

# Arquitetura de Sistemas Embarcados

Edna Barros (ensb@cin.ufpe.br)

Centro de Informática – UFPE



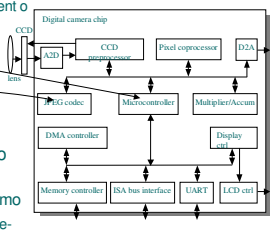
## Rot eiro

- Conceitos Básicos
- Lógica Combinacional
- Lógica Sequencial
- Proj etando um processador de propósito único

Ambiente de Projeto de Sistemas Embarcados 4

## I ntrodução

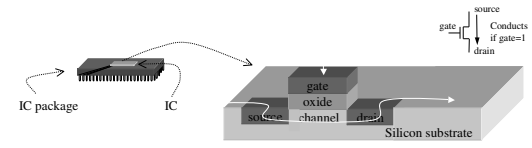
- Processador
  - Circuito digital que implementa tarefa computacional
  - Controle e unidade de processamento
  - Propósito Geral: variedade de tarefas
  - Propósito Único: uma tarefa particular
  - Propósito Único e Cust omizado: tarefa não padrão
- Processador de propósito único cust omizado:
  - Rápido, pequeno e baixo consumo
  - MAS : possui alto custo NRE, tempo-market long e apresenta pouca flexibilidade



Ambiente de Projeto de Sistemas Embarcados 2

## CMOS t ransist ores em silício

- Transistor
  - O Component e Básico dos Sistemas Digit ais
  - At ua com chave
  - Tensão no "gat e" controla fluxo de corrente da fonte para o "drain"

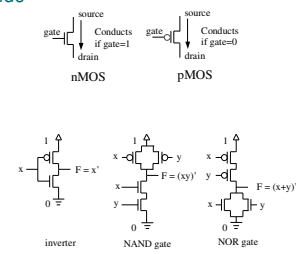


Ambiente de Projeto de Sistemas Embarcados 5

# Revisão: Proj etando um processador de propósito único cust omizado

## I mplement ações de T ransist ores CMOS

- Complement ary Metal Oxide Semiconductor
- Níveis Lógicos
  - 0 é 0V, 1 é 5V
- Dois tipos básicos
  - nMOS conduz se gat e=1
  - pMOS conduz se gat e=0
- Portas Básicas
  - I nvert er, NAND, NOR



Ambiente de Projeto de Sistemas Embarcados 6

### Port as Lógicas Básicas

<p><math>x \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>F</td></tr> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </table> <p><math>F = x</math> Driver</p>	x	F	0	0	1	1	<p><math>x \ y \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>y</td><td>F</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table> <p><math>F = x \cdot y</math> AND</p>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1	<p><math>x \ y \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>y</td><td>F</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table> <p><math>F = x + y</math> OR</p>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1	<p><math>x \ y \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>y</td><td>F</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table> <p><math>F = x \oplus y</math> XOR</p>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	F																																																					
0	0																																																					
1	1																																																					
x	y	F																																																				
0	0	0																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				
x	y	F																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	1																																																				
x	y	F																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				
<p><math>x \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>F</td></tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </table> <p><math>F = x'</math> Inverter</p>	x	F	0	1	1	0	<p><math>x \ y \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>y</td><td>F</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table> <p><math>F = (x \cdot y)'</math> NAND</p>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0	<p><math>x \ y \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>y</td><td>F</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table> <p><math>F = (x + y)'</math> NOR</p>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1	<p><math>x \ y \rightarrow F</math></p> <table border="1"> <tr><td>x</td><td>y</td><td>F</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table> <p><math>F = x \odot y</math> XNOR</p>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	F																																																					
0	1																																																					
1	0																																																					
x	y	F																																																				
0	0	1																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				
x	y	F																																																				
0	0	1																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				
x	y	F																																																				
0	0	1																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				

Ambiente de Projeto de Sistemas Embarcados 7

### Circuitos Sequenciais

<p>load, clear, n-bit Register, n, Q</p>	<p>shift, n-bit Shift register, n, Q</p>	<p>n-bit Counter, n, Q</p>
<p>Q = 0 if clear=1, 1 if load=1 and clock=1, Q(previous) otherwise.</p>	<p>Q = lsb - Content shifted - 1 stored in msb</p>	<p>Q = 0 if clear=1, 0 if clock=1, Q(prev)+1 if count=1 and clock=1.</p>

Ambiente de Projeto de Sistemas Embarcados 10

### Projeto de Circuitos Combinacionais

A) Problem description  
y is 1 if a is 1, or b and c are 1. z is 1 if b or c is 1, but not both, or if all are 1.

