ESRG
Embedded Systems
Research Group

CENTROALGORITMI

Simple Processor:
Data Path & Control Unit

Adriando Tavares, Pedro X. Matos

Universidade do Minho
Escola de Engenharia

POPH
PROGRAMA OPERACIONAL POTENCIAL HUMANO

QREN
QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL
PORTUGAL 2007.2013

UNIÃO EUROPEIA
Fundo Social Europeu

Outline

- Introduction
- Data path
- Control Unit
- Control unit + Data path
- Simulation
- On real Hardware

ESRG
Embedded Systems
Research Group

Introduction

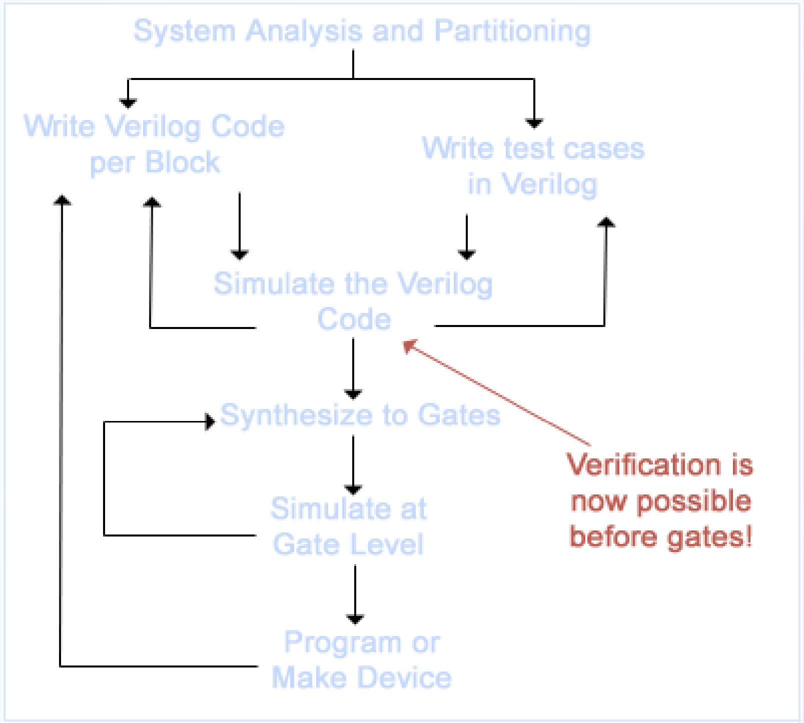
CENTROALGORITMI

- This presentation shows you a simple data path and Control Unit of a simple processor;
- The circuit was implemented using Verilog Hardware description language;
- The simulation will show us the correct behavior of the implemented circuit;
- Finaly the circuit will be burned down into your hardware boards;

Introduction

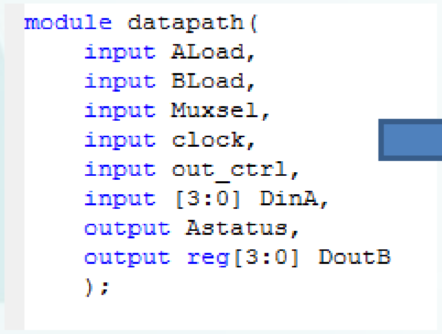
- Verilog
  - The Verilog HDL is an IEEE standard hardware description language. It is widely used in the design of digital integrated circuits.

- Design Flow

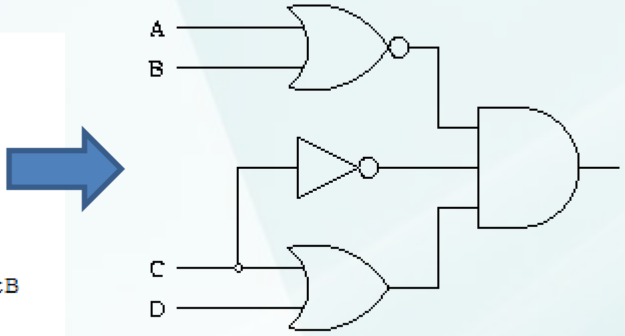

System Analysis and Partitioning
Write Verilog Code per Block
Write test cases in Verilog
Simulate the Verilog Code
Synthesize to Gates
Simulate at Gate Level
Program or Make Device
Verification is now possible before gates!

Introduction

- Tool
  - Vivado → Logic synthesize
  - ISIM simulator → Built-in Vivado → Logic Simulator

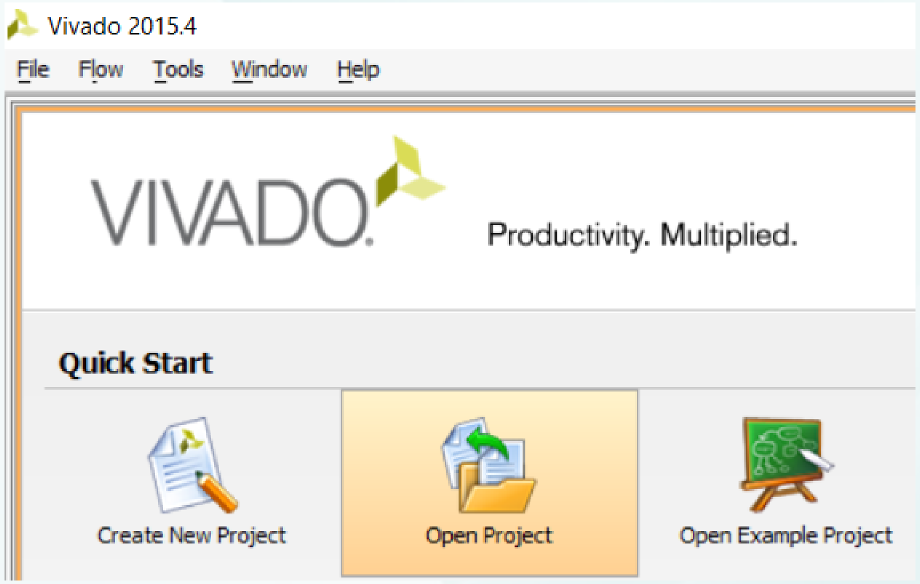


```
module datapath(
     input ALoad,
     input BLoad,
     input Muxsel,
     input clock,
     input out_ctrl,
     input [3:0] DinA,
     output Astatus,
     output reg[3:0] DoutB
     );
```



Introduction

- Copy the project into your projects folder and then, open the Vivado tool:



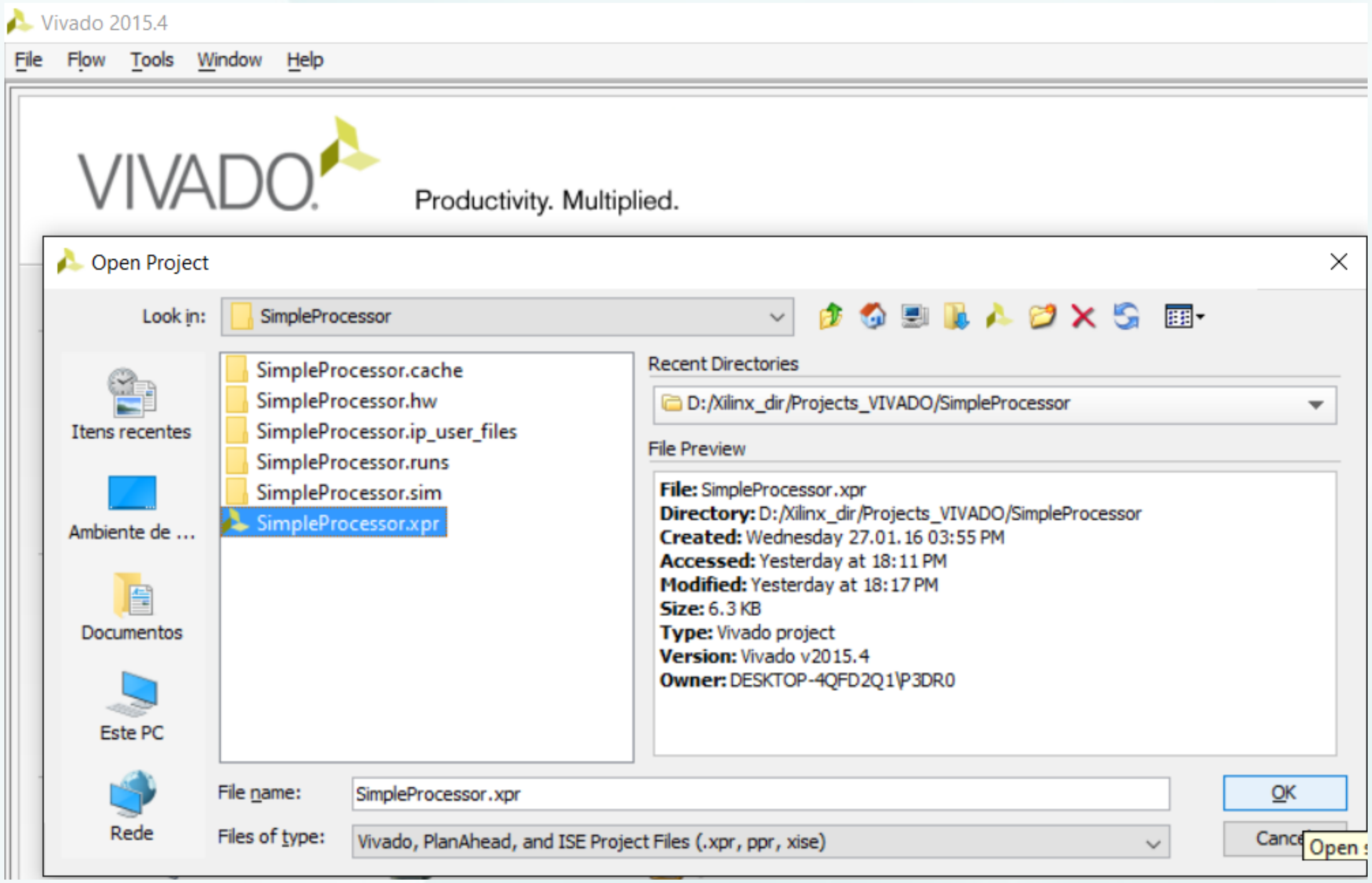
- Click on, “Open Project”.

