UNIVERSIDADE FEDERAL DE RORAIMA

Programação em VHDL

Prof. Herbert Oliveira Rocha

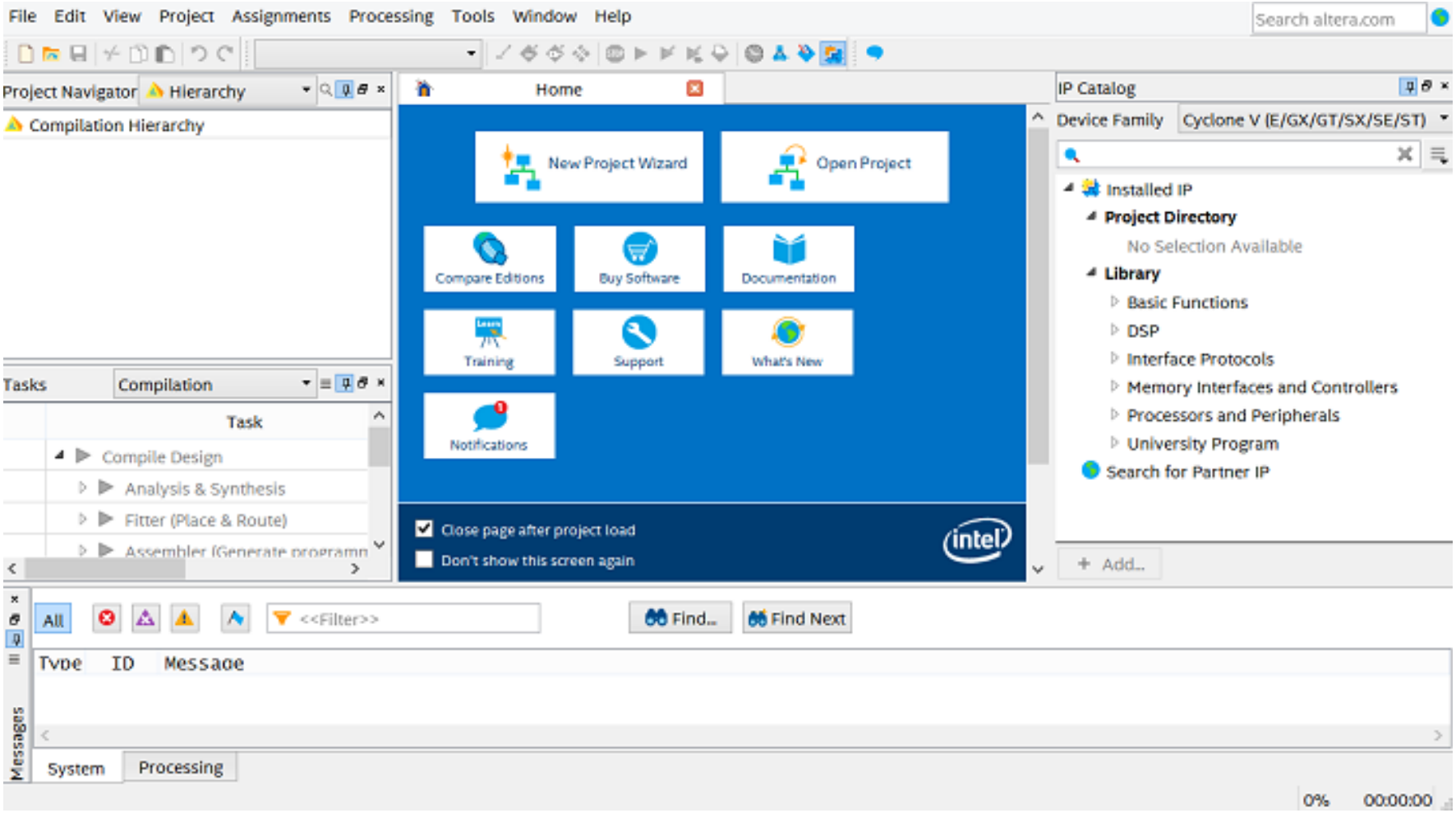
UNIVERSIDADE FEDERAL DE RORAIMA

Programação em VHDL

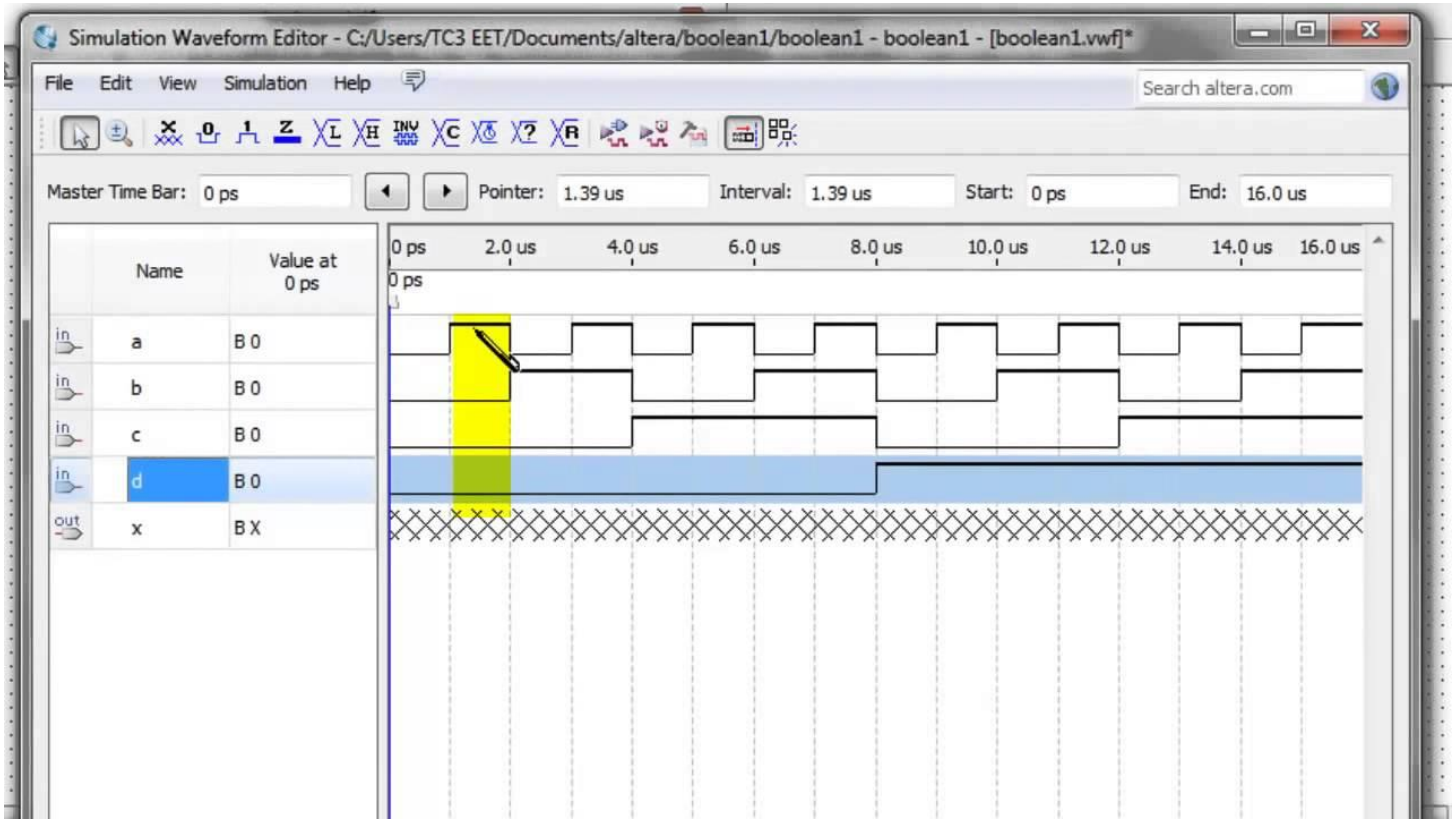
Baseado nas aulas do Prof. Dr. Mauricio Figueiredo and
Prof. Dr. Raimundo Barreto - UFAM

Prof. Herbert Oliveira Rocha

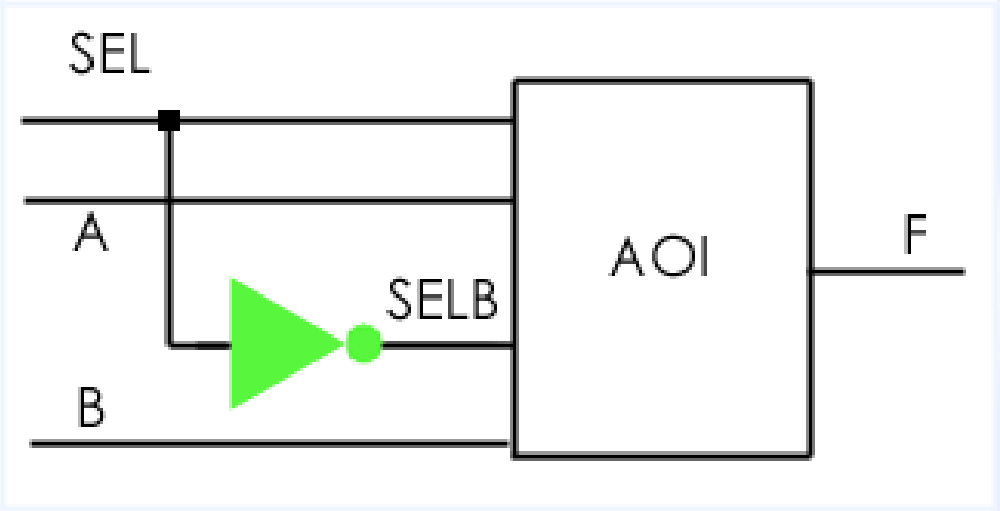
Quartus


File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Project Navigator
Hierarchy
Compilation Hierarchy
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Home
New Project Wizard
Open Project
Compare Editions
Buy Software
Documentation
Training
Support
What's New
Notifications
Close page after project load
Don't show this screen again
intel
IP Catalog
Device Family
Cyclone V (E/GX/GT/SX/SE/ST)
Installed IP
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Search for Partner IP
Add...
All
<<Filter>>
Find...
Find Next
Type ID Message
Messages
System
Processing
0%
00:00:00

Quartus


Simulation Waveform Editor - C:/Users/TC3 EET/Documents/altera/boolean1/boolean1 - boolean1 - [boolean1.vwf]*
File Edit View Simulation Help
Search altera.com
Master Time Bar: 0 ps
Pointer: 1.39 us
Interval: 1.39 us
Start: 0 ps
End: 16.0 us
Name
Value at 0 ps
a B 0
b B 0
c B 0
d B 0
x B X
0 ps 2.0 us 4.0 us 6.0 us 8.0 us 10.0 us 12.0 us 14.0 us 16.0 us