B) Truth table

a	b	c	y	z
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
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  
 $y = a'bc + ab'c + abc' + abc$   
 $z = a'b'c + a'bc' + abc' + abc$

D) Minimized output equations

abc	00	01	11	10
y	0	1	1	1
z	0	1	0	1

$y = a + bc$   
 $z = ab + b'c + bc'$

E) Logic Gates

Ambiente de Projeto de Sistemas Embarcados 8

### 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	11 10 x
0 0	0 0 0 0
0 1	0 0 1 0
1 0	0 1 0 1
1 1	1 1 0 0
0 0	1 0 1 1
0 1	1 1 0 1
1 0	1 1 1 1
1 1	1 0 1 1

Ambiente de Projeto de Sistemas Embarcados 11

### Circuitos Combinacionais

<p><math>2^m \times 1</math> Multiplexer</p>	<p><math>2^m \times 1</math> Decoder</p>	<p>Adder</p>	<p>Comparator</p>	<p>Function ALU</p>
<p>O = 0 if S=0,00 1 if S=0,01 ... 1(m-1) if S=1,11</p>	<p>O0 = 1 if In0,00 O1 = 1 if In0,01 ... O(m-1) = 1 if In1,11</p>	<p>sum = A+B (first n bits) carry = (n+1)'th bit of A+B</p>	<p>less = 1 if A&lt;B equal = 1 if A=B greater = 1 if A&gt;B</p>	<p>O = A op B op determined by S.</p>
	<p>With enable input e → all O's are 0 if e=0</p>	<p>With carry-in input C1 → sum = A + B + C1</p>		<p>May have status outputs carry, zero, etc.</p>

Ambiente de Projeto de Sistemas Embarcados 9

### Projeto de Circuitos Sequenciais

D) State Table (Moore-type)

Inputs	Outputs
Q1 Q0 a	11 10 x
0 0	0 0 0 0
0 1	0 0 1 0
1 0	0 1 0 1
1 1	1 1 0 0
0 0	1 0 1 1
0 1	1 1 0 1
1 0	1 1 1 1
1 1	1 0 1 1

E) Minimized Output Equations

$11 - Q1'Q0a + Q1a' + Q1Q0'$   
 $10 - Q0a' + Q0'a$   
 $x - Q1Q0$

F) Combinational Logic

Ambiente de Projeto de Sistemas Embarcados 12

### Modelo Básico de um Processador de Propósito Único

The diagram illustrates the basic architecture of a single-purpose processor. On the left, a high-level block diagram shows external control inputs and outputs connected to a controller, and external data inputs and outputs connected to a datapath. On the right, a detailed view of the controller and datapath shows the controller containing next-state and control logic, a state register, and registers. The datapath contains functional units and registers. Arrows indicate the flow of control signals and data between these components.

controller and datapath

a view inside the controller and datapath

Fonte: Universidade Federal de Pernambuco

Ambiente de Projeto de Sistemas Embarcados 13

### 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 escritos
  - Use multiplexador para fontes múltiplas
- Crie um identificador único para cada saída e entrada da unidade de processamento

This diagram shows the internal structure of the datapath unit. It features several registers (labeled 1 through 10) that store variables. These registers are connected to various functional units: an adder (x + y), a subtractor (x - y), a multiplier (x \* y), and a divider (x / y). Multiplexers are used to select between different data sources for the functional units. The datapath also includes control logic for operations like 'add', 'sub', 'mul', and 'div'.