ESRG
Embedded Systems
Research Group

Introduction

CENTROALGORITMI

- Choose the SimpleProcessor project and hit "ok"



Look in: SimpleProcessor

SimpleProcessor.cache
SimpleProcessor.hw
SimpleProcessor.ip_user_files
SimpleProcessor.runs
SimpleProcessor.sim
SimpleProcessor.xpr

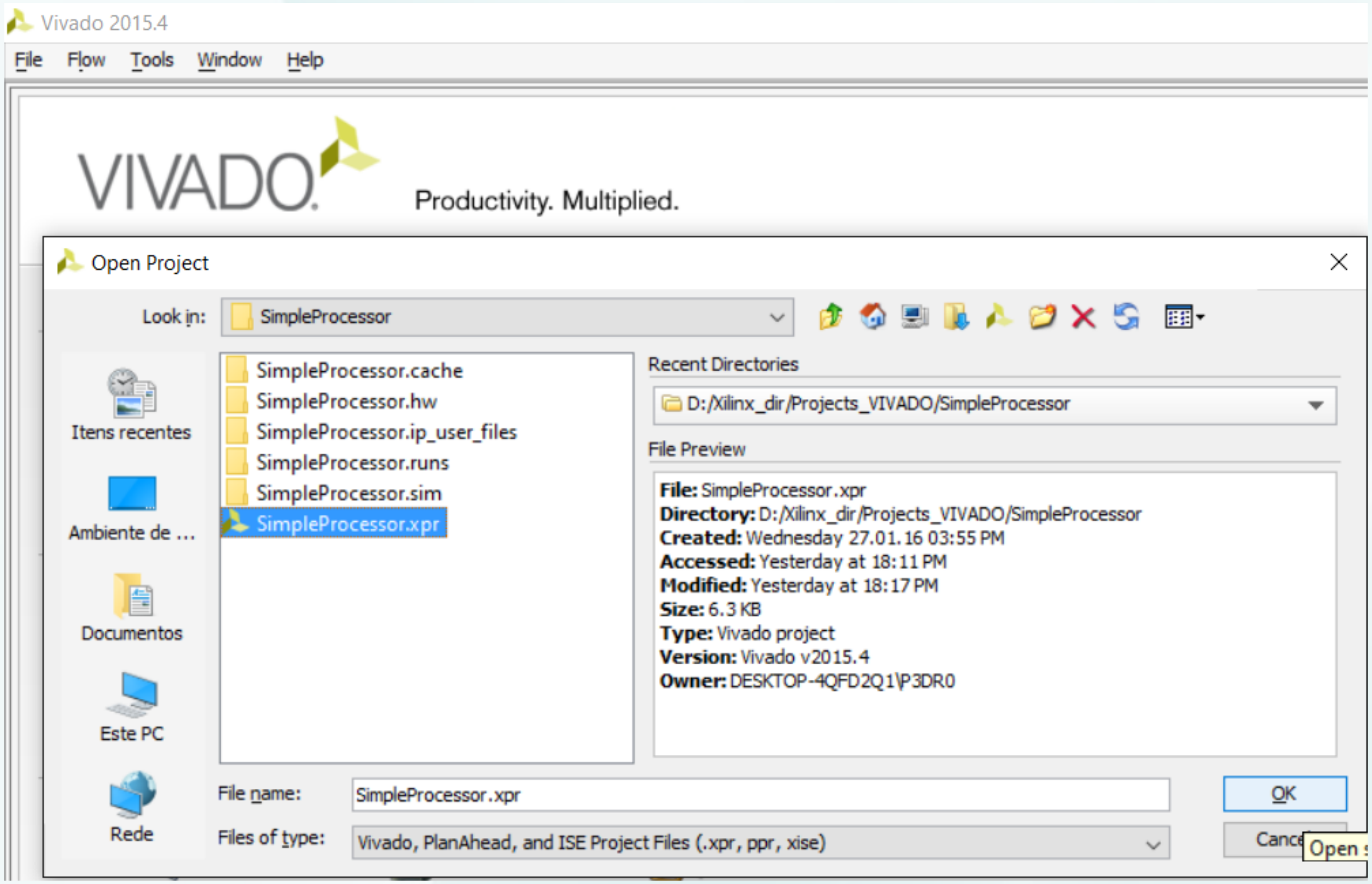
Recent Directories
D:/Xilinx_dir/Projects_VIVADO/SimpleProcessor

File Preview
File: SimpleProcessor.xpr
Directory: D:/Xilinx_dir/Projects_VIVADO/SimpleProcessor
Created: Wednesday 27.01.16 03:55 PM
Accessed: Yesterday at 18:11 PM
Modified: Yesterday at 18:17 PM
Size: 6.3 KB
Type: Vivado project
Version: Vivado v2015.4
Owner: DESKTOP-4QFD2Q1\P3DR0

File name: SimpleProcessor.xpr
Files of type: Vivado, PlanAhead, and ISE Project Files (.xpr, ppr, xise)

Introduction

- Choose the SimpleProcessor project and hit “ok”



Look in: SimpleProcessor

SimpleProcessor.cache
SimpleProcessor.hw
SimpleProcessor.ip_user_files
SimpleProcessor.runs
SimpleProcessor.sim
SimpleProcessor.xpr

Recent Directories: D:/Xilinx_dir/Projects_VIVADO/SimpleProcessor

File: SimpleProcessor.xpr
Directory: D:/Xilinx_dir/Projects_VIVADO/SimpleProcessor
Created: Wednesday 27.01.16 03:55 PM
Accessed: Yesterday at 18:11 PM
Modified: Yesterday at 18:17 PM
Size: 6.3 KB
Type: Vivado project
Version: Vivado v2015.4
Owner: DESKTOP-4QFD2Q1\P3DR0

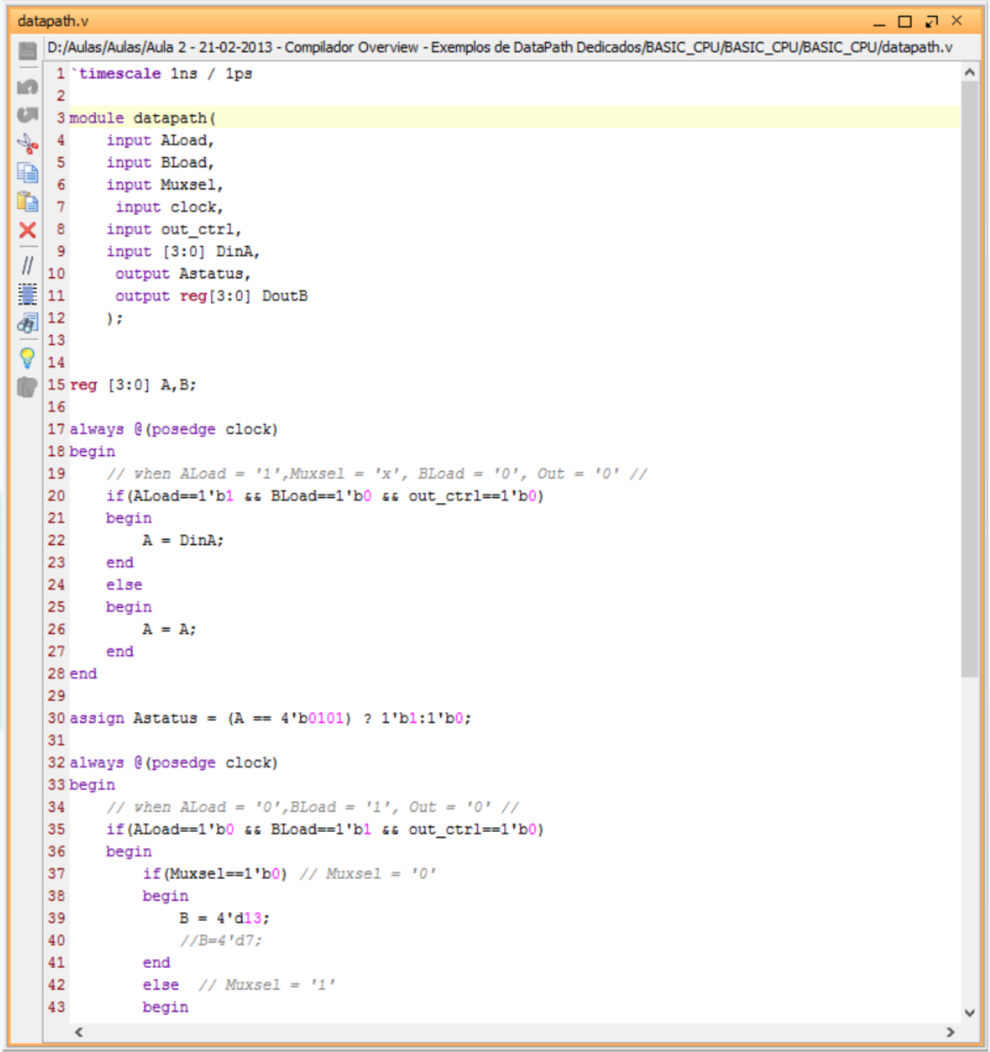
File name: SimpleProcessor.xpr
Files of type: Vivado, PlanAhead, and ISE Project Files (.xpr, ppr, xise)

ESRG
Embedded Systems
Research Group

Datapath

CENTROALGORITMI

- Open the datapath.v file and examine it carefully:


datapath.v

D:/Aulas/Aulas/Aula 2 - 21-02-2013 - Compilador Overview - Exemplos de DataPath Dedicados/BASIC_CPU/BASIC_CPU/BASIC_CPU/datapath.v