Components and Port Maps



Components and Port Maps

Conexão entre componentes já existentes

Criação de um componente a partir de uma **entity**

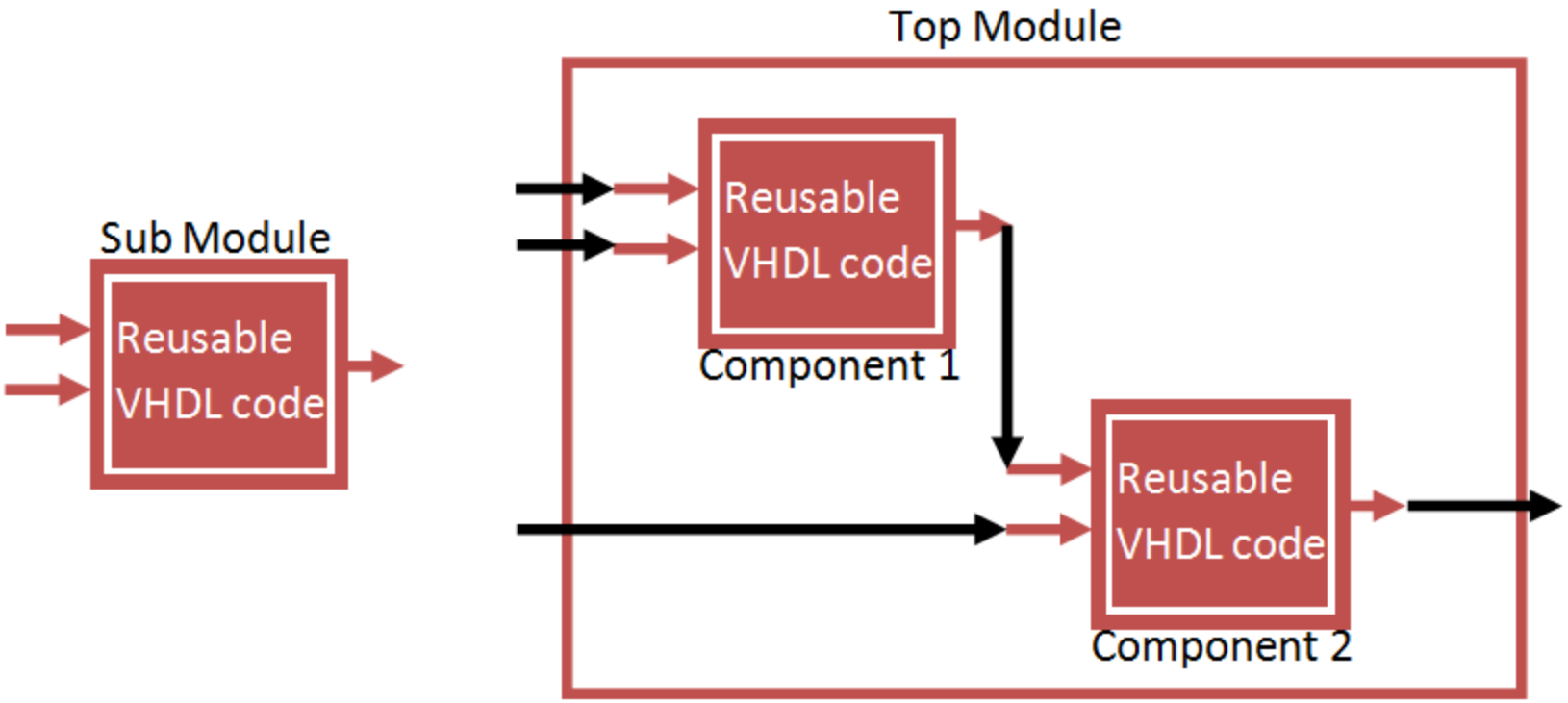
Entity

```
entity INV is
  port (A: in STD_LOGIC;
  F: out STD_LOGIC);
end INV;
```

Component

```
component INV
  port (A: in STD_LOGIC;
  F: out STD_LOGIC);
end component;
```

Components and Port Maps



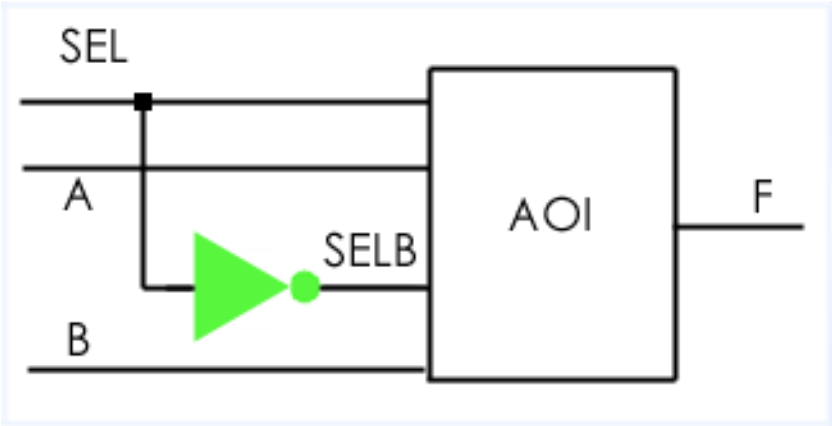
Components and Port Maps

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity MUX2 is
  port (SEL, A, B: in STD_LOGIC;
  F : out STD_LOGIC);
end;

architecture STRUCTURE of MUX2 is

component INV
  port (A: in STD_LOGIC;
  F: out STD_LOGIC);
end component;
```

As declarações de dois componentes (para INV e AOI) aparecem na parte declarativa da arquitetura (que é um termo técnico VHDL que significa que as declarações de componente são codificadas antes do início).

Components and Port Maps

