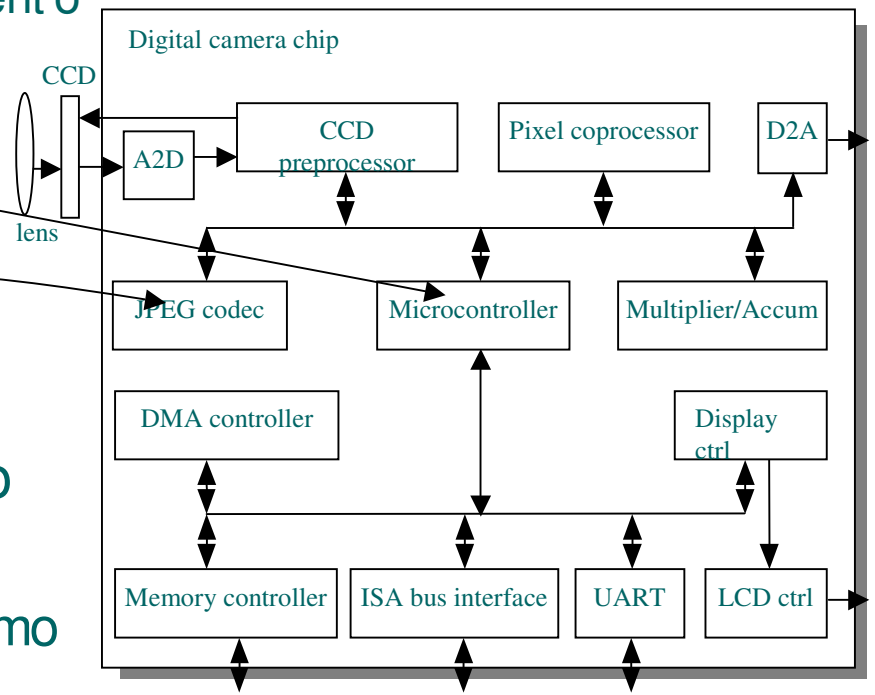
Arquitetura de Sistemas Embarcados

Edna Barros *(ensb@cin.ufpe.br)*

Centro de Informática – UFPE

Introdução

- Processador
  - Circuito digital que implementa tarefa computacional
  - Controle e unidade de processamento
  - Propósito Geral: variedade de tarefas
  - Propósito Único: uma tarefa particular
  - Propósito Único e Customizado: tarefa não padrão
- Processador de propósito único customizado:
  - Rápido, pequeno e baixo consumo
  - MAS : possui alto custo NRE, time-to-market long e apresenta pouca flexilibidade



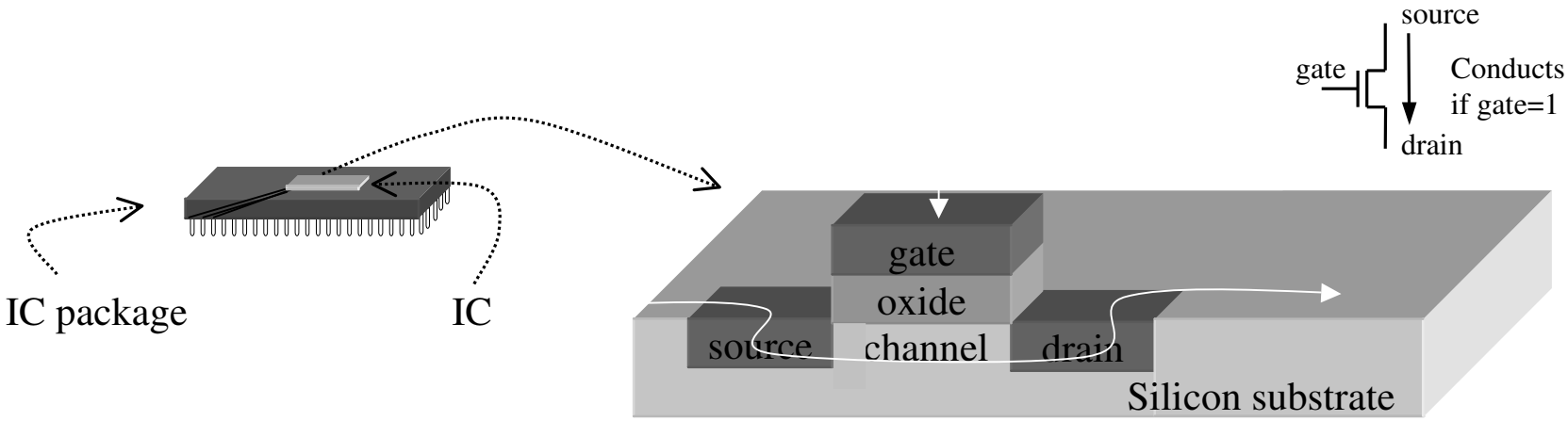
Revisão: Projetando um processador de propósito único customizado

Roteiro

- Conceitos Básicos
- Lógica Combinacional
- Lógica Sequencial
- Projetando um processador de propósito único