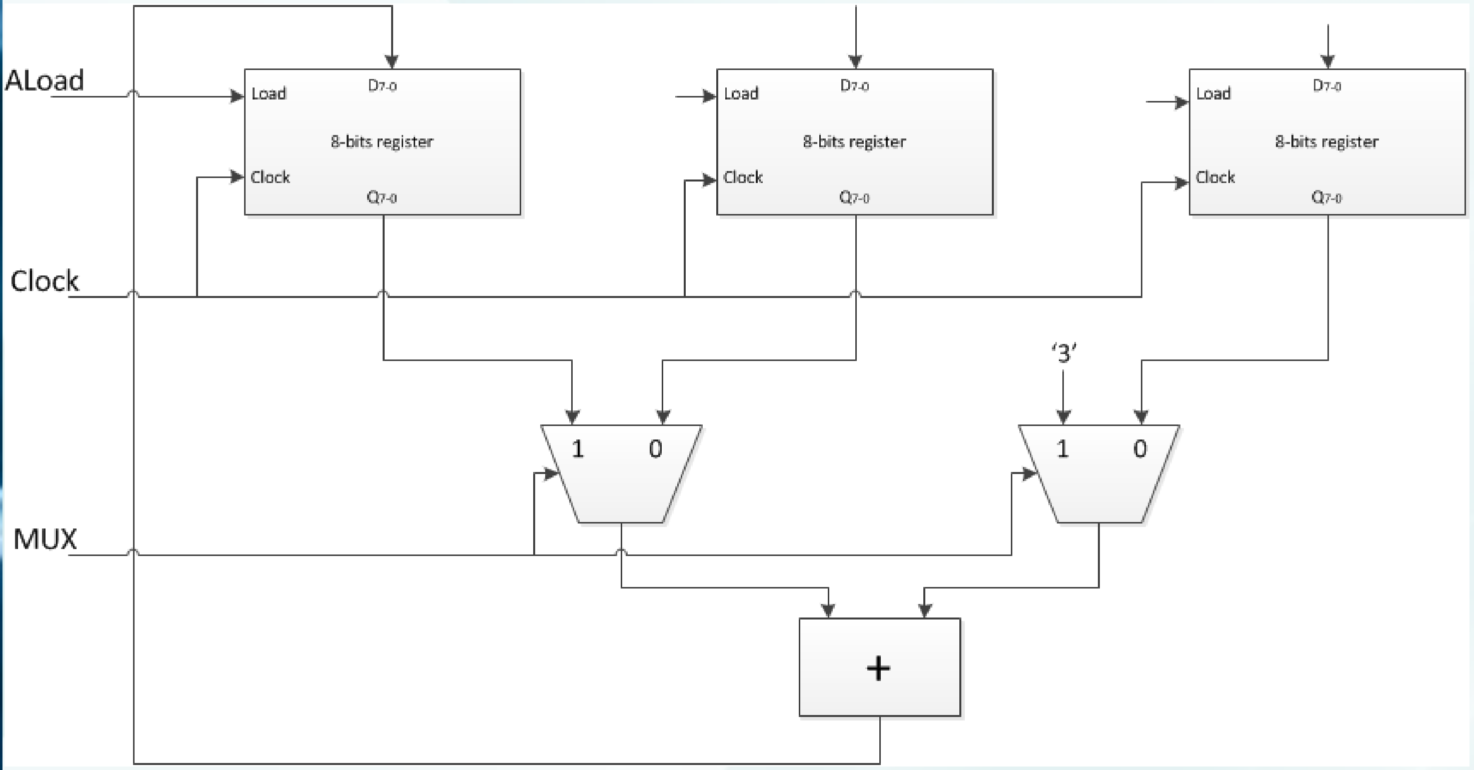
```
`timescale 1ns / 1ps

module datapath(
    input ALoad,
    input BLoad,
    input Muxsel,
     input clock,
    input out_ctrl,
    input [3:0] DinA,
     output Astatus,
     output reg[3:0] DoutB
    );


reg [3:0] A,B;

always @(posedge clock)
begin
    // when ALoad = '1',Muxsel = 'x', BLoad = '0', Out = '0' //
    if(ALoad==1'b1 && BLoad==1'b0 && out_ctrl==1'b0)
    begin
        A = DinA;
    end
    else
    begin
        A = A;
    end
end

assign Astatus = (A == 4'b0101) ? 1'b1:1'b0;

always @(posedge clock)
begin
    // when ALoad = '0',BLoad = '1', Out = '0' //
    if(ALoad==1'b0 && BLoad==1'b1 && out_ctrl==1'b0)
    begin
        if(Muxsel==1'b0) // Muxsel = '0'
        begin
            B = 4'd13;
            //B=4'd7;
        end
        else  // Muxsel = '1'
        begin
```

Datapath

- What is datapath ?
  - is a set of functional units that carry out data required for processing operations.

- The datapath presents several control signals that when activated or deactivated in a given clock cycle allow execution of different RT operations
  - The execution of an RT operation requires activation or deactivation of each control signals
  - The control signals of a datapath are grouped in the so-called **control word**:
    - ✓ A single bit is dedicated to each control signal
    - ✓ Each RT operation is specified by a set of bit in the control word
    - ✓ Combining several control words in a given sequence, the datapath will perform specific operations in a given order, **i.e., execute a program**

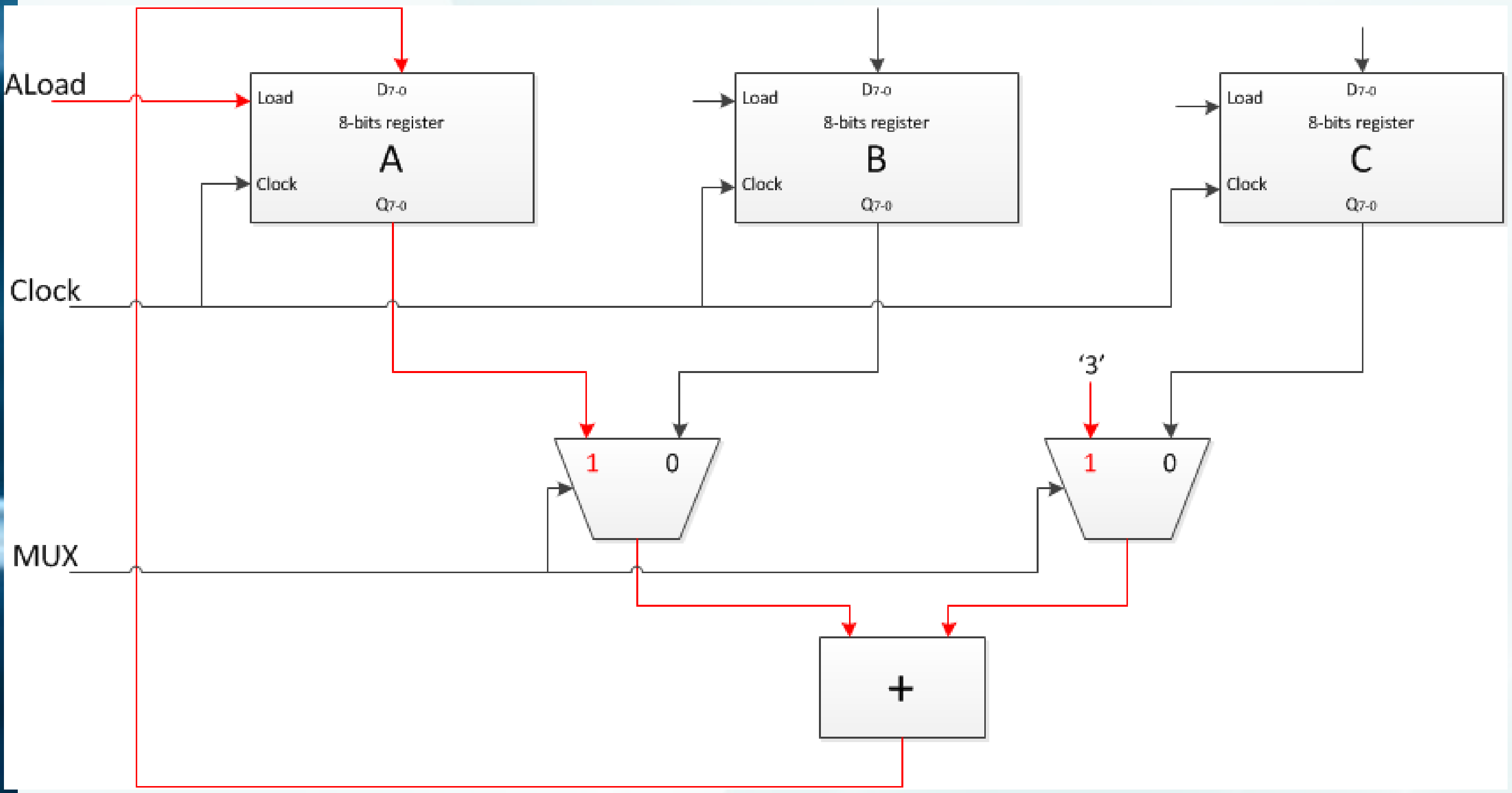
Datapath



Control Word	RT Operation	Control Signal	
		ALoad	Mux
1	A = A + 3	1	1
2	A = B + C	1	0

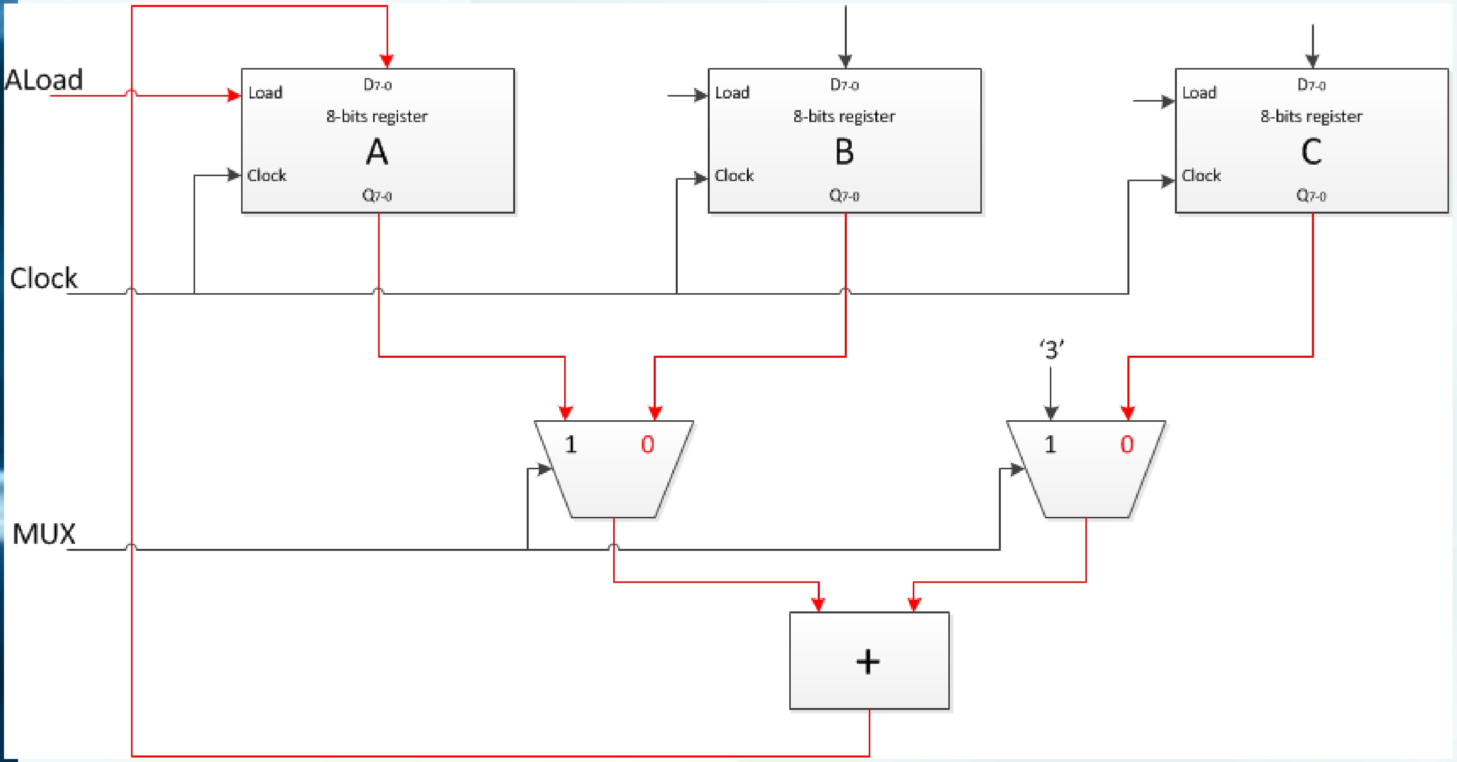
ESRG
Embedded Systems
Research Group

Datapath



Control Word	RT Operation	Control Signal	
		ALoad	Mux
1	A = A + 3	1	1
2	A = B + C	1	0