```
component AOI
  port (A, B, C, D: in STD_LOGIC;
  F : out STD_LOGIC);
end component;

signal SELB: STD_LOGIC;

begin
  G1: INV port map (SEL, SELB);
  G2: AOI port map (SEL, A, SELB, B, F);
end;
```



Components and Port Maps

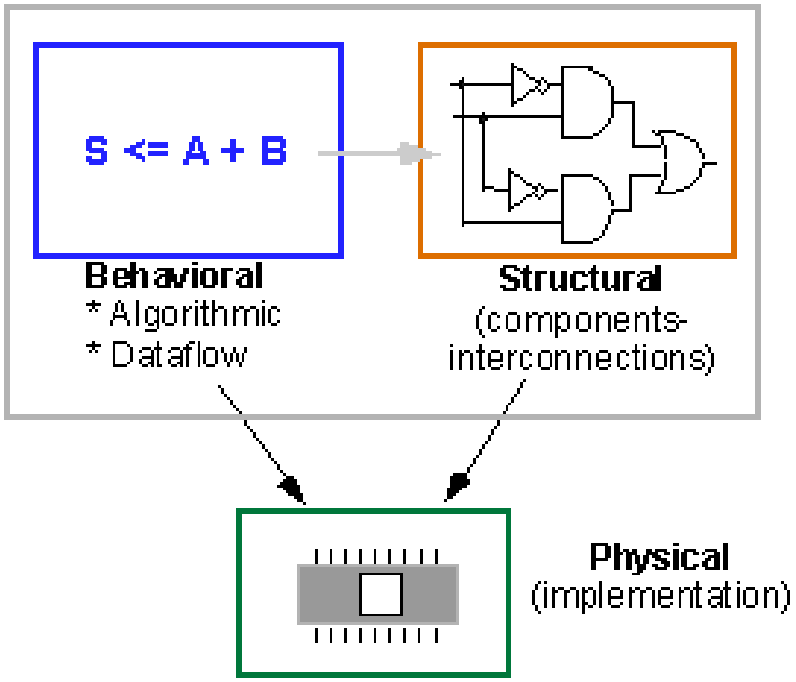
Port Maps

- As portas em uma declaração de componente normalmente deve coincidir com as portas na declaração da entidade um-para-um.

- A declaração de componente define os nomes, a ordem, o modo e os tipos de portas para ser usado quando o componente é instanciado no corpo arquitetura.