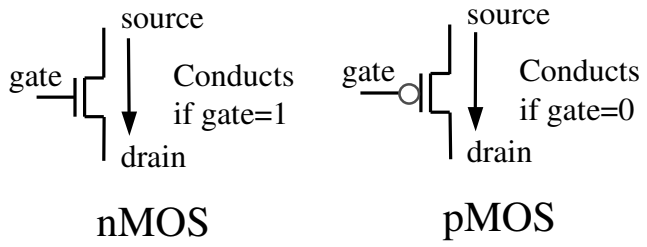
CMOS transistores em silício

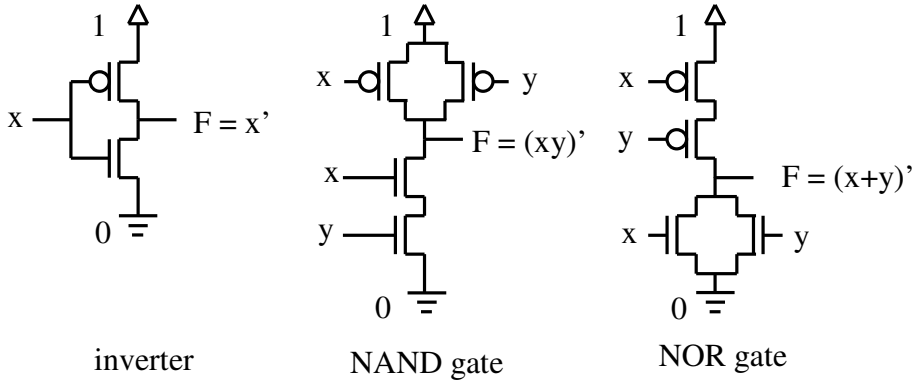
- Transistor
  - O Componente Básico dos Sistemas Digitais
  - Atua com chave
  - Tensão no "gate" controla fluxo de corrente da fonte para o "drain"



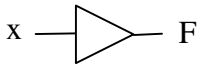
Implementações de Transistores CMOS

- Complementary Metal Oxide Semiconductor
- Níveis Lógicos
  - 0 é 0V, 1 é 5V
- Dois tipos básicos
  - nMOS conduz se gate=1
  - pMOS conduz se gate=0
- Portas Básicas
  - Inverter, NAND, NOR



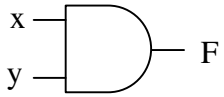


Portas Lógicas Básicas



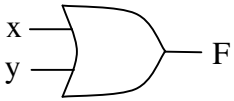
x	F
0	0
1	1

F = x
Driver



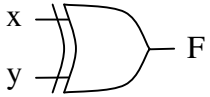
x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

F = x y
AND



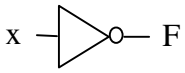
x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

F = x + y
OR



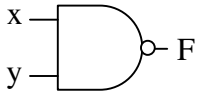
x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

F = x ⊕ y
XOR



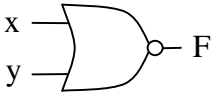
x	F
0	1
1	0

F = x'
Inverter



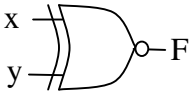
x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

F = (x y)'
NAND



x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

F = (x+y)'
NOR



x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

F = x ⊙ y
XNOR

Projeto de Circuitos Combinacionais

A) Problem description

y is 1 if a is to 1, or b and c are 1. z is 1 if b or c is to 1, but not both, or if all are 1.

B) Truth table

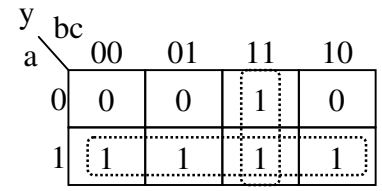
Inputs			Outputs	
a	b	c	y	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

C) Output equations

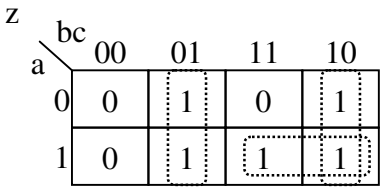
y = a'bc + ab'c' + ab'c + abc' + abc

z = a'b'c + a'bc' + ab'c + abc' + abc

D) Minimized output equations

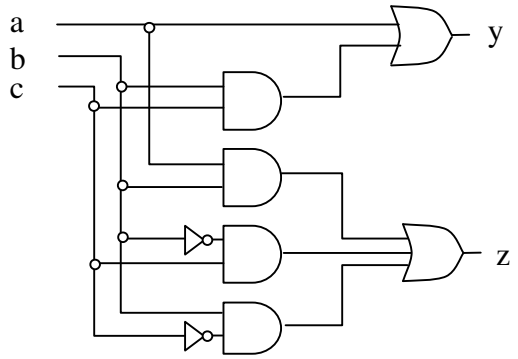


y = a + bc

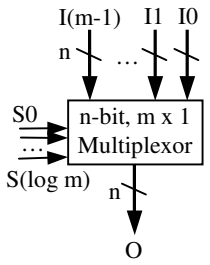
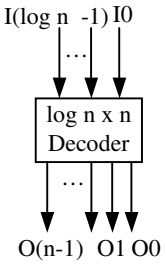
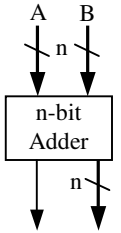
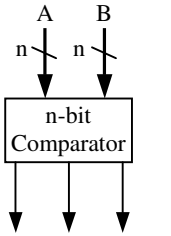
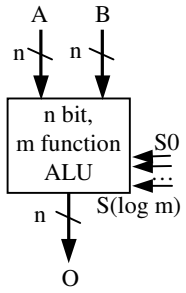


z = ab + b'c + bc'