Datapath



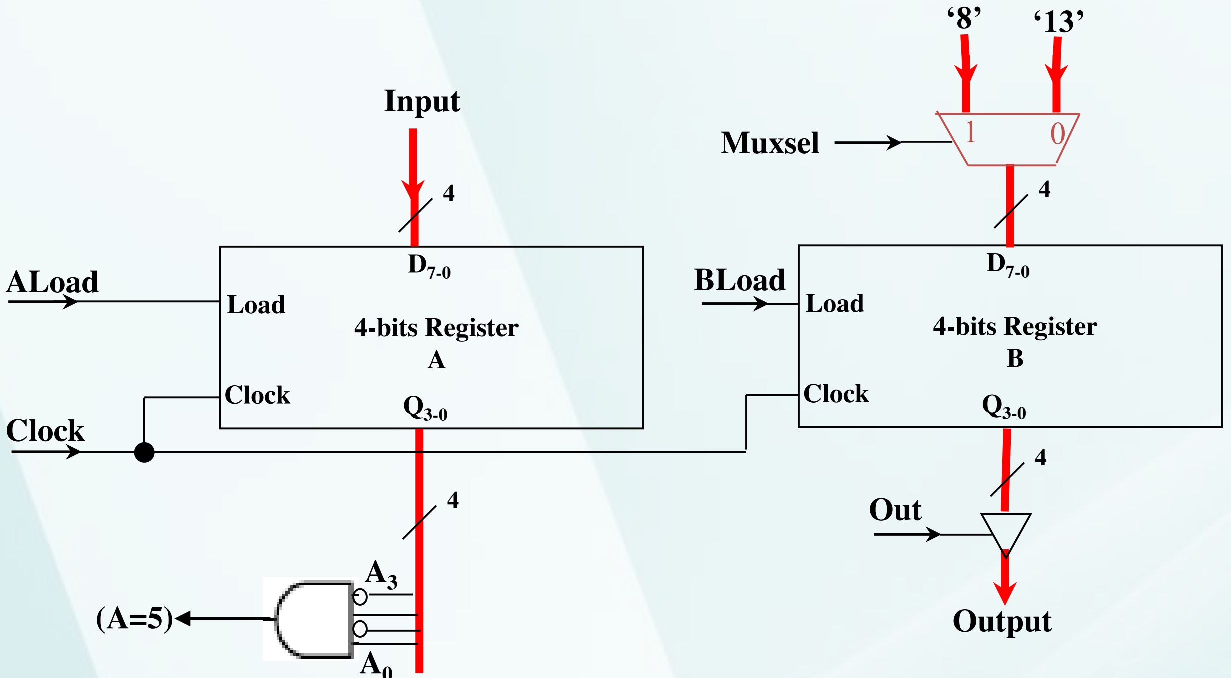
Control Word	RT Operation	Control Signal	
		ALoad	Mux
1	A = A + 3	1	1
2	A = B + C	1	0

Datapath

- Discuss the implementation of a datapath and control unit for the following algorithm:

```
1   INPUT A
2   IF (A = 5) THEN
3       B = 8
4   ELSE
5       B = 13
6   END IF
7   OUTPUT B
```

Datapath



```
1  INPUT A
2  IF (A = 5) THEN
3     B = 8
4  ELSE
5     B = 13
6  END IF
7  OUTPUT B
```

Note that by connecting the comparator output directly to the multiplexer input:

a) The datapath generates one state signal less

b) And it would be necessary one control signal less

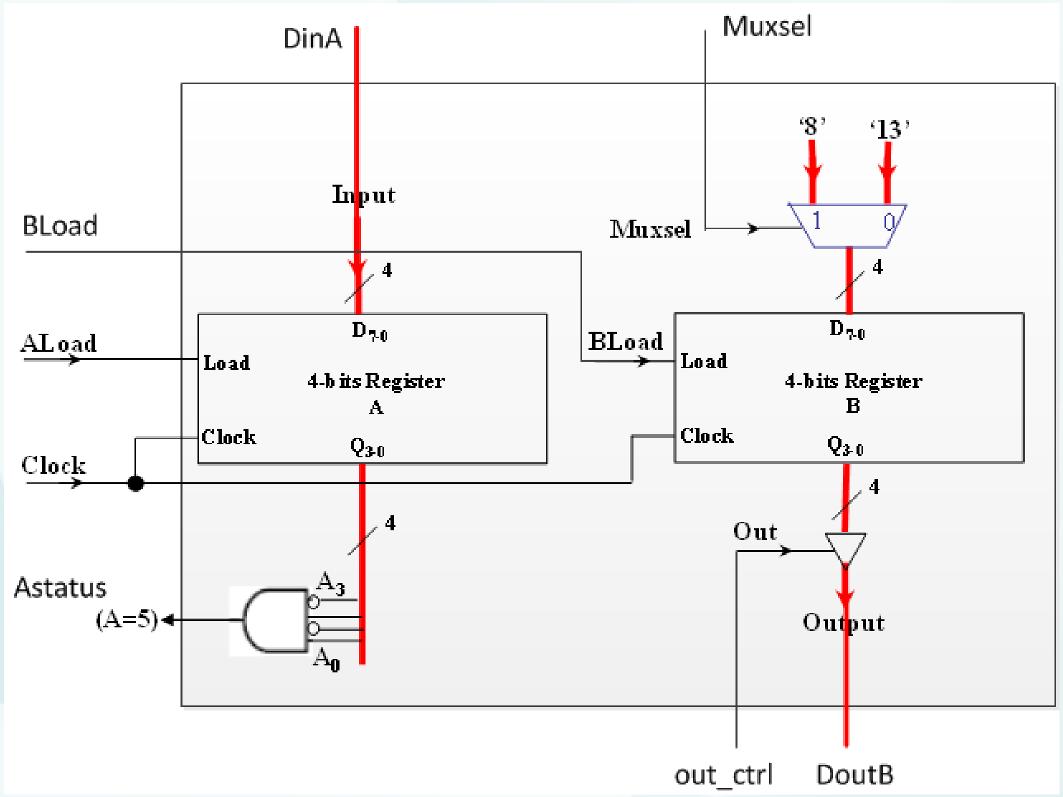
c) However, there are cases where timing problems will arise if state signals are used directly to drive some control signals

Control Word	RT Operation	Control Signals			
		ALoad	Muxsel	BLoad	Out
1	INPUT A	1	x	0	0
2	B = 8	0	1	1	0
3	B = 13	0	0	1	0
4	OUTPUT B	0	x	0	1

ESRG
Embedded Systems
Research Group

Datapath: interface

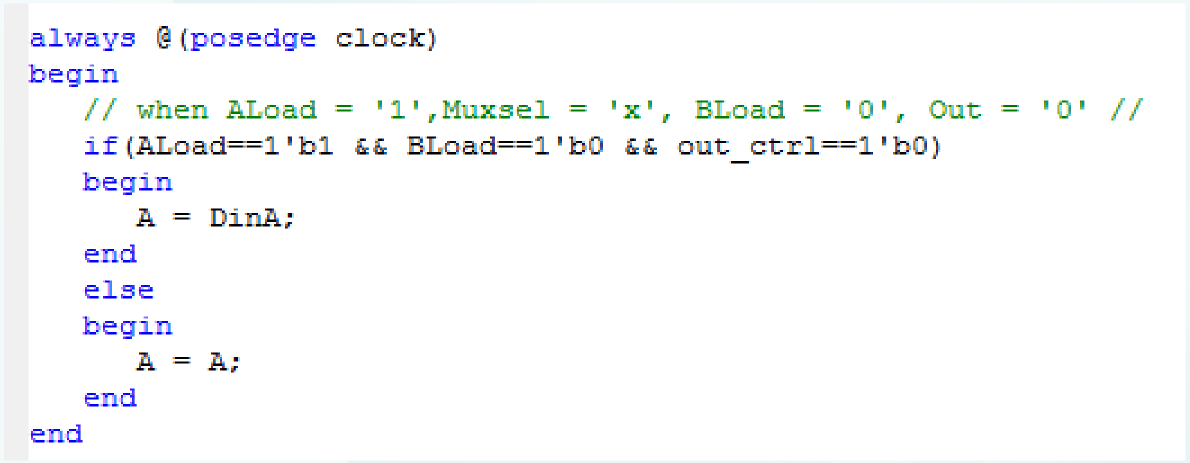
CENTROALGORITMI



How to create in Verilog

```
module datapath(
    input ALoad,
    input BLoad,
    input Muxsel,
    input clock,
    input out_ctrl,
    input [3:0] DinA,
    output Astatus,
    output reg[3:0] DoutB
    );
```