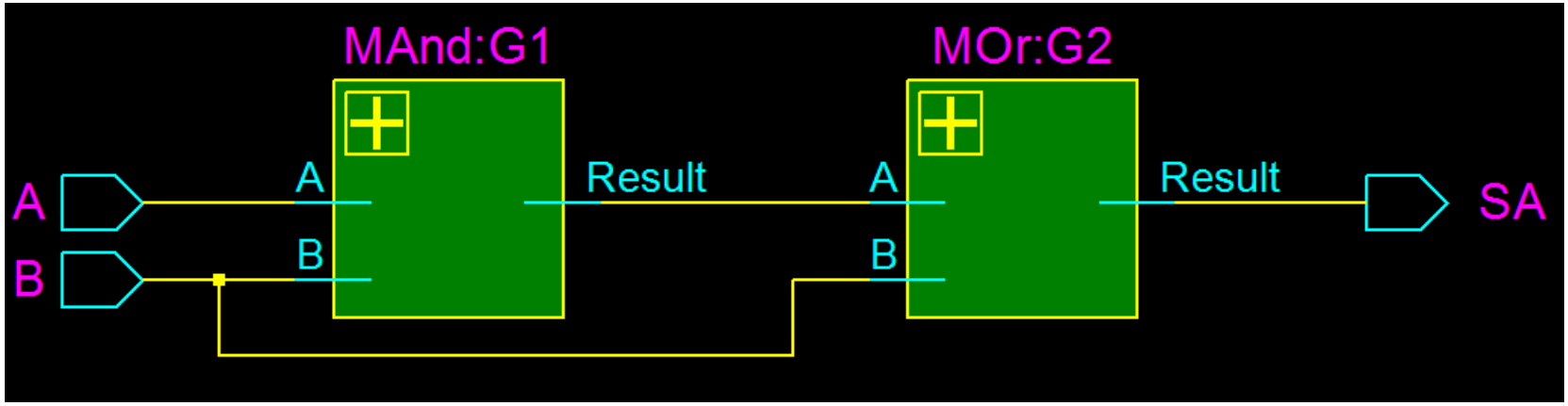
- Instanciando um componente implica fazer uma cópia local da entidade correspondente no projeto - um componente é declarado uma vez dentro de qualquer arquitetura, mas pode ser instanciado qualquer número de vezes.

Praticando!!!

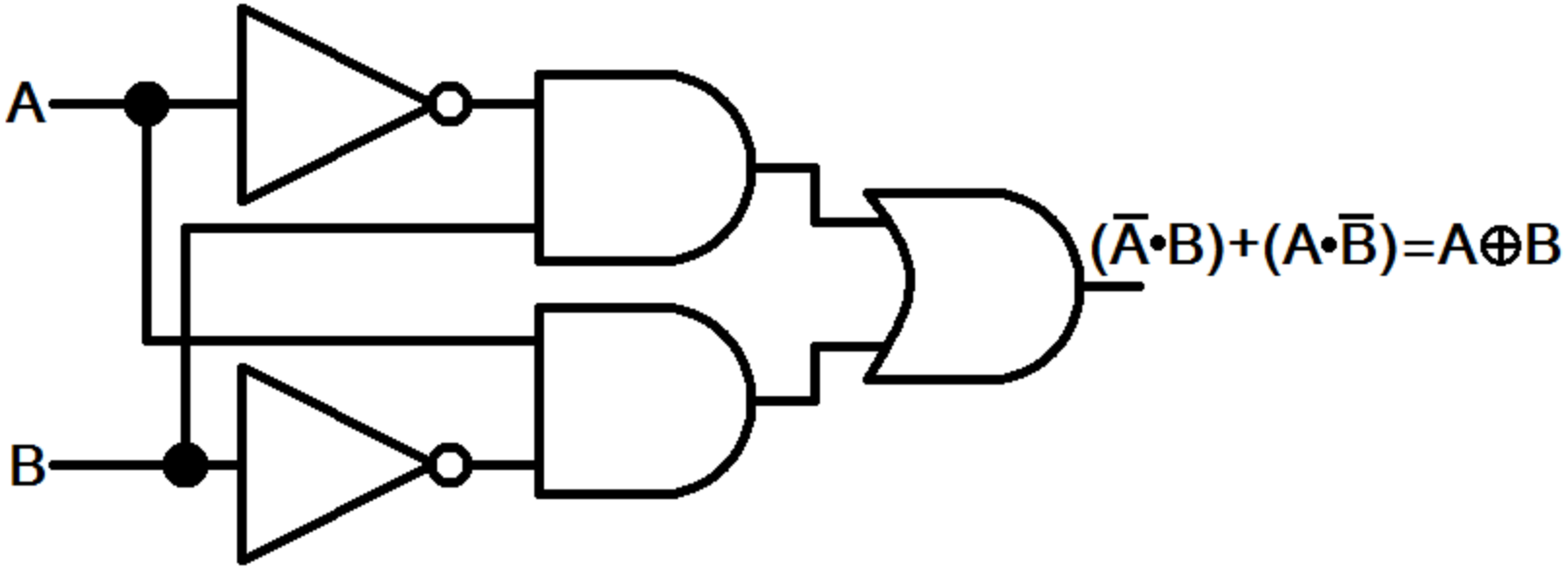

S <= A + B
Behavioral
* Algorithmic
* Dataflow
Structural
(components-
interconnections)
Physical
(implementation)

Praticando!!!

Components and Port Maps



XOR usando componentes



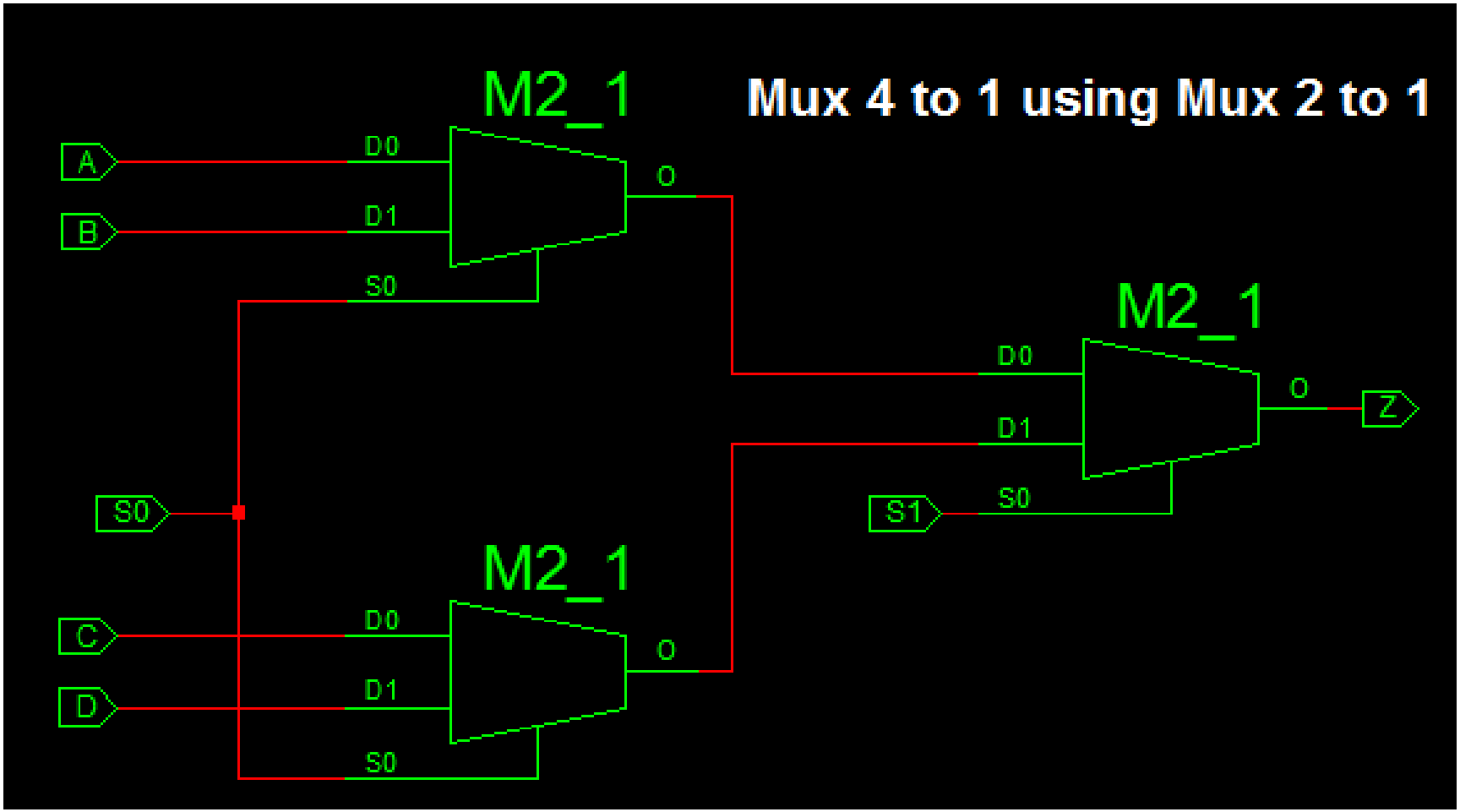
Exemplo 1

Multiplexador 2x1

