mais próximos possíveis da implementação final do projeto.

Exemplo: Máquina de vender Bomboms

- **Descrição**
  - **A máquina libera um pacote de bomboms se o cliente depositar pelo menos 15 centavos. A máquina não dá trôco.**
  - **Moedas aceitas na máquina**
    - **5 centavos (C)**
    - **10 centavos (D)**
- **Diagrama da máquina**

Diagrama de bloco

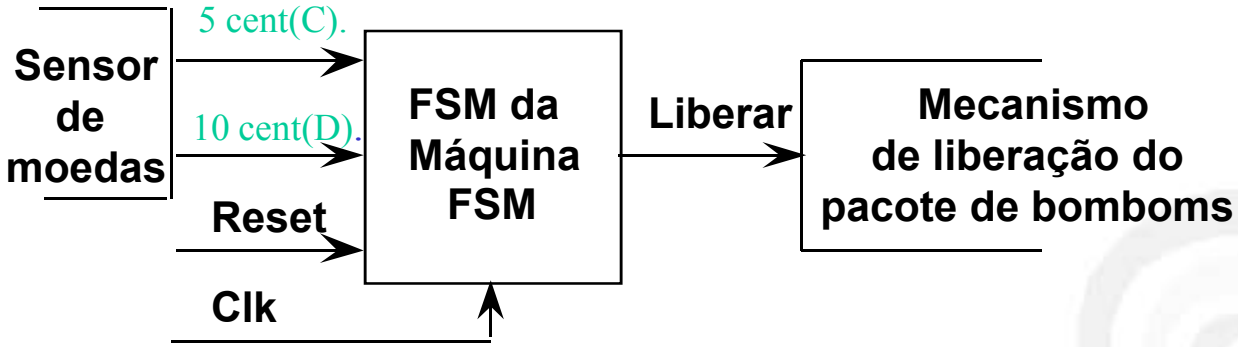
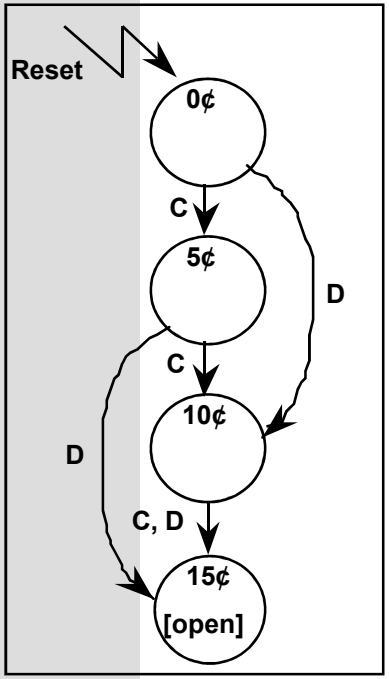


Diagrama de estados

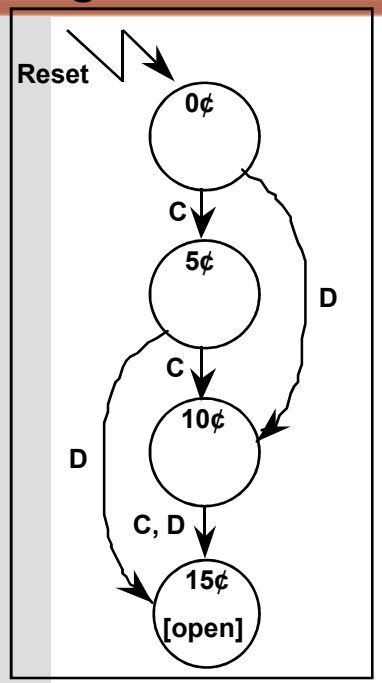


Máquina de vender bombom (código VHDL)

```
entity bombons is
      port  (
                        reset : in bit;
                        clk   : in bit;
                        c           : in bit;
                        d           : in bit;
                        open  : out bit
                  );
end bombons;

architecture arc_bombons of bombons is
      type estados is (tem_zero, tem_cinco, tem_dez, tem_quinze);

      signal estado_atual : estados;
begin
      process (reset, clk)
      begin
            if (reset = '1') then
                  estado_atual <= tem_zero;
                  open <= '0';
            elsif (clk = '1' and clk'event) then
                  case estado_atual is
                        when tem_zero =>
                              if (c = '1') then
                                    estado_atual <= tem_cinco;
                              elsif (d = '1') then
                                    estado_atual <= tem_dez;
                              end if;

                        when tem_cinco =>
                              if (c = '1') then
                                    estado_atual <= tem_dez;
                              elsif (d = '1') then
                                    estado_atual <= tem_quinze;
                                    open <= '1';
                              end if;

                        when tem_dez =>
                              if (c = '1') then
                                    estado_atual <= tem_quinze;
                                    open <= '1';
                              elsif (d = '1') then
                                    estado_atual <= tem_quinze;
                                    open <= '1';
                              end if;

                        when tem_quinze =>
                              estado_atual <= tem_zero;
                              open <= '0';

                  end case;
            end if;
      end process;
end arc_bombons;
```