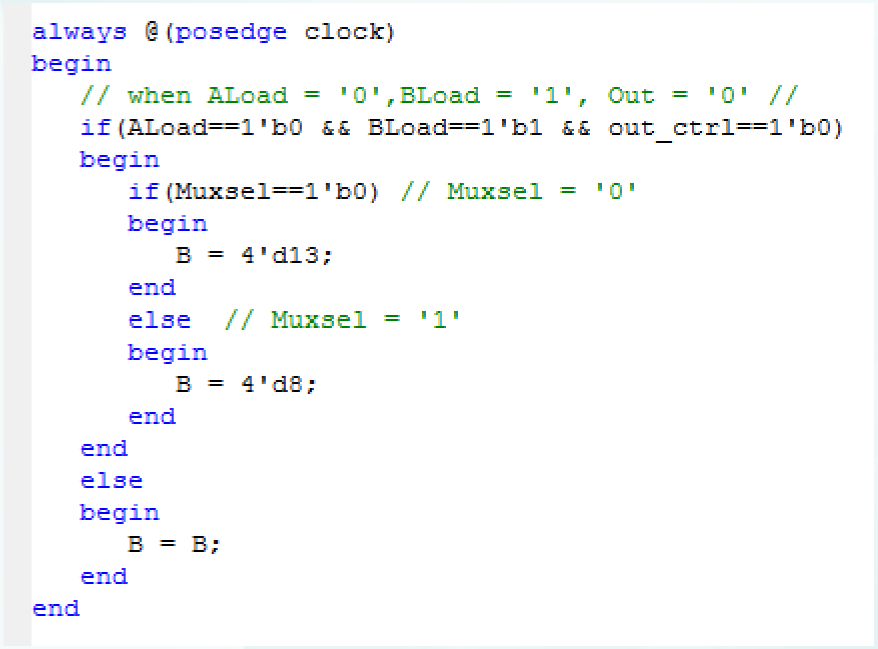
Datapath: internal logic



```
always @(posedge clock)
begin
   // when ALoad = '1',Muxsel = 'x', BLoad = '0', Out = '0' //
   if(ALoad==1'b1 && BLoad==1'b0 && out_ctrl==1'b0)
   begin
      A = DinA;
   end
   else
   begin
      A = A;
   end
end
```

Control Word	RT Operation	Control Signals			
		ALoad	Muxsel	BLoad	Out
1	INPUT A	1	x	0	0
2	B = 8	0	1	1	0
3	B = 13	0	0	1	0
4	OUTPUT B	0	x	0	1

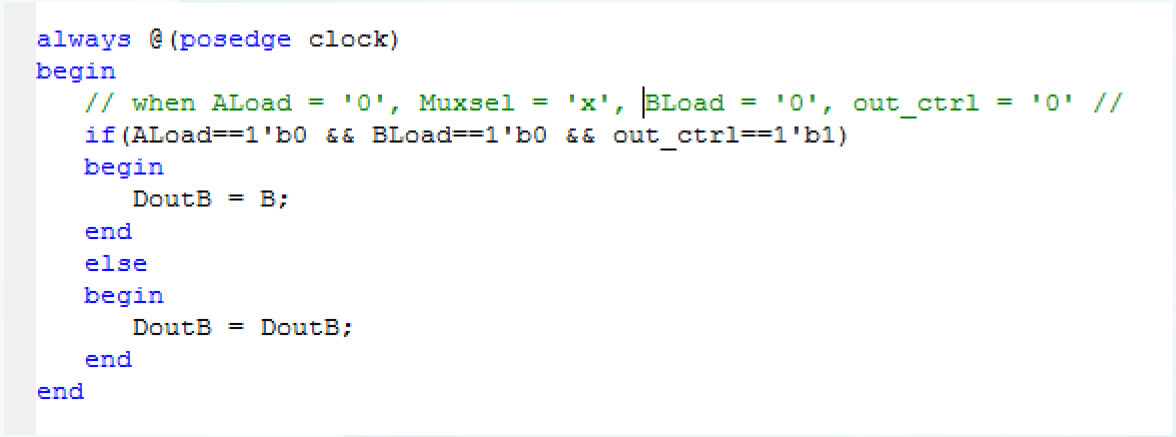
Datapath: internal logic



```
always @(posedge clock)
begin
   // when ALoad = '0',BLoad = '1', Out = '0' //
   if(ALoad==1'b0 && BLoad==1'b1 && out_ctrl==1'b0)
   begin
      if(Muxsel==1'b0) // Muxsel = '0'
      begin
         B = 4'd13;
      end
      else  // Muxsel = '1'
      begin
         B = 4'd8;
      end
   end
   else
   begin
      B = B;
   end
end
```

Control Word	RT Operation	Control Signals			
		ALoad	Muxsel	BLoad	Out
1	INPUT A	1	x	0	0
2	B = 8	0	1	1	0
3	B = 13	0	0	1	0
4	OUTPUT B	0	x	0	1

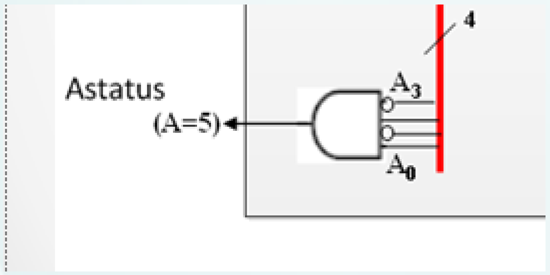
Datapath: internal logic



```
always @(posedge clock)
begin
   // when ALoad = '0', Muxsel = 'x', BLoad = '0', out_ctrl = '0' //
   if(ALoad==1'b0 && BLoad==1'b0 && out_ctrl==1'b1)
   begin
      DoutB = B;
   end
   else
   begin
      DoutB = DoutB;
   end
end
```

Control Word	RT Operation	Control Signals			
		ALoad	Muxsel	BLoad	Out
1	INPUT A	1	x	0	0
2	B = 8	0	1	1	0
3	B = 13	0	0	1	0
4	**OUTPUT B**	**0**	**x**	**0**	**1**

Datapath: internal logic

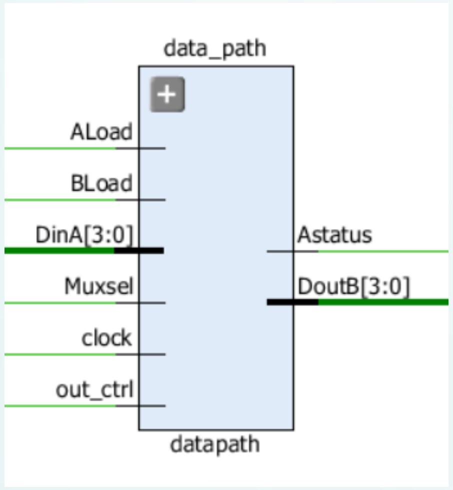


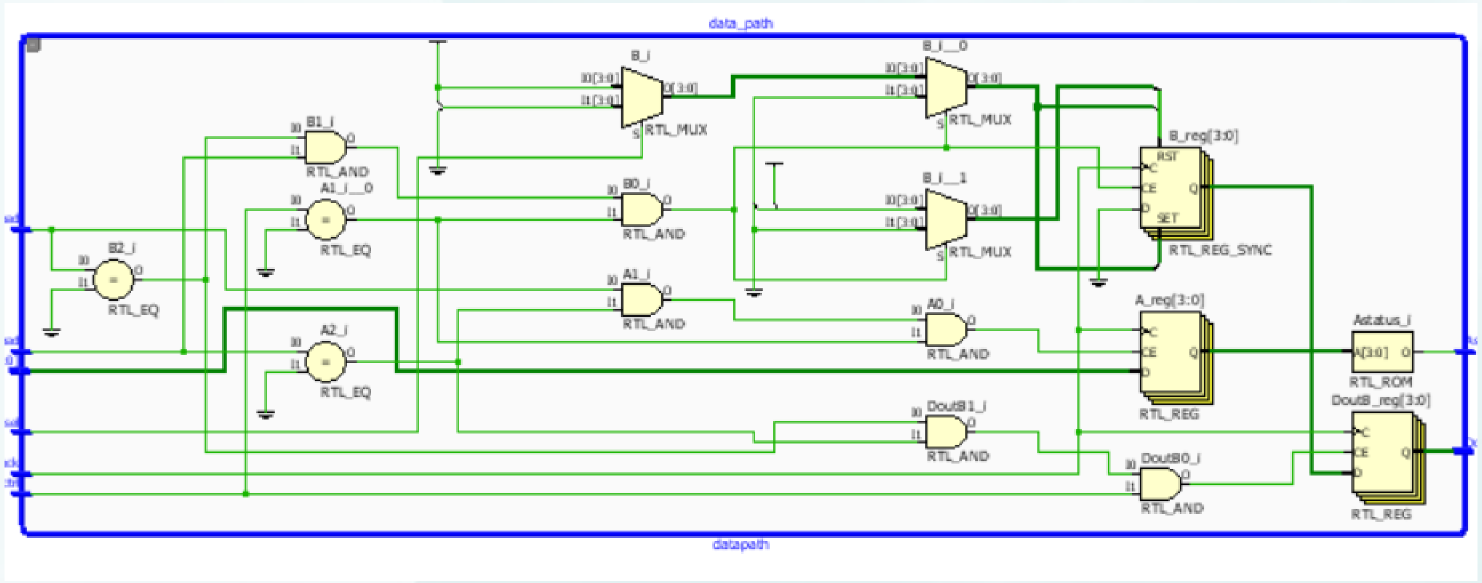
```
assign Astatus = (A == 4'b0101) ? 1'b1:1'b0;
```