```
ENTITY Multiplex2x1 IS
       PORT(
         A, B, S     : IN  BIT;
             Saida   : OUT BIT
       );
END Multiplex2x1;


ARCHITECTURE Mux2x1 OF Multiplex2x1 IS
BEGIN

       Saida <= (A AND S) OR (B AND NOT(S));

END Mux2x1;
```

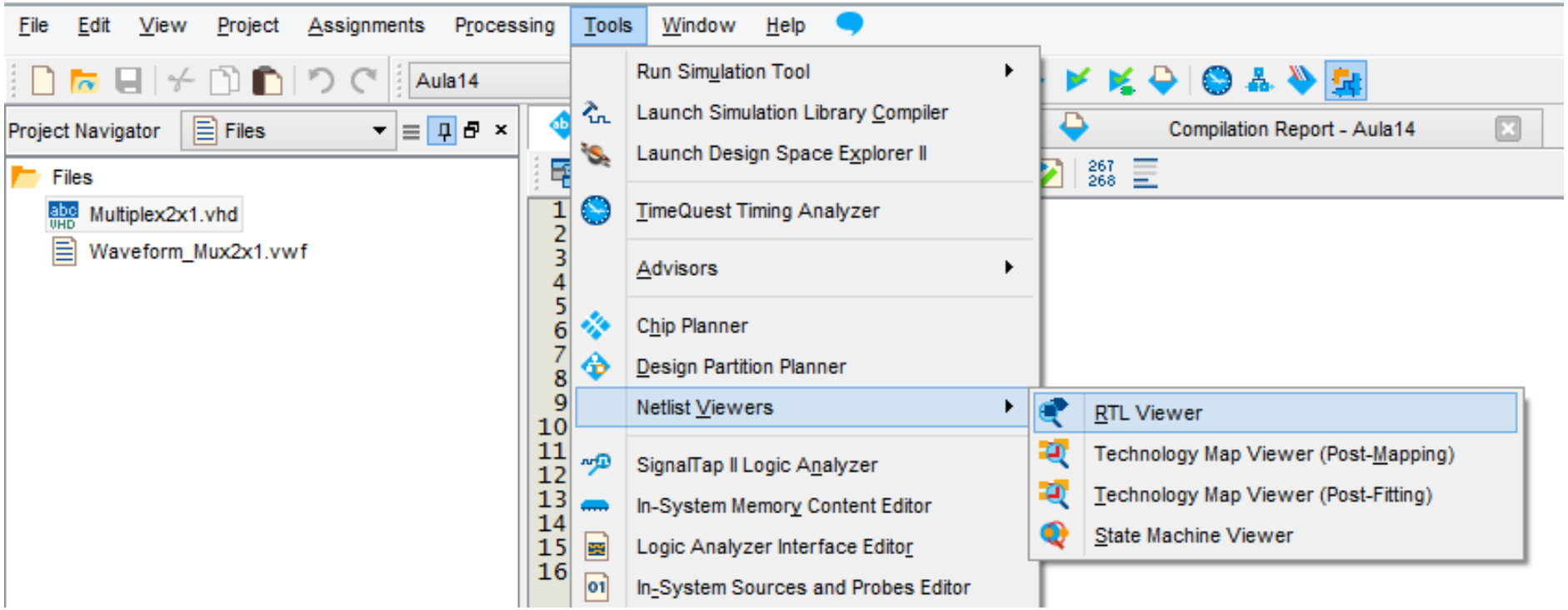
Exemplo 1

Multiplexador 4x1



Exemplo 1