Síntese

Diagrama de estados



VHDL

Tabela de estados

Present State	INPUT D	INPUT C	Next State	Output Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	X	X
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	X	X
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	X	X
15¢	X	X	15¢	1

Diagrama esquemático da FSM

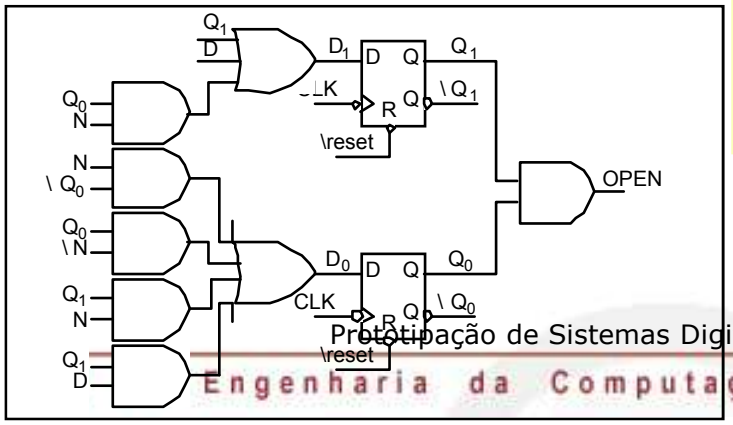
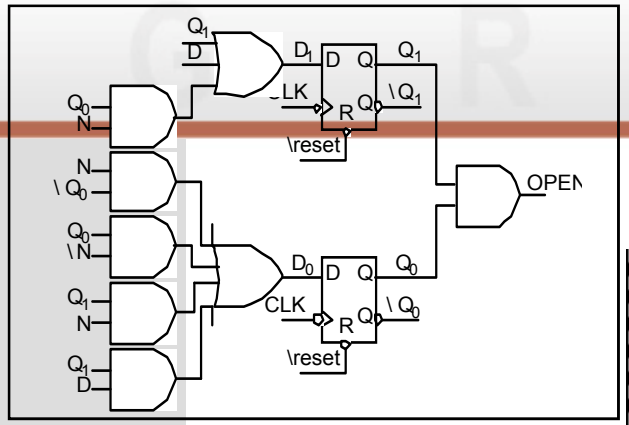
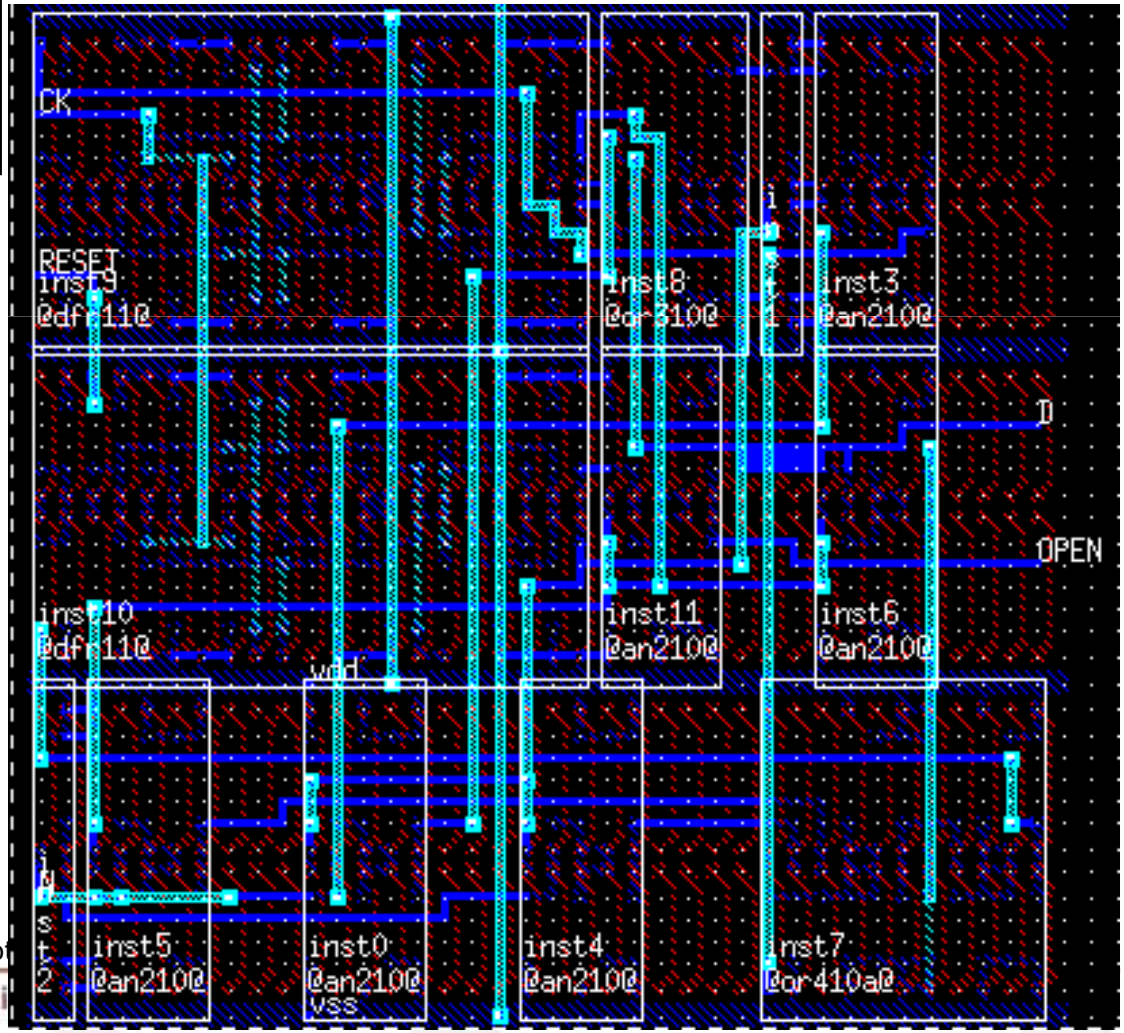