ESRG
Embedded Systems
Research Group

Datapath: interface & internal logic

CENTROALGORITMI



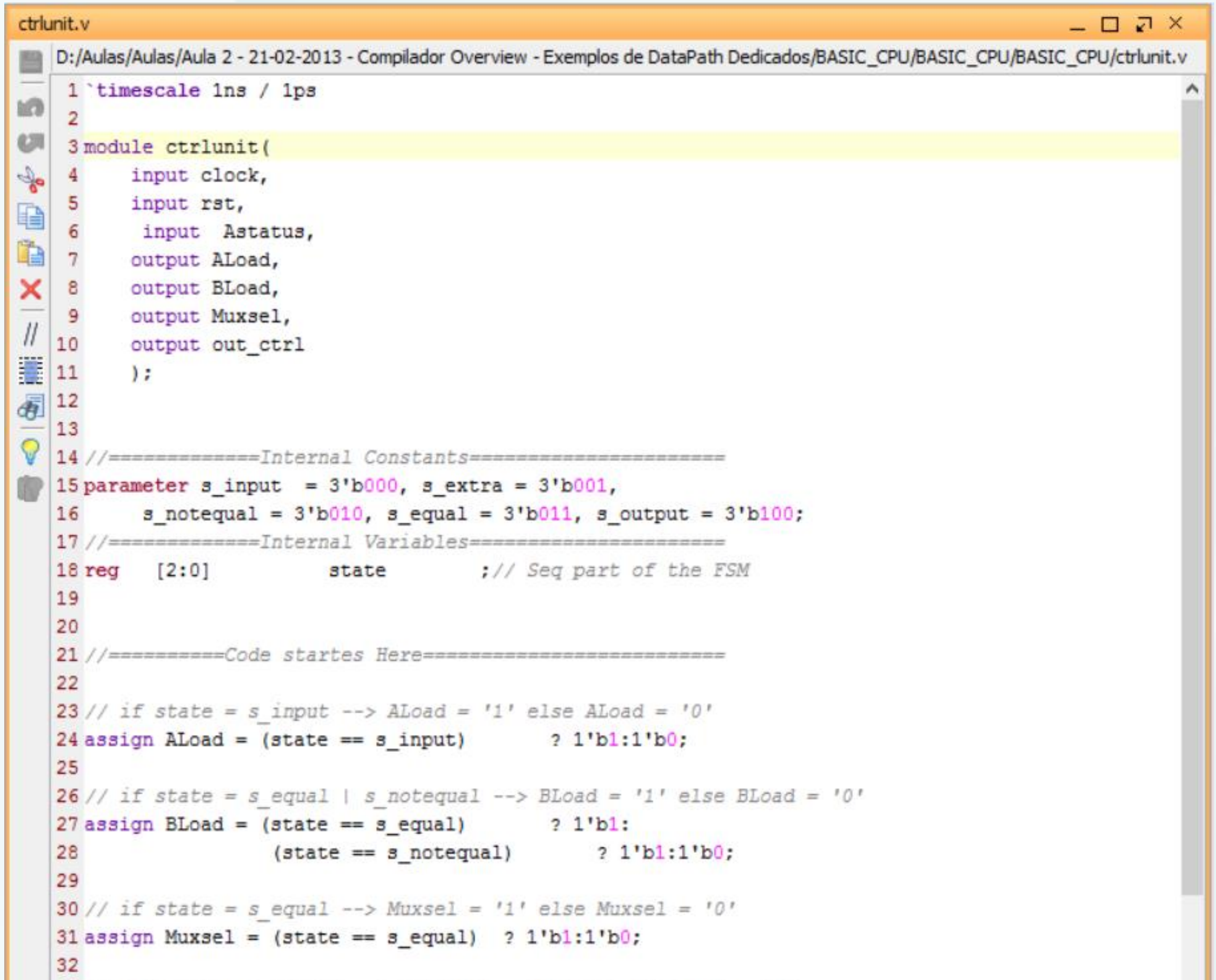


ESRG
Embedded Systems
Research Group

Control unit

CENTROALGORITMI

- Open the ctrlunit.v file and examine it carefully:


ctrlunit.v

D:/Aulas/Aulas/Aula 2 - 21-02-2013 - Compilador Overview - Exemplos de DataPath Dedicados/BASIC_CPU/BASIC_CPU/BASIC_CPU/ctrlunit.v

```
`timescale 1ns / 1ps

module ctrlunit(
    input clock,
    input rst,
     input  Astatus,
    output ALoad,
    output BLoad,
    output Muxsel,
    output out_ctrl
    );


//=============Internal Constants======================
parameter s_input  = 3'b000, s_extra = 3'b001,
     s_notequal = 3'b010, s_equal = 3'b011, s_output = 3'b100;
//=============Internal Variables======================
reg   [2:0]          state        ;// Seq part of the FSM


//==========Code startes Here==========================

// if state = s_input --> ALoad = '1' else ALoad = '0'
assign ALoad = (state == s_input)       ? 1'b1:1'b0;

// if state = s_equal | s_notequal --> BLoad = '1' else BLoad = '0'
assign BLoad = (state == s_equal)       ? 1'b1:
               (state == s_notequal)       ? 1'b1:1'b0;

// if state = s_equal --> Muxsel = '1' else Muxsel = '0'
assign Muxsel = (state == s_equal)  ? 1'b1:1'b0;

```

Control unit

- Control Unit
  - or in other words a state machine

i. Each state corresponds to a control word

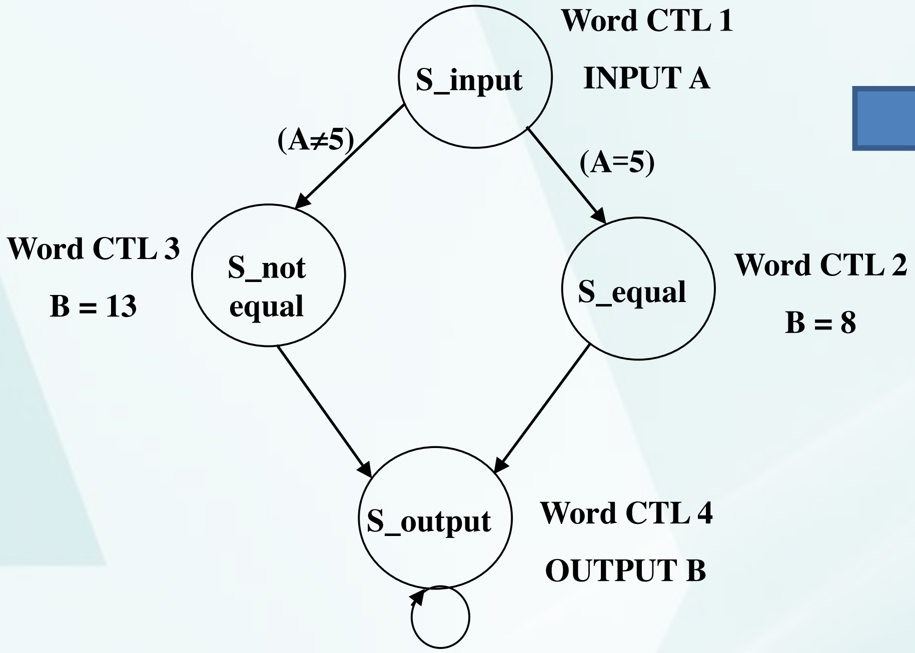
ii. Each state is executed in a clock cycle

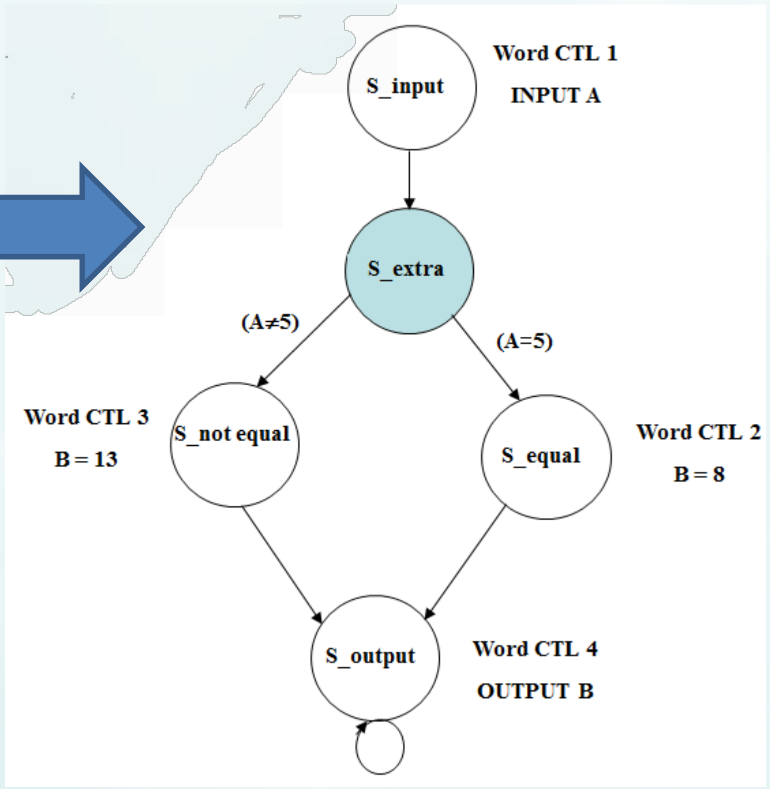
iii. Transitions are dictated by the sequence in which algorithm instructions are executed

iv. Be aware of (**due to implementation with a D flip-flop that changes output at rising edge of the clock**):

- At the rising edge of the clock a register is update with new value if the given signal LOAD is enabled
- At each rising edge of the clock, the FSM enters a new state - **the next state**