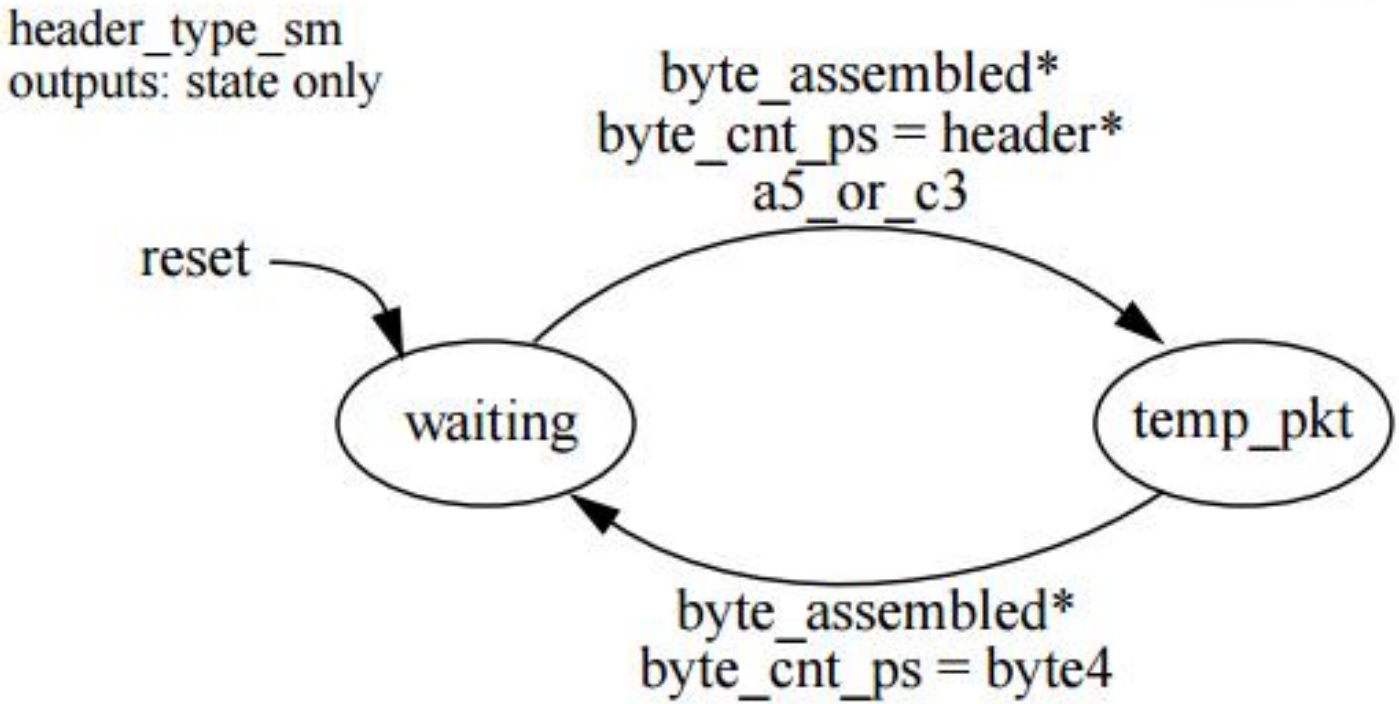
Multiplexador 4x1 – RTL Viewer



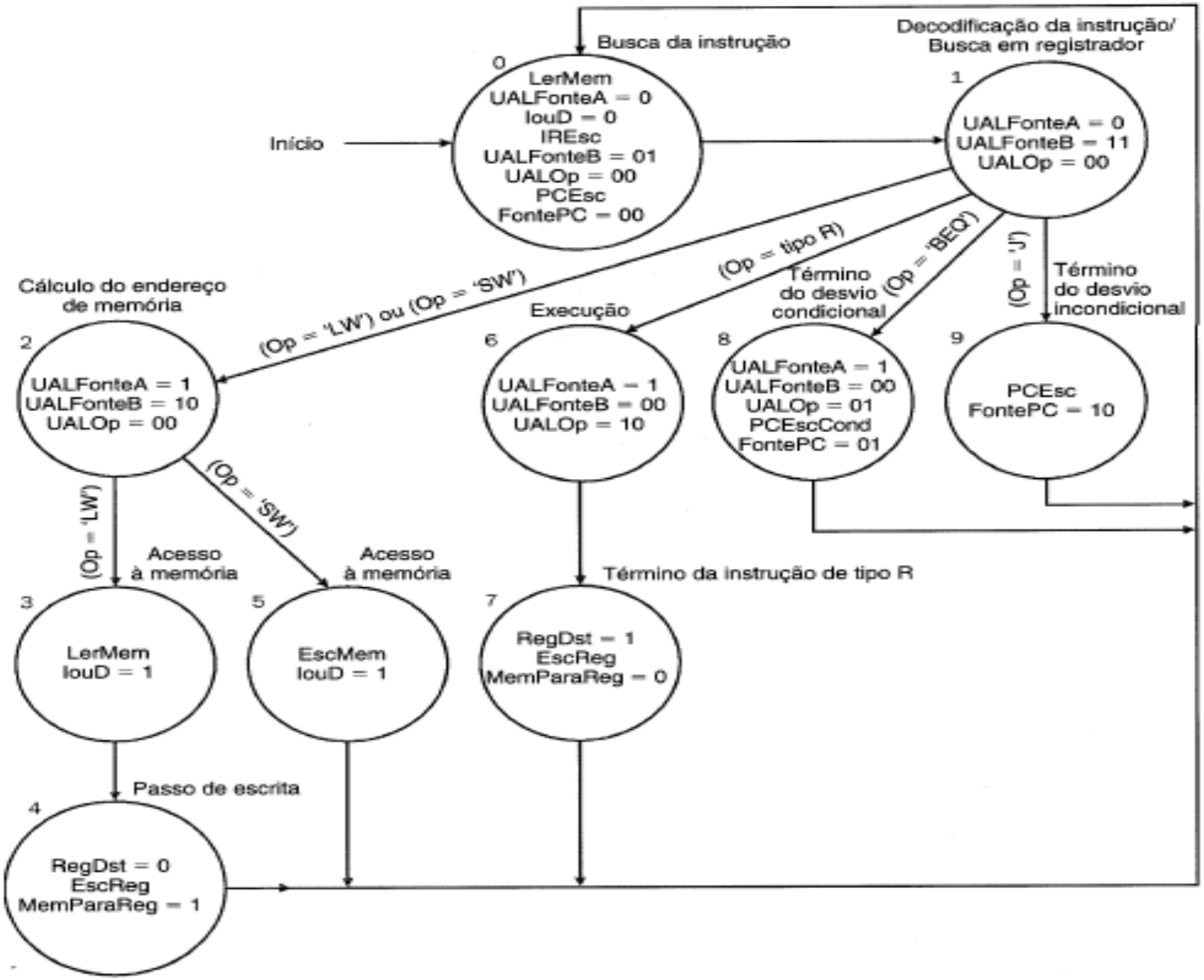
State Machine

Maquina de Estados em VHDL

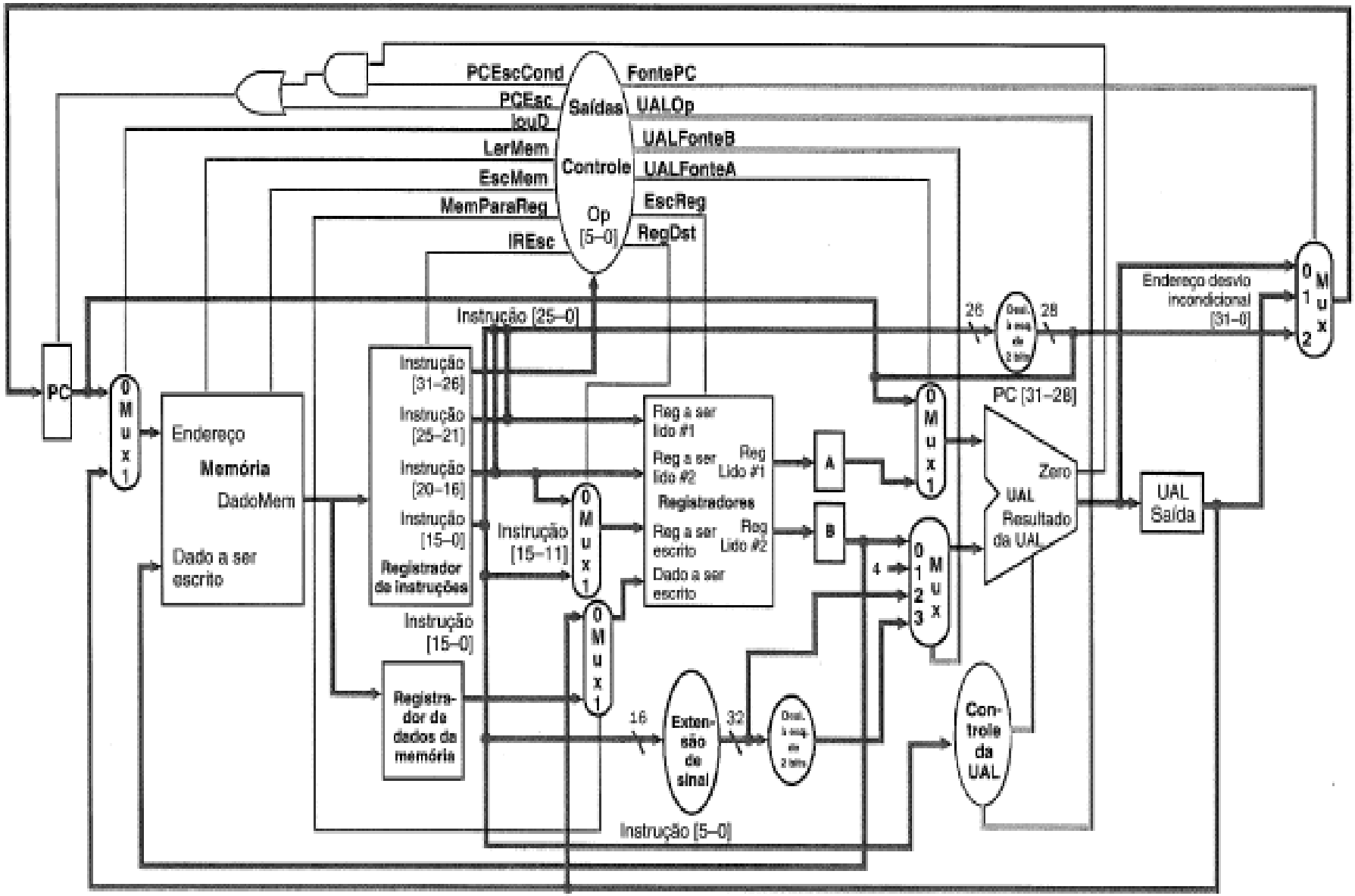
Destinam-se a ser portátil, de fácil compreensão, limpo, e dar resultados consistentes com praticamente qualquer ferramenta de síntese.