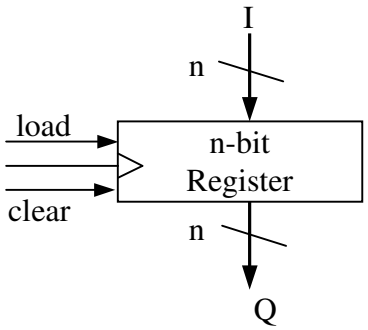
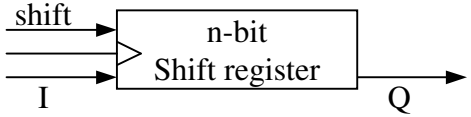
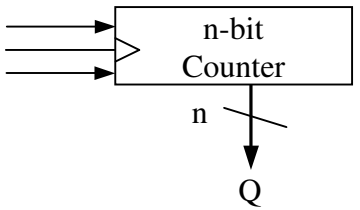
E) Logic Gates



Circuitos Combinacionais

				
O = I0 if S=0..00 I1 if S=0..01 … I(m-1) if S=1..11	O0 =1 if I=0..00 O1 =1 if I=0..01 … O(n-1) =1 if I=1..11	sum = A+B (first n bits) carry = (n+1)'th bit of A+B	less = 1 if A<B equal =1 if A=B greater=1 if A>B	O = A *op* B *op* determined by S.
	With enable input e → all O's are 0 if e=0	With carry-in input Ci→ sum = A + B + Ci		May have status outputs carry, zero, etc.

Circuitos Sequenciais

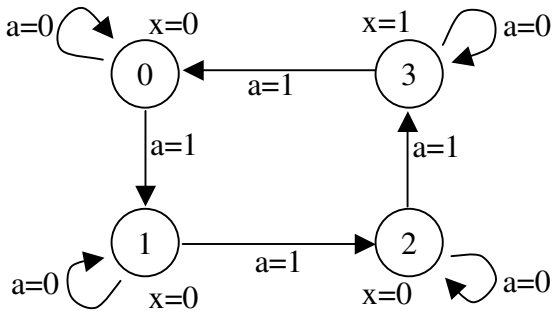
		
Q = 0 if clear=1, I if load=1 and clock=1, Q(previous) otherwise.	Q = lsb - Content shifted - I stored in msb	Q = 0 if clear=1, Q(prev)+1 if count=1 and clock=1.

Projeto de Circuitos Sequenciais

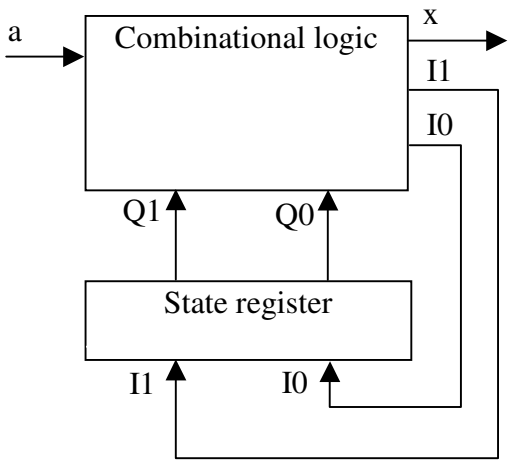
A) Problem Description

You want to construct a clock divider. Slow down your pre-existing clock so that you output a 1 for every four clock cycles

B) State Diagram



C) Implementation Model



D) State Table (Moore-type)

Inputs			Outputs		
Q1	Q0	a	I1	I0	x
0	0	0	0	0	0
0	0	1	0	1	
0	1	0	0	1	0
0	1	1	1	0	
1	0	0	1	0	0
1	0	1	1	1	
1	1	0	1	1	1
1	1	1	0	0	

Projeto de Circuitos Sequenciais

D) State Table (Moore-type)

Inputs			Outputs		
Q1	Q0	a	I1	I0	x
0	0	0	0	0	0
0	0	1	0	1	
0	1	0	0	1	0
0	1	1	1	0	
1	0	0	1	0	0
1	0	1	1	1	
1	1	0	1	1	1
1	1	1	0	0	

E) Minimized Output Equations

I1 Q1Q0

a \ Q1Q0	00	01	11	10
0	0	0	1	1
1	0	1	0	1

I1 = Q1'Q0a + Q1a' + Q1Q0'