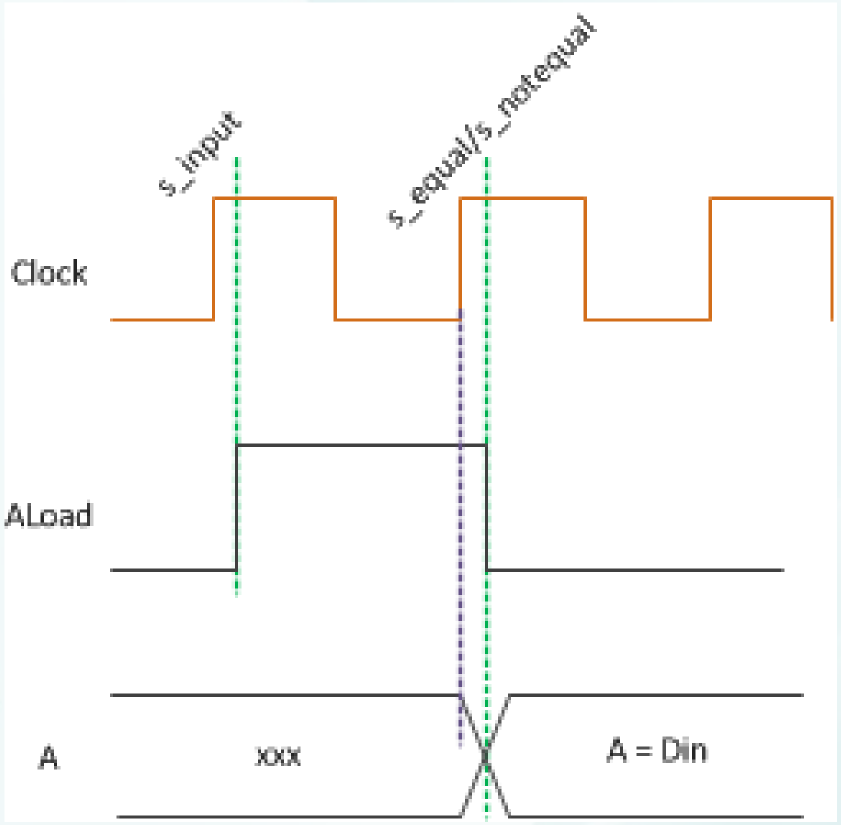
Control unit



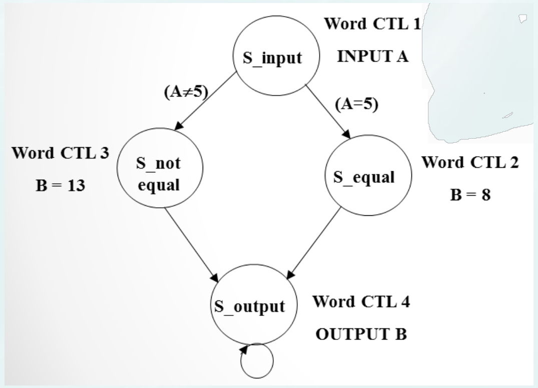


Control unit

- Gate delay

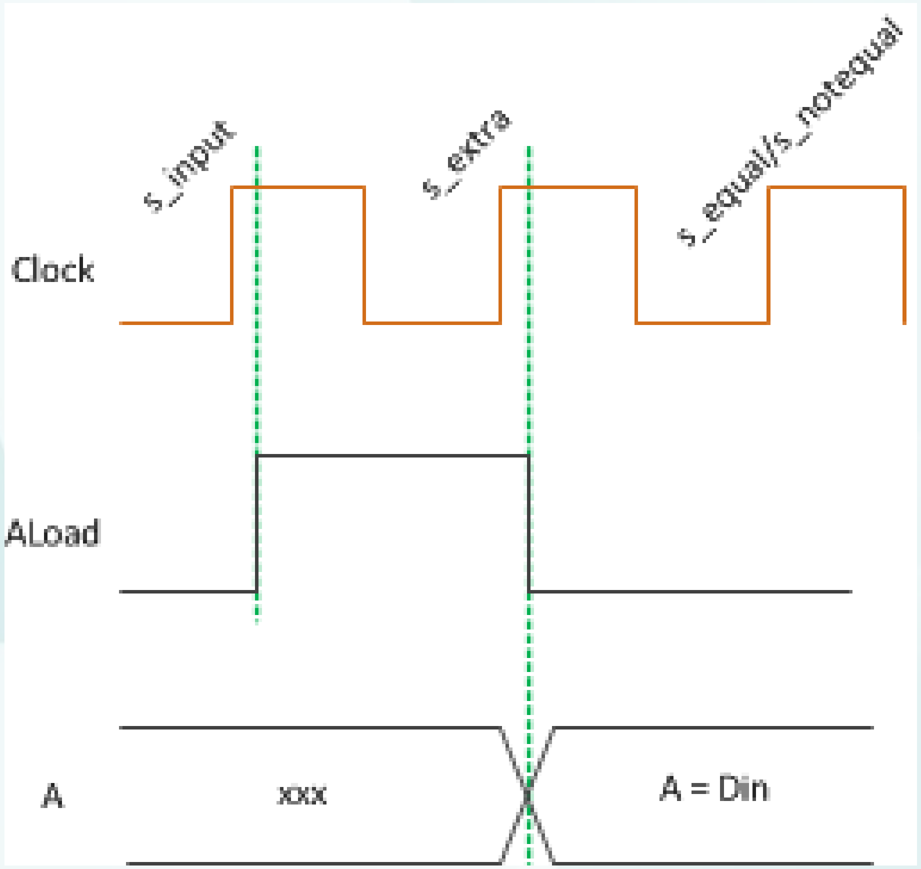


- The control unit want to load data to register A at the first clock
- Aload is not change immediately
- The register A at the second clock is not collect. (it's xxx)

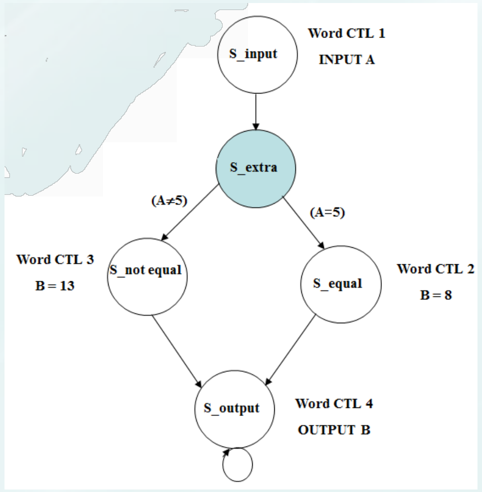


Control unit

- Gate delay



- The control unit want to load data to register A at the first clock
- Aload is not changed immediately



Control unit

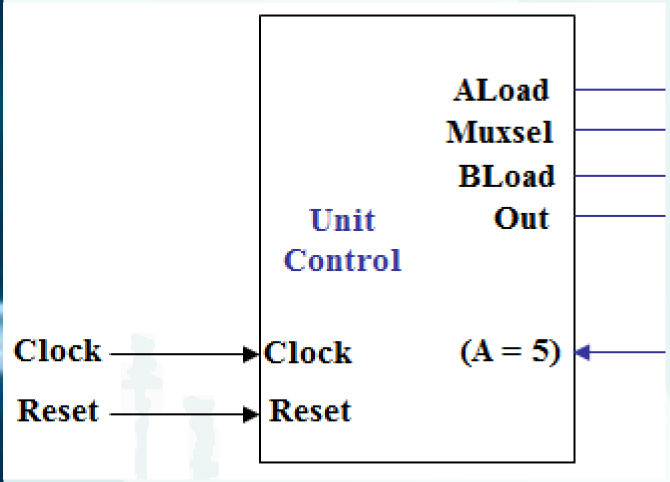
– Control Unit: Building the next state table

i. Three registers will be used to encode the five states. For instance:
   - (000→S_input)
   - (001→S_extra)
   - (010→S_notequal)
   - (011→S_equal)
   - (100→ S_output)

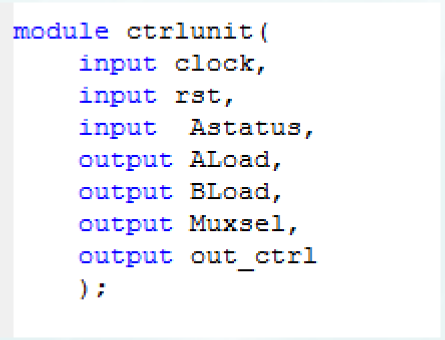
ii. Due to noise in the circuit, the FSM can reach one of the unused states (101, 110 or 111). Therefore, we assign the state S_input as the next state from these states

   - Another solution would be using **don't care values** to simplify the equations of excitement, if it does not matter which is the next state

Control unit: interface



How to create in Verilog



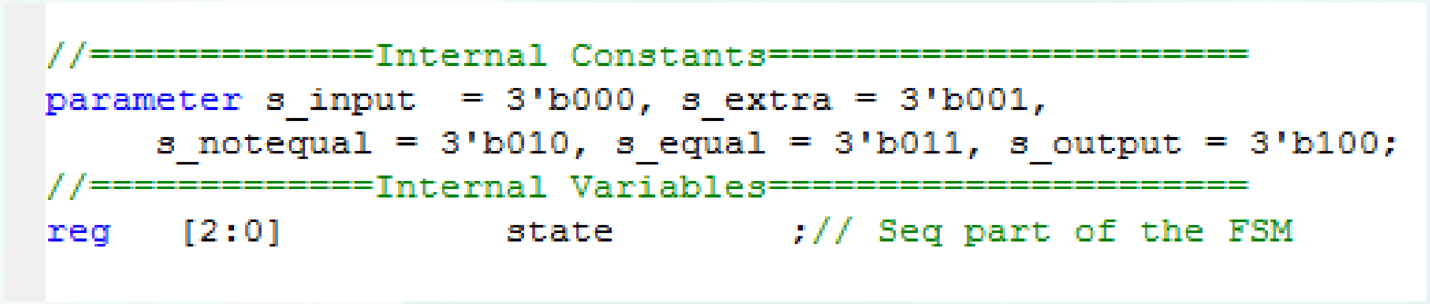
```
module ctrlunit(
    input clock,
    input rst,
    input  Astatus,
    output ALoad,
    output BLoad,
    output Muxsel,
    output out_ctrl
    );
```

Control unit: FSM

- FSM : finite state machine
- Create FSM for control unit
- Assign constant to each state

How to assign binary value to these state in verilog ?

- (000→S_input)
- (001→S_extra)
- (010→S_notequal)
- (011→S_equal)
- (100→ S_output)



```
//=============Internal Constants======================
parameter s_input  = 3'b000, s_extra = 3'b001,
     s_notequal = 3'b010, s_equal = 3'b011, s_output = 3'b100;
//=============Internal Variables======================
reg   [2:0]          state        ;// Seq part of the FSM
```

Control unit

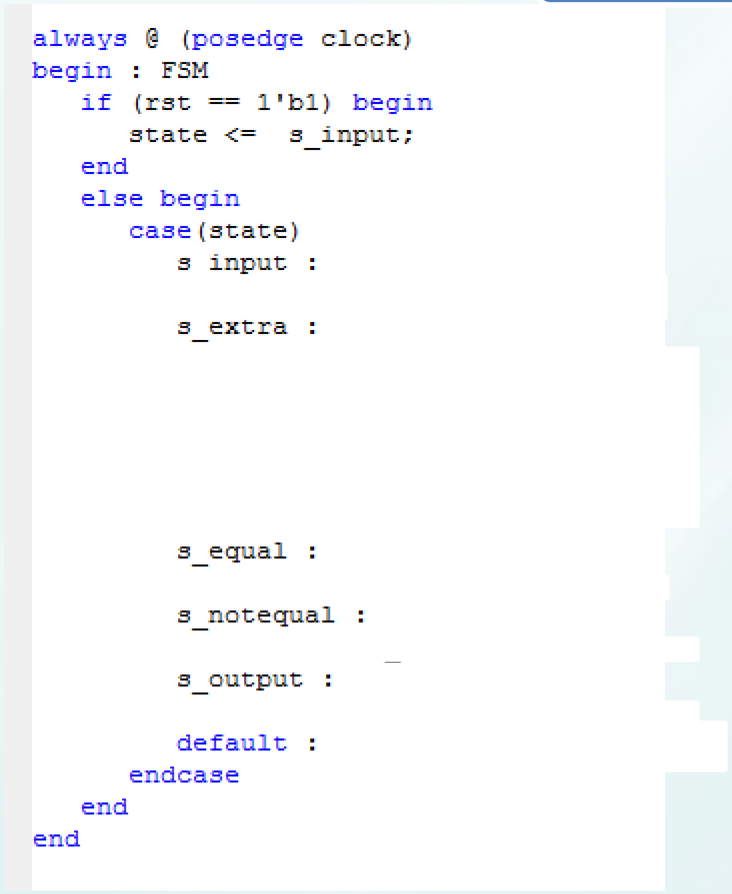
- Coding Verilog follow the table
- Create FSM as sequential statement
- Generate control signal as combination logic