```
header_type_sm:
 PROCESS (clk_50, reset_n, a5_or_c3, byte_assembled, byte_cnt_ps,
header_type_ps, header_type_ns)
 BEGIN
   --clocked part
   IF (reset_n = '0') THEN
     header_type_ps <= waiting;
   ELSIF (clk_50'EVENT AND clk_50 = '1') THEN
     header_type_ps <= header_type_ns;
   END IF;

   --combinatorial part
   CASE header_type_ps IS
     WHEN waiting =>
      IF (byte_assembled = '1') AND (byte_cnt_ps = header) AND
(a5_or_c3 = '1') THEN

        header_type_ns <= temp_pkt;
      ELSE
        header_type_ns <= waiting;
      END IF ;
     WHEN temp_pkt =>
      IF (byte_assembled = '1') AND (byte_cnt_ps = byte4) THEN
        header_type_ns <= waiting;
      ELSE
        header_type_ns <= temp_pkt;
      END IF ;
   END CASE;
 END PROCESS header_type_sm;
```


Busca da instrução
Decodificação da instrução/
Busca em registrador
0
LerMem
UALFonteA = 0
IouD = 0
IREsc
UALFonteB = 01
UALOp = 00
PCEsc
FontePC = 00
Início
1
UALFonteA = 0
UALFonteB = 11
UALOp = 00
(Op = 'LW') ou (Op = 'SW')
(Op = tipo R)
(Op = 'BEQ')
(Op = 'J')
Cálculo do endereço
de memória
Execução
Término
do desvio
condicional
Término
do desvio
incondicional
2
UALFonteA = 1
UALFonteB = 10
UALOp = 00
6
UALFonteA = 1
UALFonteB = 00
UALOp = 10
8
UALFonteA = 1
UALFonteB = 00
UALOp = 01
PCEscCond
FontePC = 01
9
PCEsc
FontePC = 10
(Op = 'LW')
(Op = 'SW')
Acesso
à memória
Acesso
à memória
Término da instrução de tipo R
3
LerMem
IouD = 1
5
EscMem
IouD = 1
7
RegDst = 1
EscReg
MemParaReg = 0
Passo de escrita
4
RegDst = 0
EscReg
MemParaReg = 1

Praticando


PCEscCond
PCEsc
IouD
LerMem
EscMem
MemParaReg
IREsc
Saídas
Controle
Op
[5–0]
FontePC
UALOp
UALFonteB
UALFonteA
EscReg
RegDst
Instrução [25–0]
26
28
Desl. à esq. de 2 bits
Endereço desvio incondicional [31–0]
PC [31–28]
PC
0
M
u
x
1
Endereço
Memória
DadoMem
Dado a ser escrito
Instrução [31–26]
Instrução [25–21]
Instrução [20–16]
Instrução [15–0]
Registrador de instruções
Instrução [15–11]
Instrução [15–0]
Registrador de dados de memória
Reg a ser lido #1
Reg a ser lido #2
Registradores
Reg a ser escrito
Dado a ser escrito
Reg Lido #1
Reg Lido #2
A
B
4
Zero
UAL
Resultado da UAL
UAL Saída
16
Extensão de sinal
32
Instrução [5–0]
Controle da UAL