I0 Q1Q0

a \ Q1Q0	00	01	11	10
0	0	1	1	0
1	1	0	0	1

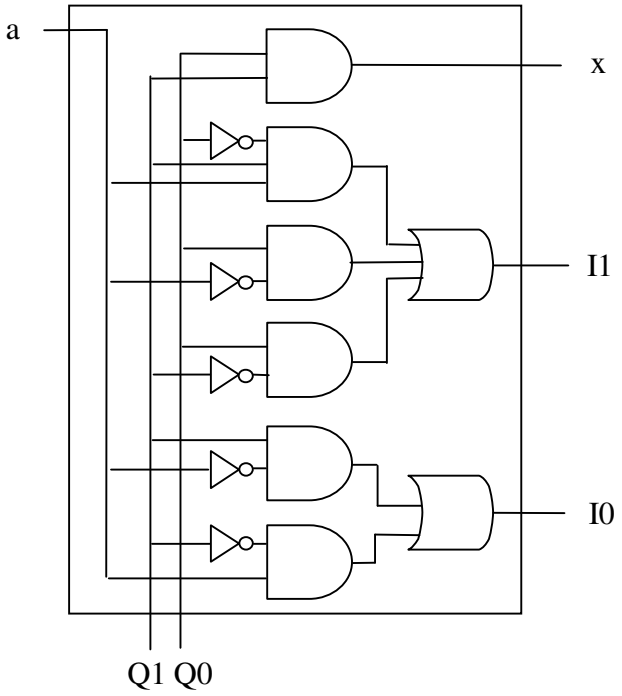
I0 = Q0a' + Q0'a

x Q1Q0

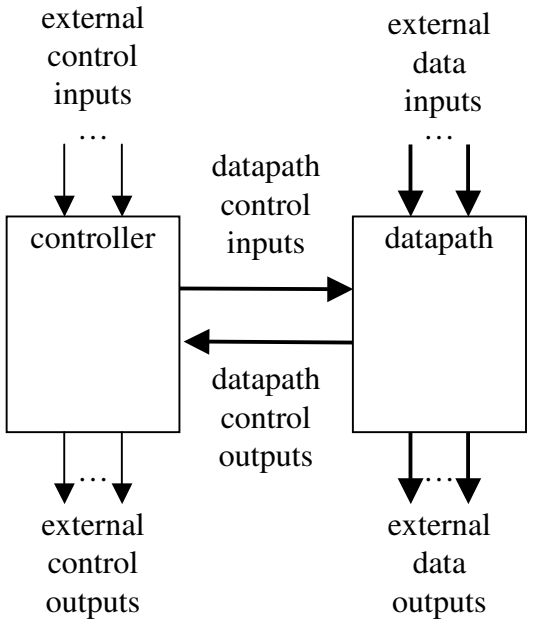
a \ Q1Q0	00	01	11	10
0	0	0	1	0
1	0	0	1	0

x = Q1Q0

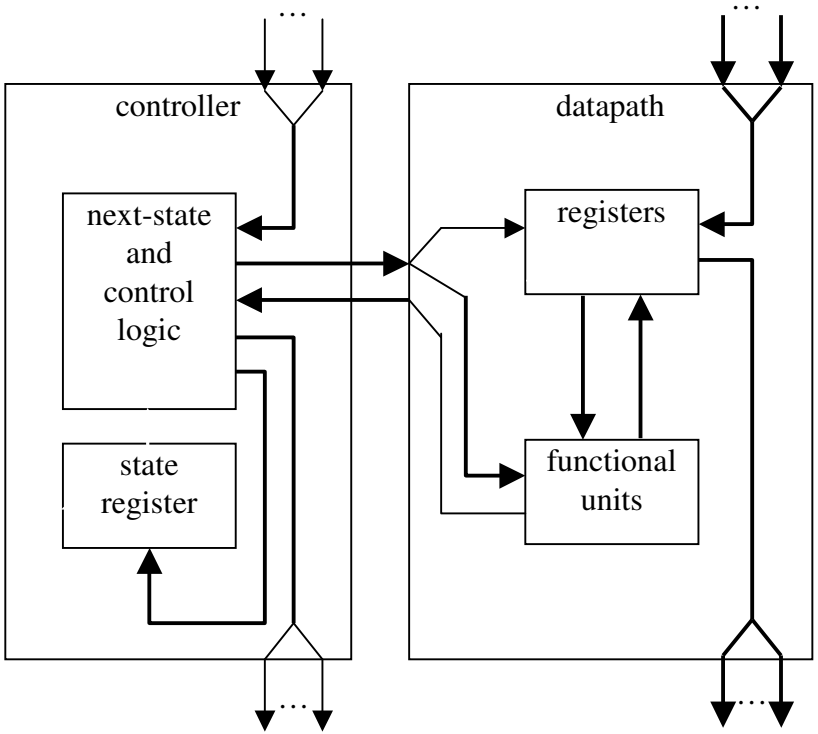
F) Combinational Logic



Modelo Básico de um Processador de Propósito Único



controller and datapath

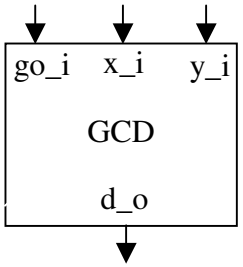


a view inside the controller and datapath

Exemplo: greatest common divisor

- Descreva o algoritmo
- Converta o algoritmo para uma FSM complexa
  - FSMD: finite-state machine with datapath
  - Templates podem ser usados

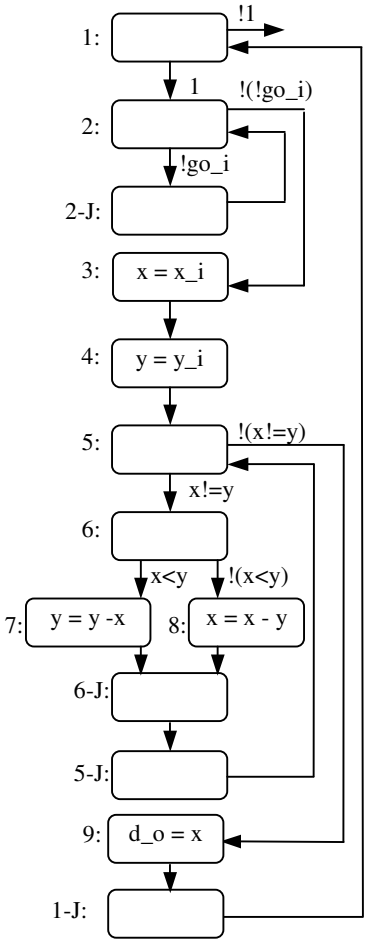
(a) black-box view



(b) desired functionality

```
0: int  x, y;
1: while (1) {
2:   while (!go_i);
3:   x = x_i;
4:   y = y_i;
5:   while  (x != y)  {
6:      if  (x < y)
7:         y = y - x;
       else
8:         x = x - y;
     }
9:   d_o = x;
   }
```