Diagrama esquemático da FSM

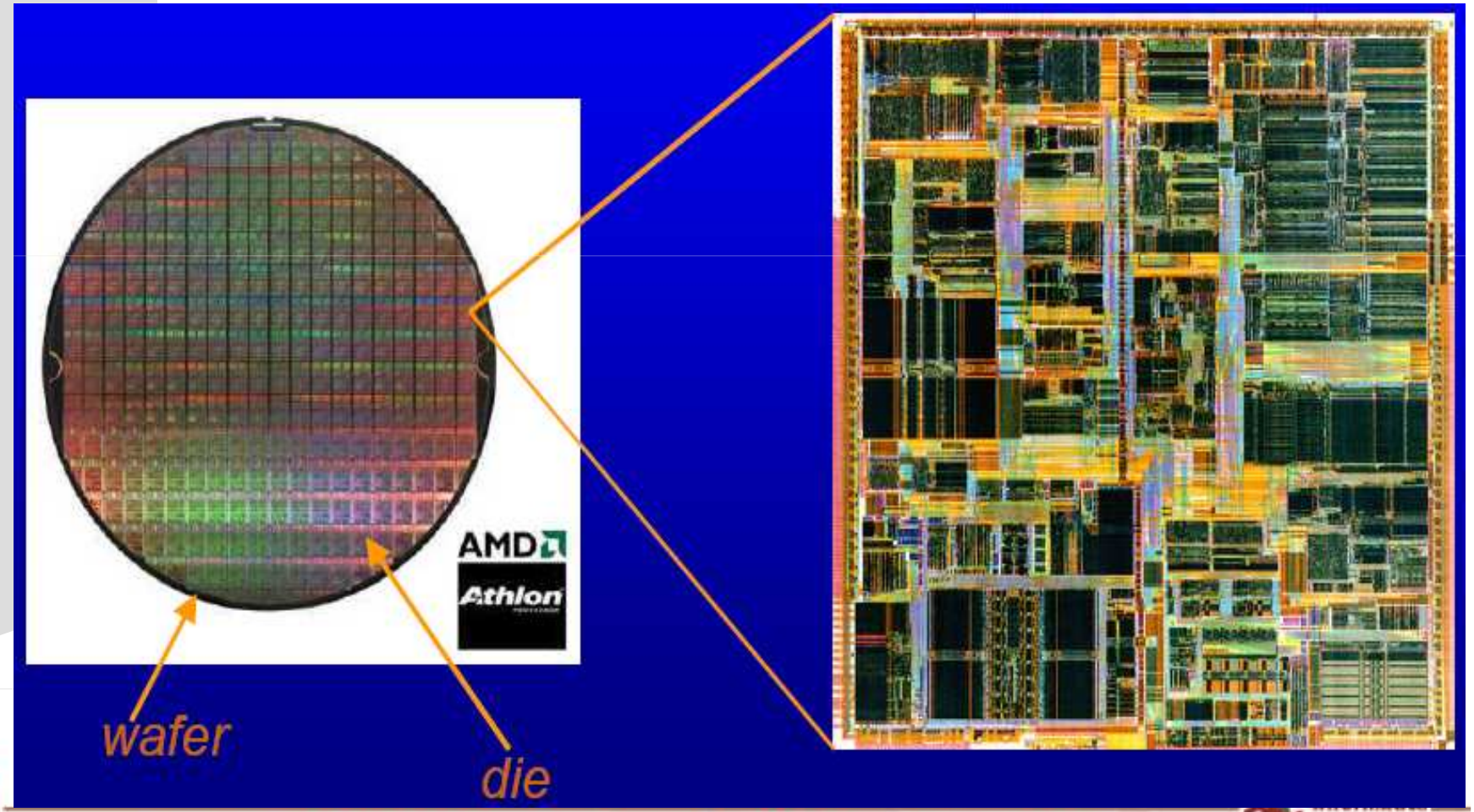




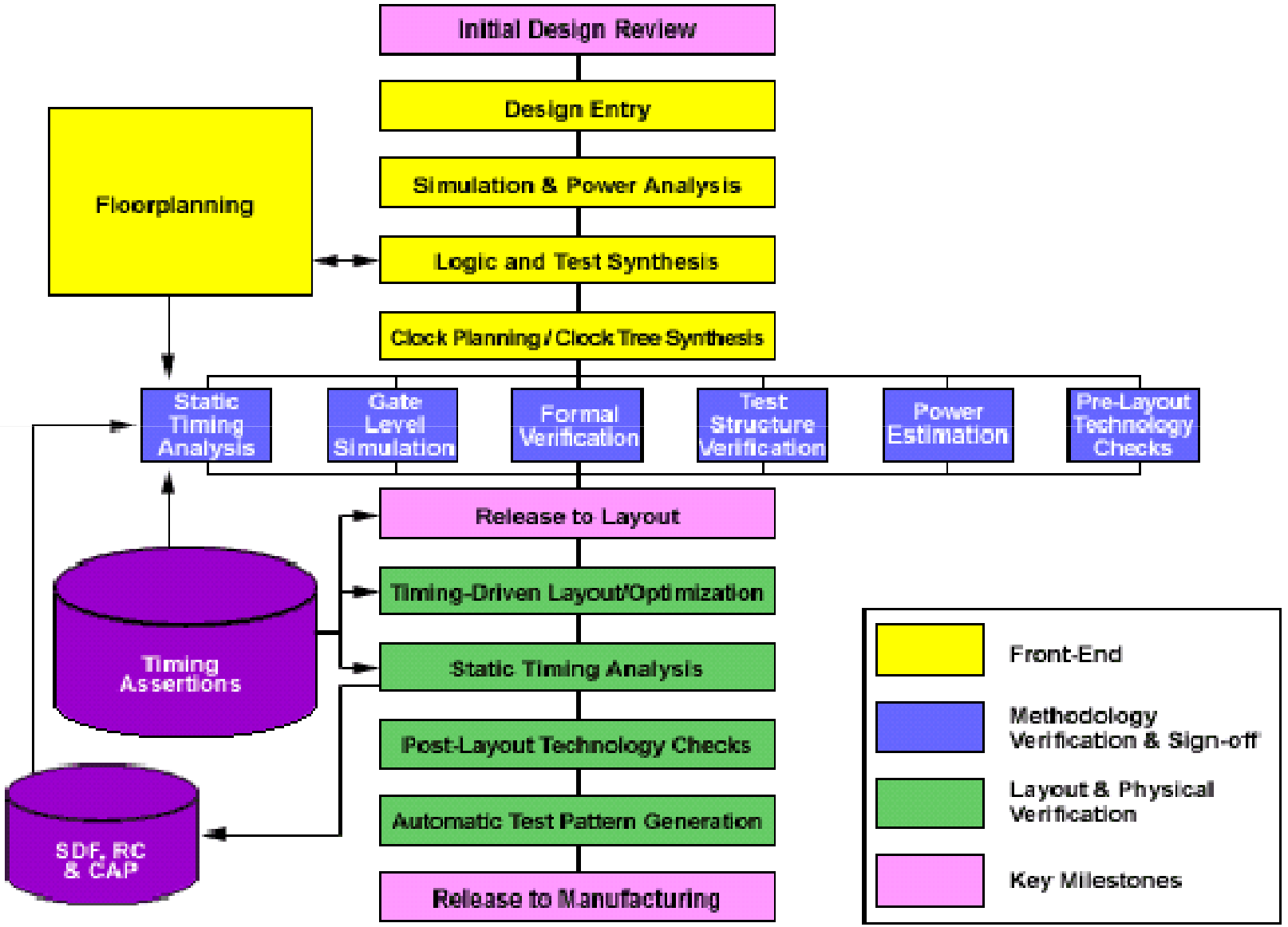
Layout