Fonte: Universidade Federal de Pernambuco

Ambiente de Projeto de Sistemas Embarcados 16

### Exemplo: great est common divisor

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

The example illustrates the conversion of a GCD algorithm into a finite state machine (FSM). (a) shows a black-box view of the GCD function with inputs x<sub>i</sub>, y<sub>i</sub> and output d<sub>o</sub>. (b) shows the desired functionality in pseudocode:
 

```

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

 (c) shows the state diagram for this FSM, with states 1 through 13 representing different stages of the algorithm's execution.

Fonte: Universidade Federal de Pernambuco

Ambiente de Projeto de Sistemas Embarcados 14

### Criando um controle baseado em FSM

- Mesma estrutura que uma FSM
- Substitua ações complexas por configurações da unidade de processamento

This diagram shows the FSM control logic implemented in a controller. The controller has states 0000 through 1100, each corresponding to a step in the GCD algorithm. The control logic uses combinational logic to generate control signals for the datapath unit, such as 'add', 'sub', 'mul', and 'div'. The datapath unit is shown on the right, receiving these control signals and performing the corresponding operations on the data from the registers.

Fonte: Universidade Federal de Pernambuco

Ambiente de Projeto de Sistemas Embarcados 17

### Templatas 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
```

The flowcharts illustrate the state transitions for these control structures. For an assignment statement, the state moves from the assignment to the next statement. For a loop, the state moves from the condition to the loop body, and back to the condition if it is true. For a branch, the state moves to the appropriate branch based on the condition and then to the next statement.

Fonte: Universidade Federal de Pernambuco

Ambiente de Projeto de Sistemas Embarcados 15

### Conectando controle e unidade de processamento

This diagram shows the connection between the controller implementation model and the datapath unit. The controller implementation model shows the control logic and state register. The datapath unit is shown on the right, receiving control signals from the controller and performing operations on the data from the registers.

Fonte: Universidade Federal de Pernambuco

Ambiente de Projeto de Sistemas Embarcados 18

## Tabela de Estado para o controle do GCD

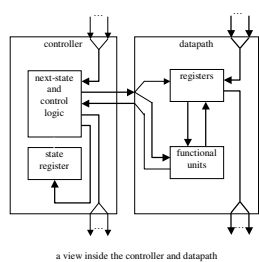
Inputs							Outputs								
Q3	Q2	Q1	Q0	x_n	x_lt	go_i	E3	E2	E1	E0	x_se	y_se	x_ld	y_ld	d_ld
0	0	0	0	*	*	*	0	0	0	1	X	X	0	0	0
0	0	0	1	*	*	*	0	0	1	0	X	X	0	0	0
0	0	1	0	*	*	*	1	0	0	1	1	X	X	0	0
0	0	1	1	*	*	*	0	0	0	1	X	X	0	0	0
0	1	0	0	*	*	*	0	1	0	1	0	0	X	1	0
0	1	0	1	*	*	*	1	0	1	1	X	X	0	0	1
0	1	1	0	*	*	*	0	1	1	0	X	X	0	0	0
0	1	1	1	*	*	*	1	0	0	0	X	X	0	0	0
1	0	0	0	*	*	*	1	0	0	1	1	X	1	0	1
1	0	0	1	*	*	*	1	0	1	1	0	X	X	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

## Otimizando a Descrição Inicial

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

## 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



## Otimizando o programa

```

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

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

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

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).

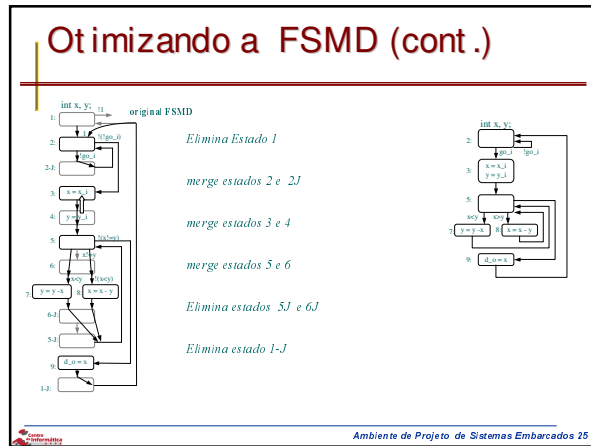
GCD(42, 8) - 3 iterations to complete the loop  
 x and y values evaluated as follows: (42, 8), (8, 2), (2, 0)

## Otimizando processadores de propósito único

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

## 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



- ### Resumo
- Processadores de Propósito Único
    - Técnicas de projeto
    - Implementação de Algoritmos
    - Tipicamente se começa com uma FSM
    - Ferramentas de CAD são imprescindíveis
- Ambiente de Projeto de Sistemas Embarcados 26