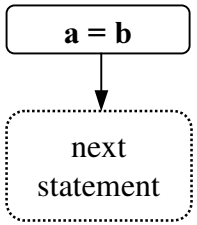
(c) state diagram



Templates de Diagramas de Estado

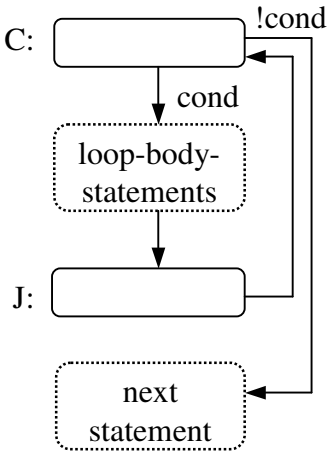
Assignment statement

a = b
next statement



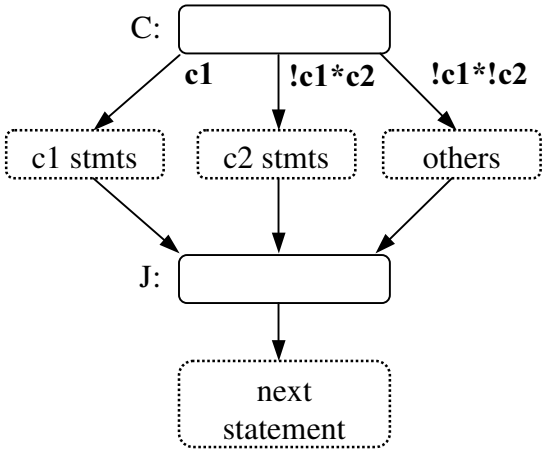
Loop statement

while (cond) {
    loop-body-
        statements
}
next statement



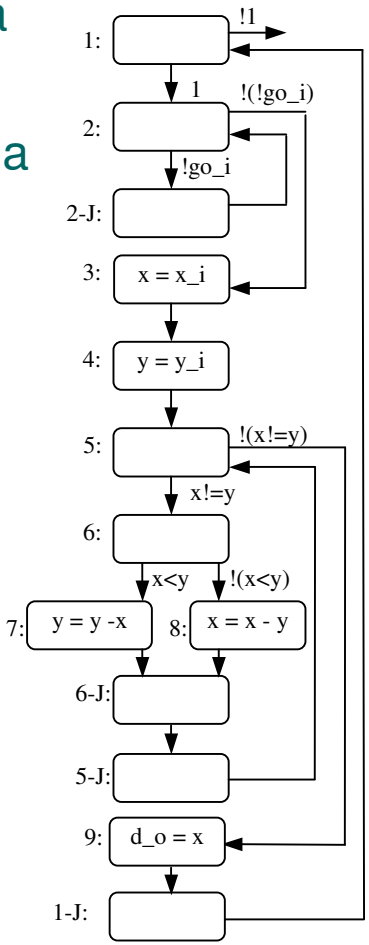
Branch statement

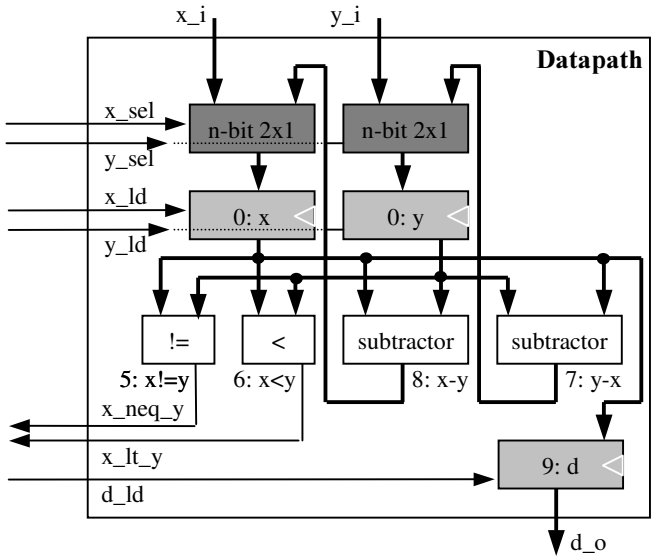
if (c1)
    c1 stmts
else if c2
    c2 stmts
else
    other stmts
next statement