- ***Placement***
- ***Detailed Routing***

Exemplo 2 - layout

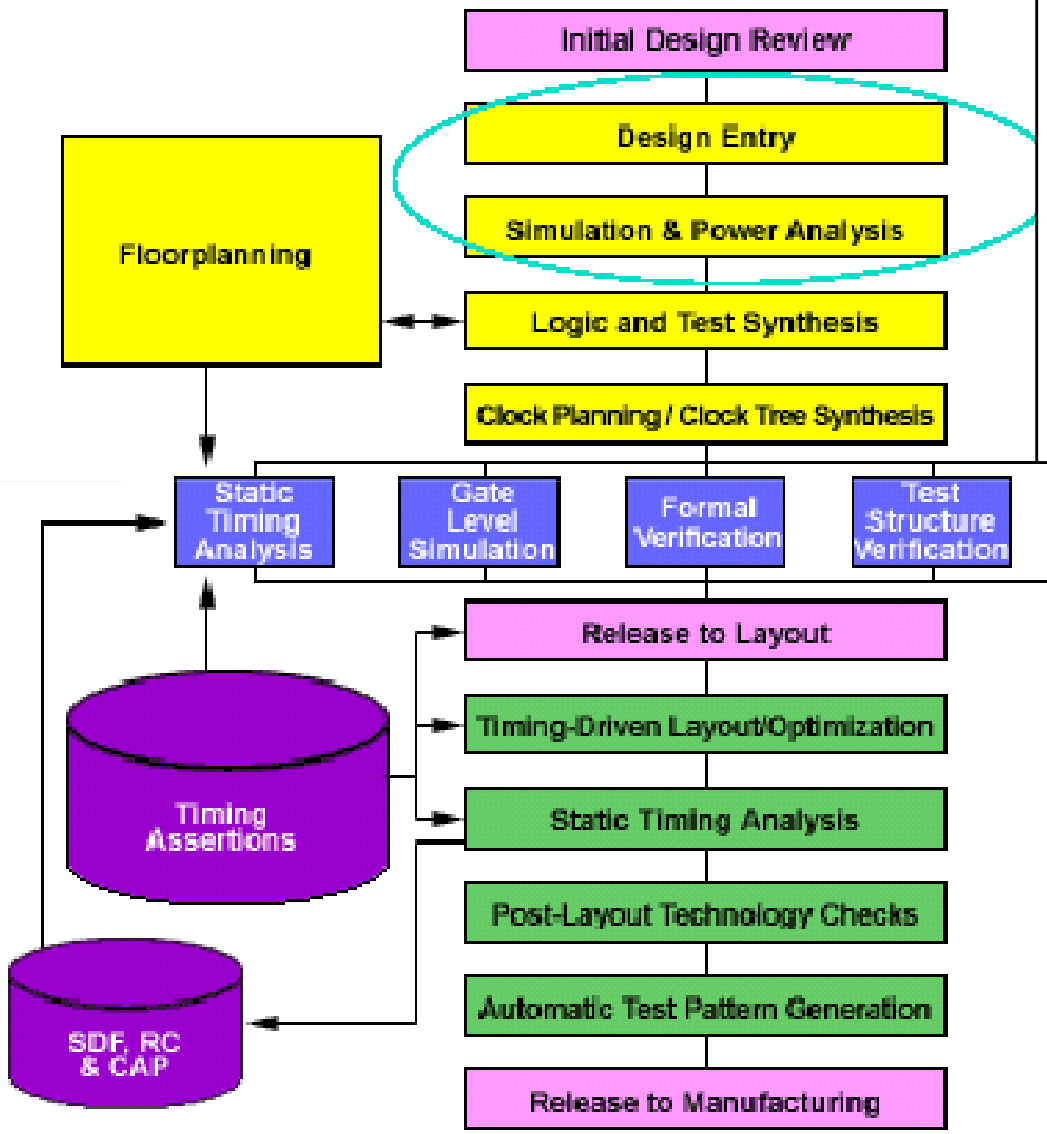


The IBM ASIC Design Flow