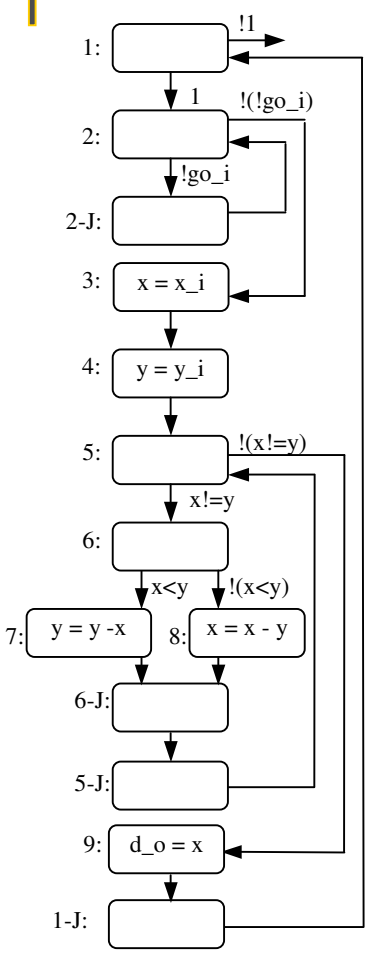
Criando a Unidade de Processamento

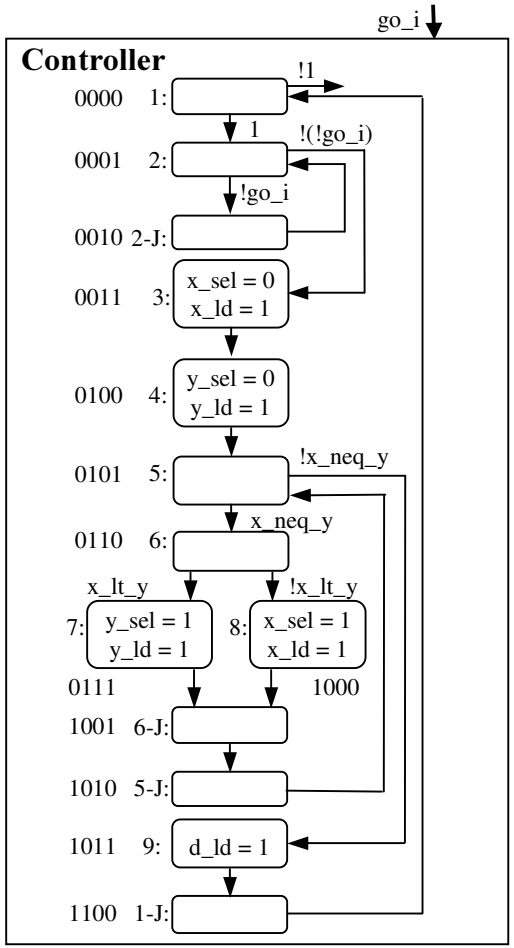
- Crie um registrador para cada variável declarada
- Crie uma unidade funcional para cada operação aritmética
- Faça a conexão dos registradores às unidades funcionais
  - Baseado nas leituras e escritas
  - Use multiplexadores para fontes múltiplas
- Crie um identificador único para cada saída e entrada da unidade de processamento



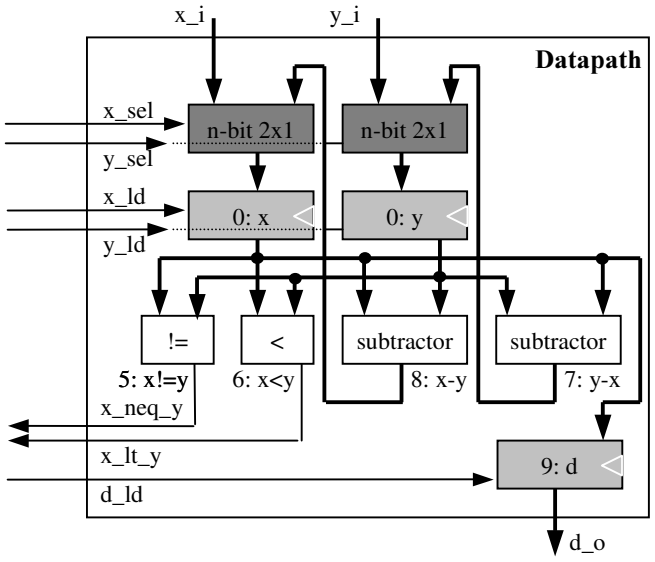


Criando um controle baseado em FSM





- Mesma estrutura que uma FSMD
- Substitua ações complexas por configurações da unid. de processamento



Conectando controle e unidade de processamento

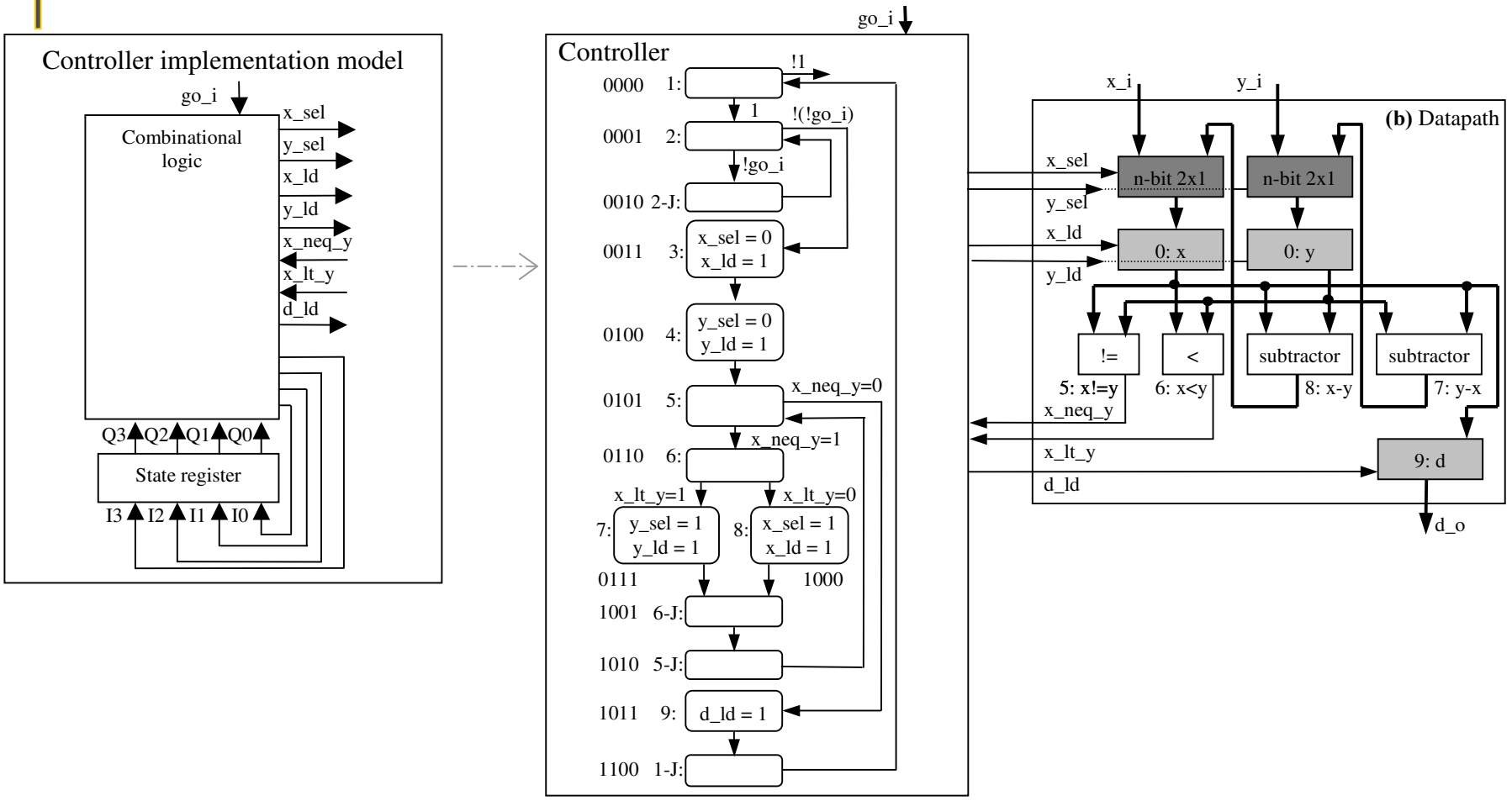
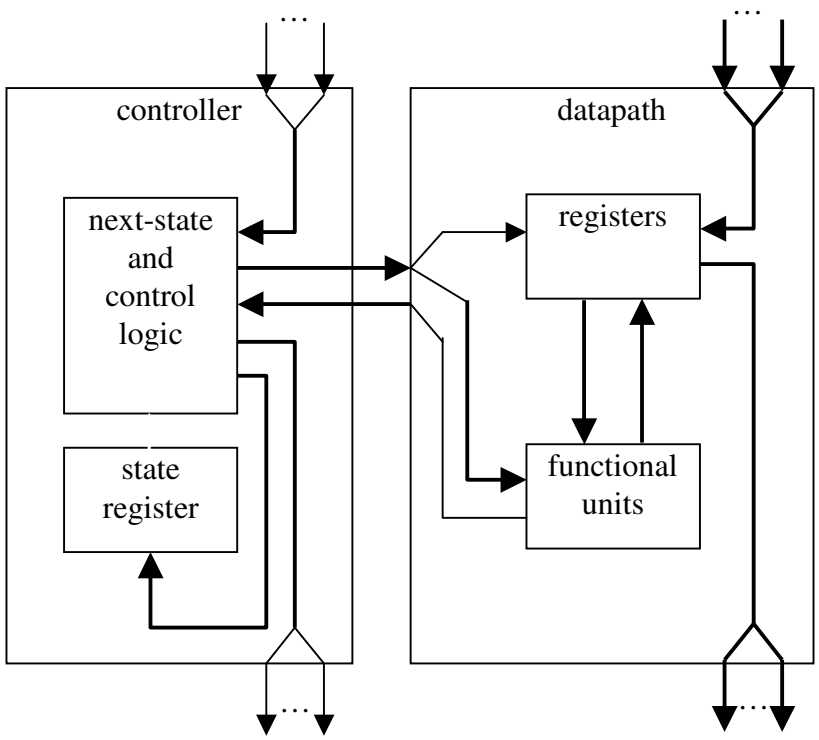


Tabela de Estado para o controle do GCD

Inputs							Outputs								
Q3	Q2	Q1	Q0	x_neq_y	x_lt_y	go_i	I3	I2	I1	I0	x_sel	y_sel	x_ld	y_ld	d_ld
0	0	0	0	*	*	*	0	0	0	1	X	X	0	0	0
0	0	0	1	*	*	0	0	0	1	0	X	X	0	0	0
0	0	0	1	*	*	1	0	0	1	1	X	X	0	0	0
0	0	1	0	*	*	*	0	0	0	1	X	X	0	0	0
0	0	1	1	*	*	*	0	1	0	0	0	X	1	0	0
0	1	0	0	*	*	*	0	1	0	1	X	0	0	1	0
0	1	0	1	0	*	*	1	0	1	1	X	X	0	0	0
0	1	0	1	1	*	*	0	1	1	0	X	X	0	0	0
0	1	1	0	*	0	*	1	0	0	0	X	X	0	0	0
0	1	1	0	*	1	*	0	1	1	1	X	X	0	0	0
0	1	1	1	*	*	*	1	0	0	1	X	1	0	1	0
1	0	0	0	*	*	*	1	0	0	1	1	X	1	0	0
1	0	0	1	*	*	*	1	0	1	0	X	X	0	0	0
1	0	1	0	*	*	*	0	1	0	1	X	X	0	0	0
1	0	1	1	*	*	*	1	1	0	0	X	X	0	0	1
1	1	0	0	*	*	*	0	0	0	0	X	X	0	0	0
1	1	0	1	*	*	*	0	0	0	0	X	X	0	0	0
1	1	1	0	*	*	*	0	0	0	0	X	X	0	0	0
1	1	1	1	*	*	*	0	0	0	0	X	X	0	0	0