The IBM ASIC Design Flow

Entrada e análise

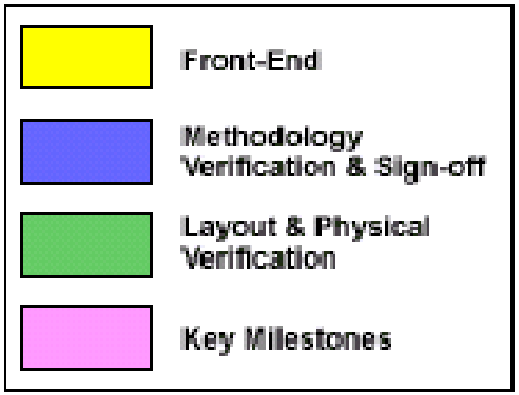


Entrada:

- Esquemática
- HDL

Análise:

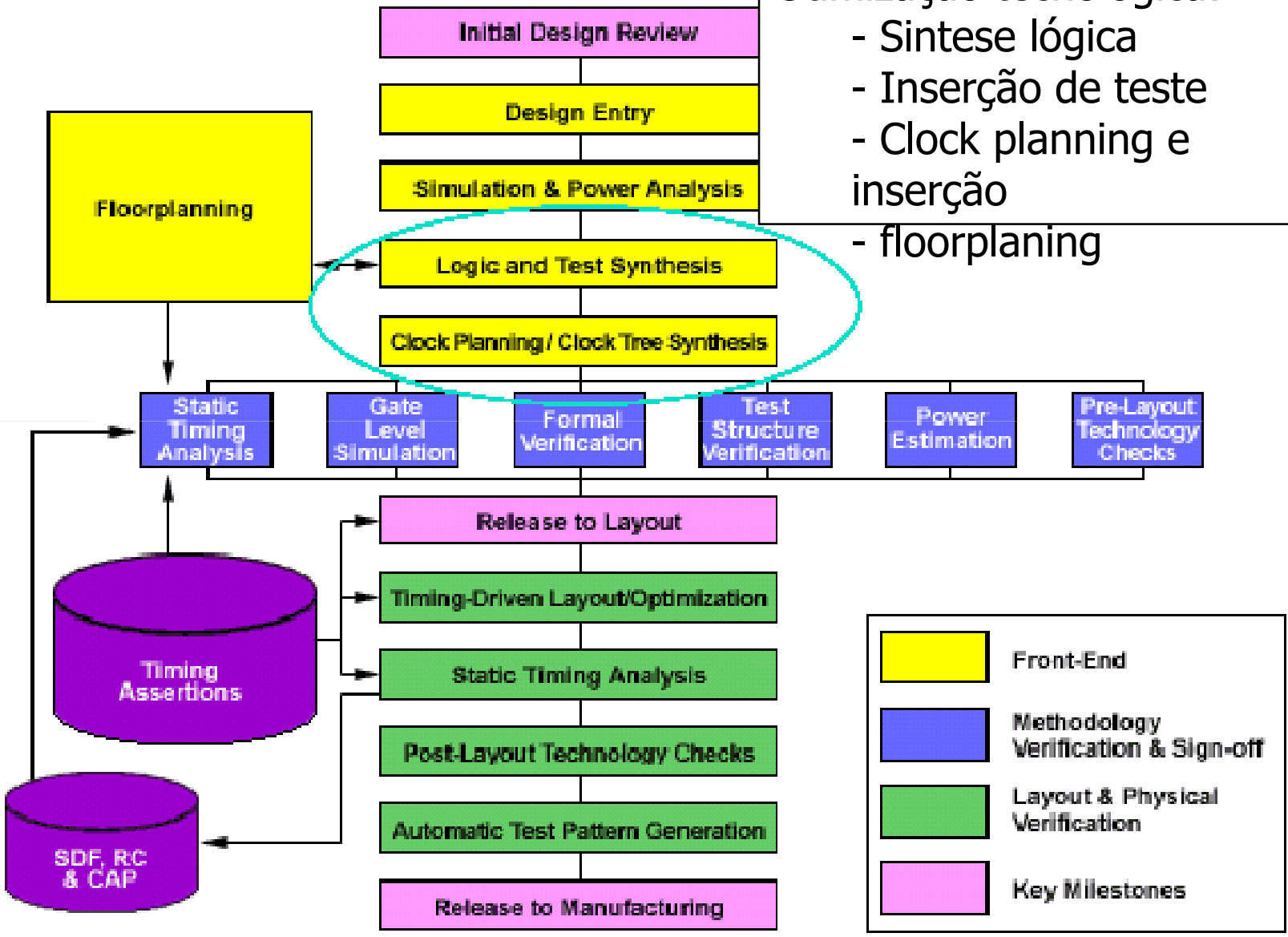
- Simulação funcional
- Testbench
- Análise de potência (estimativa)