Completando o projeto do Processador de Funcionalidade Única - GCD

- Projeto dos Componentes da Unidade de Processamento
- Implementação da lógica combinacional para a unidade de controle
- Esta não é uma implementação otimizada mas FUNCIONA



a view inside the controller and datapath

Otimizando processadores de propósito único

- Otimização: melhorar métricas de projeto
- Oportunidades de Otimização
  - Especificação inicial
  - FSMD
  - Unidade de Processamento
  - FSM

Otimizando a Descrição Inicial

- Análise do programa de forma a identificar possíveis otimizações
  - Número computações
  - Tamanho das variáveis
  - Complexidade de tempo e espaço
  - Operações usadas

Otimizando o programa

original program

```
 0: int  x, y;
 1: while (1) {
 2:   while (!go_i);
 3:   x = x_i;
 4:   y = y_i;
 5:   while  (x != y)  {
 6:      if  (x < y)
 7:         y = y - x;
        else
 8:         x = x - y;
      }
 9:   d_o = x;
    }
```

replace the subtraction operation(s) with modulo operation in order to speed up program

optimized program

```
 0: int  x, y, r;
 1: while (1) {
 2:   while (!go_i);
      // x must be the larger number
 3:   if (x_i >= y_i)  {
 4:      x=x_i;
 5:      y=y_i;
      }
 6:   else {
 7:      x=y_i;
 8:      y=x_i;
      }
 9:   while  (y != 0)  {
10:      r = x % y;
11:      x = y;
12:      y = r;
      }
13:   d_o = x;
    }
```

GCD(42, 8) - 9 iterations to complete the loop

x and y values evaluated as follows : (42, 8), (43, 8), (26,8), (18,8), (10, 8), (2,8), (2,6), (2,4), (2,2).

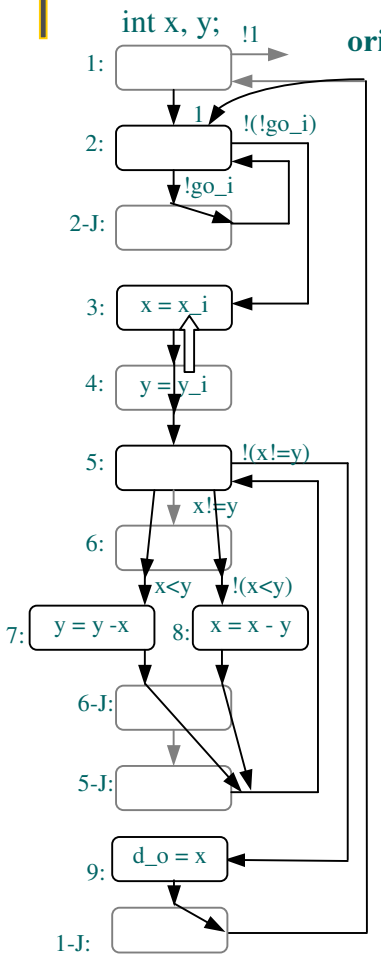
GCD(42,8) - 3 iterations to complete the loop

x and y values evaluated as follows: (42, 8), (8,2), (2,0)

Otimizando a FSMD

- Possíveis otimizações
  - Merge de Estados
    - Estados com constantes nas transições podem ser eliminados, transições são conhecidas
    - Estados com operações independentes podem ser agrupados
  - Separação de Estados
    - Estados incluindo operações complexas (a*b*c*d) podem ser quebrados em estados menores
  - Escalonamento

Otimizando a FSMD (cont.)



Elimina Estado 1

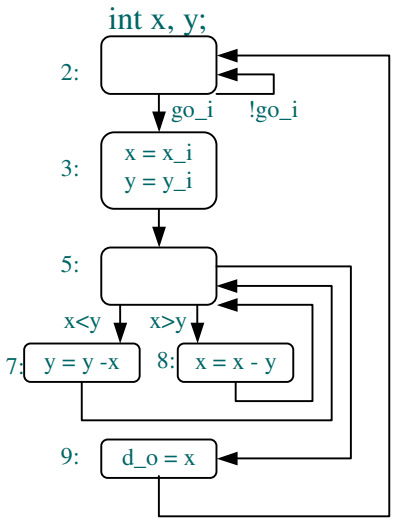
merge estados 2 e 2J

merge estados 3 e 4

merge estados 5 e 6

Elimina estados 5J e 6J

Elimina estado 1-J



Resumo

- Processadores de Propósito ÚNICO
  - Técnicas de projeto
  - Implementação de Algoritmos
  - Tipicamente se começa com uma FSMD
  - Ferramentas de CAD são imprescindíveis