The IBM ASIC Design Flow

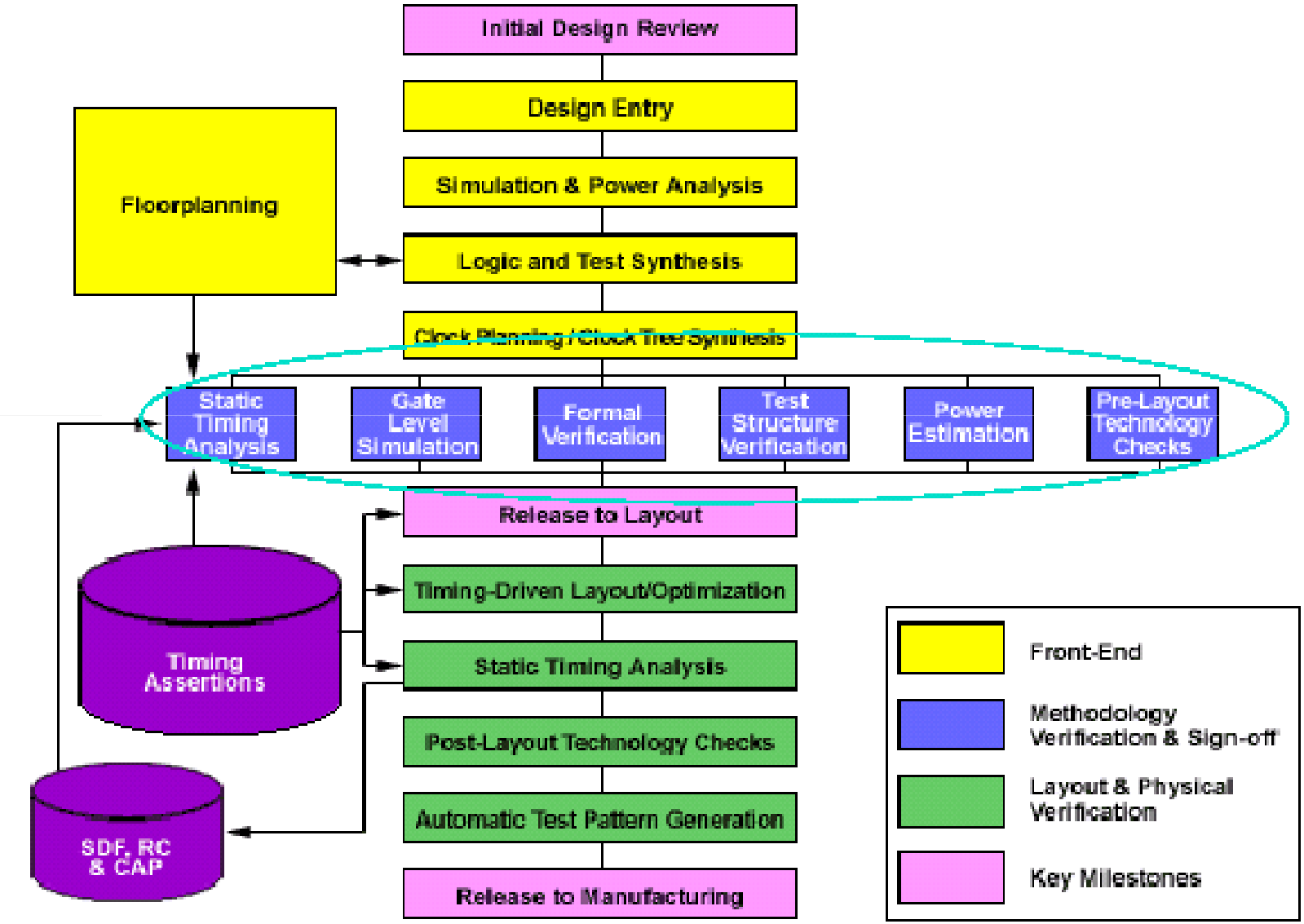
Otimização tecnológica

Otimização tecnológica:
- Sintese lógica
- Inserção de teste
- Clock planning e inserção
- floorplaning



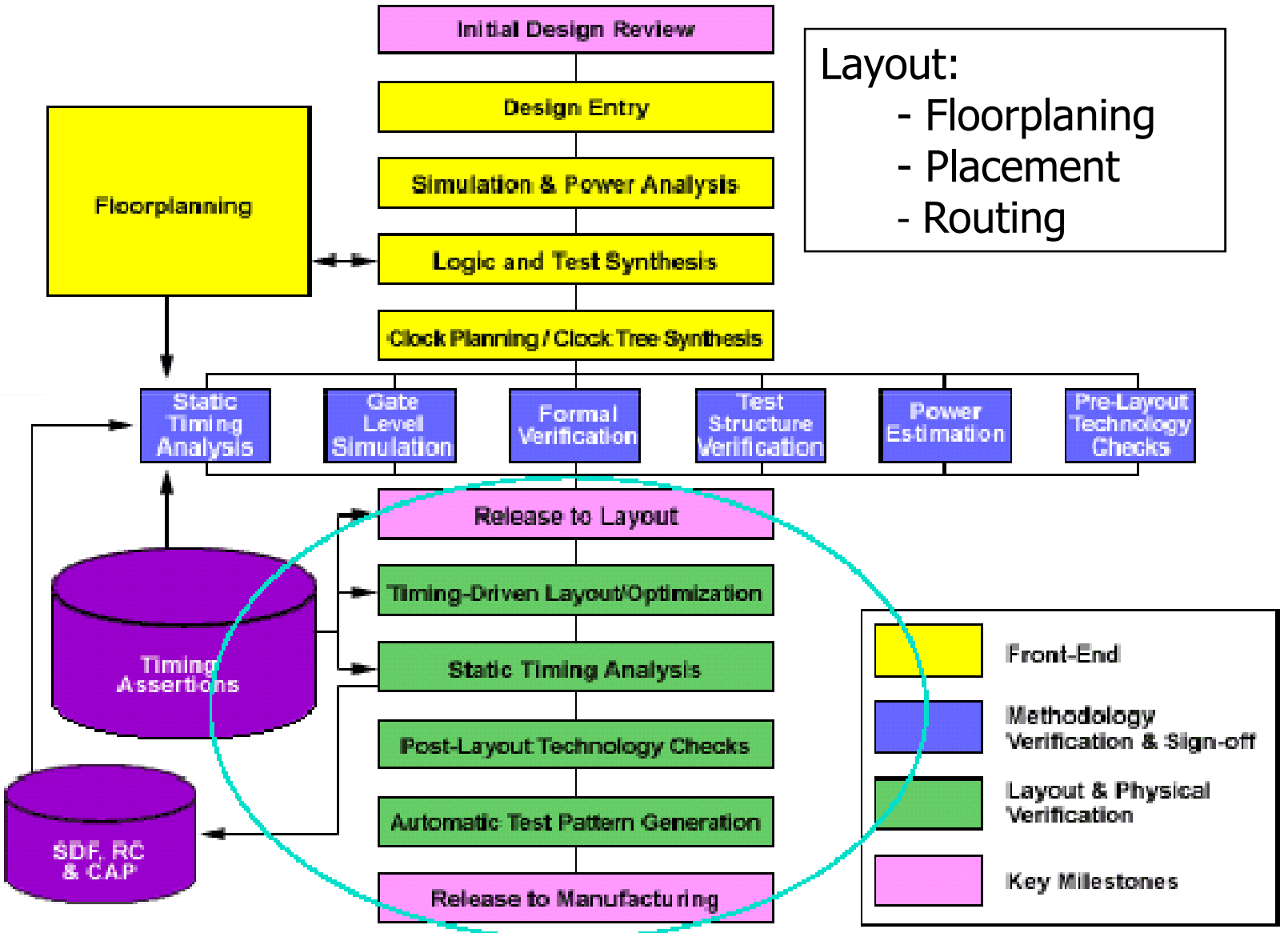
The IBM ASIC Design Flow

Verificação



The IBM ASIC Design Flow

Layout(projeto físico)



Layout:
- Floorplaning
- Placement
- Routing