Current State $Q_2Q_1Q_0$	Next State $Q_{2next}Q_{1next}Q_{0next}$	
	(A=5)'	(A=5)
(S_input, 000)	(S_extra, 001)	(S_extra, 001)
(S_extra, 001)	(S_notequal, 010)	(S_equal, 011)
(S_notequal, 010)	(S_output, 100)	(S_output, 100)
(S_equal, 011)	(S_output, 100)	(S_output, 100)
(S_output, 100)	(S_output, 100)	(S_output, 100)
(unused, 101)	(S_input, 000)	(S_input, 000)
(unused, 110)	(S_input, 000)	(S_input, 000)
(unused, 111)	(S_input, 000)	(S_input, 000)

Control unit: next state

FSM's next state

Current State $Q_2Q_1Q_0$	Next State $Q_{2next}Q_{1next}Q_{0next}$	
	(A=5)'	(A=5)
(S_input, 000)	(S_extra, 001)	(S_extra, 001)
(S_extra, 001)	(S_notequal, 010)	(S_equal, 011)
(S_notequal, 010)	(S_output, 100)	(S_output, 100)
(S_equal, 011)	(S_output, 100)	(S_output, 100)
(S_output, 100)	(S_output, 100)	(S_output, 100)
(unused, 101)	(S_input, 000)	(S_input, 000)
(unused, 110)	(S_input, 000)	(S_input, 000)
(unused, 111)	(S_input, 000)	(S_input, 000)



```
always @ (posedge clock)
begin : FSM
   if (rst == 1'b1) begin
      state <=  s_input;
   end
   else begin
      case(state)
         s input :

         s_extra :









         s_equal :

         s_notequal :

         s_output :

         default :
      endcase
   end
end
```

ESRG
Embedded Systems
Research Group

Control unit: next state

CENTROALGORITMI

FSM's next state

Current State $Q_2Q_1Q_0$	Next State $Q_{2next}Q_{1next}Q_{0next}$	
	(A=5)'	(A=5)
(S_input, 000)	(S_extra, 001)	(S_extra, 001)
(S_extra, 001)	(S_notequal, 010)	(S_equal, 011)
(S_notequal, 010)	(S_output, 100)	(S_output, 100)
(S_equal, 011)	(S_output, 100)	(S_output, 100)
(S_output, 100)	(S_output, 100)	(S_output, 100)
(unused, 101)	(S_input, 000)	(S_input, 000)
(unused, 110)	(S_input, 000)	(S_input, 000)
(unused, 111)	(S_input, 000)	(S_input, 000)

```
always @ (posedge clock)
begin : FSM
   if (rst == 1'b1) begin
      state <=  s_input;
   end
   else begin
      case(state)
         s_input :
            state <= s_extra;
         s_extra :




         s_equal :

         s_notequal :

         s_output :

         default :
      endcase
   end
end
```

ESRG
Embedded Systems
Research Group

Control unit: next state

CENTROALGORITMI

Current State $Q_2Q_1Q_0$	Next State $Q_{2next}Q_{1next}Q_{0next}$	
	(A=5)’	(A=5)
(S_input, 000)	(S_extra, 001)	(S_extra, 001)
(S_extra, 001)	(S_notequal, 010)	(S_equal, 011)
(S_notequal, 010)	(S_output, 100)	(S_output, 100)
(S_equal, 011)	(S_output, 100)	(S_output, 100)
(S_output, 100)	(S_output, 100)	(S_output, 100)
(unused, 101)	(S_input, 000)	(S_input, 000)
(unused, 110)	(S_input, 000)	(S_input, 000)
(unused, 111)	(S_input, 000)	(S_input, 000)

FSM’s next state

```
always @ (posedge clock)
begin : FSM
   if (rst == 1'b1) begin
      state <=  s_input;
   end
   else begin
      case(state)
         s_input :
            state <= s_extra;
         s_extra :
            if (Astatus==1'b1) begin
               state <= s_equal;
            end
            else begin
               state <= s_notequal;
            end
         s_equal :

         s_notequal :

         s_output :

         default :
      endcase
   end
end
```

ESRG
Embedded Systems
Research Group

Control unit: next state

CENTROALGORITMI

Current State $Q_2Q_1Q_0$	Next State $Q_{2next}Q_{1next}Q_{0next}$	
	(A=5)'	(A=5)
(S_input, 000)	(S_extra, 001)	(S_extra, 001)
(S_extra, 001)	(S_notequal, 010)	(S_equal, 011)
(S_notequal, 010)	(S_output, 100)	(S_output, 100)
(S_equal, 011)	(S_output, 100)	(S_output, 100)
(S_output, 100)	(S_output, 100)	(S_output, 100)
(unused, 101)	(S_input, 000)	(S_input, 000)
(unused, 110)	(S_input, 000)	(S_input, 000)
(unused, 111)	(S_input, 000)	(S_input, 000)

FSM's next state

```
always @ (posedge clock)
begin : FSM
   if (rst == 1'b1) begin
      state <=  s_input;
   end
   else begin
      case(state)
         s_input :
            state <= s_extra;
         s_extra :
            if (Astatus==1'b1) begin
               state <= s_equal;
            end
            else begin
               state <= s_notequal;
            end
         s_equal :
            state <= s_output;
         s_notequal :
            state <= s_output;
         s_output :

         default :
      endcase
   end
end
```

ESRG
Embedded Systems
Research Group

Control unit: next state

CENTROALGORITMI

Current State $Q_2Q_1Q_0$	Next State $Q_{2next}Q_{1next}Q_{0next}$	
	(A=5)'	(A=5)
(S_input, 000)	(S_extra, 001)	(S_extra, 001)
(S_extra, 001)	(S_notequal, 010)	(S_equal, 011)
(S_notequal, 010)	(S_output, 100)	(S_output, 100)
(S_equal, 011)	(S_output, 100)	(S_output, 100)
(S_output, 100)	(S_output, 100)	(S_output, 100)
(unused, 101)	(S_input, 000)	(S_input, 000)
(unused, 110)	(S_input, 000)	(S_input, 000)
(unused, 111)	(S_input, 000)	(S_input, 000)

FSM's next state

```
always @ (posedge clock)
begin : FSM
   if (rst == 1'b1) begin
      state <=  s_input;
   end
   else begin
      case(state)
         s_input :
            state <= s_extra;
         s_extra :
            if (Astatus==1'b1) begin
               state <= s_equal;
            end
            else begin
               state <= s_notequal;
            end
         s_equal :
            state <= s_output;
         s_notequal :
            state <= s_output;
         s_output :
            state <= s_output;
         default :
      endcase
   end
end
```

Control unit: FSM output

- How to implement a control unit output signals in Verilog

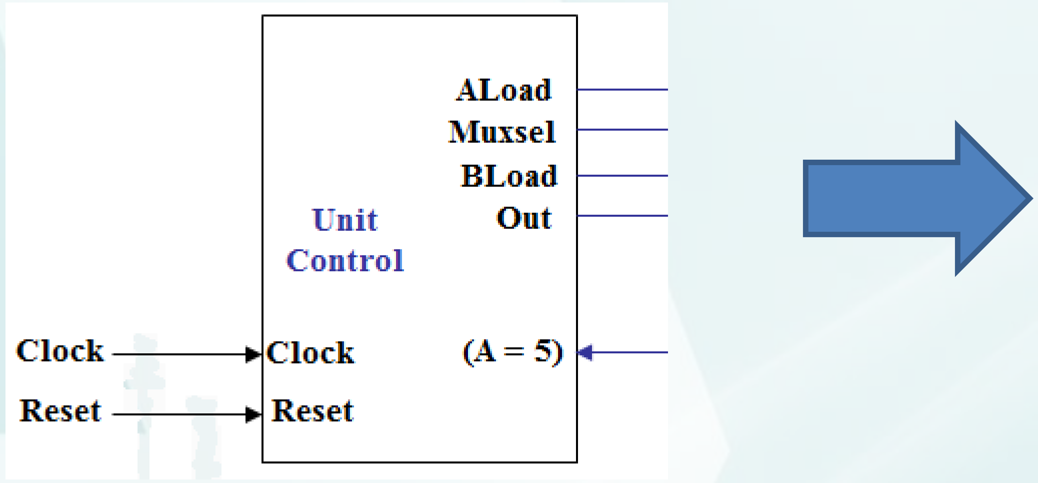
FSM's output

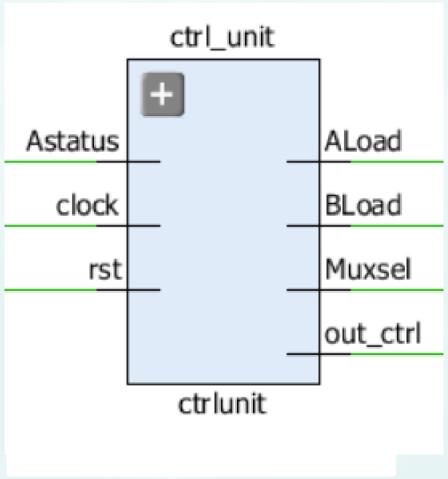
$Q_2Q_1Q_0$	RT Operation	Control Signal			
		ALoad	Muxsel	BLoad	Out
000	INPUT A	1	x	0	0
001	NOP	0	x	0	0
010	B = 8	0	1	1	0
011	B = 13	0	0	1	0
100	OUTPUT B	0	x	0	1
101	NOP	0	x	0	0
110	NOP	0	x	0	0
111	NOP	0	x	0	0

```verilog
// if state = s_input --> ALoad = '1' else ALoad = '0'
assign ALoad = (state == s_input)       ? 1'b1:1'b0;

// if state = s_equal | s_notequal --> BLoad = '1' else BLoad = '0
assign BLoad = (state == s_equal)       ? 1'b1:
               (state == s_notequal)    ? 1'b1:1'b0;

// if state = s_equal --> Muxsel = '1' else Muxsel = '0'
assign Muxsel = (state == s_equal)  ? 1'b1:1'b0;

// if state = s_output --> out_ctrl = '1' else out_ctrl = '0'
assign out_ctrl = (state == s_output)  ? 1'b1:1'b0;
```

Control unit: interface

- After synthesizing, we got the control unit module
- What are the expected synthesizer outputs ?

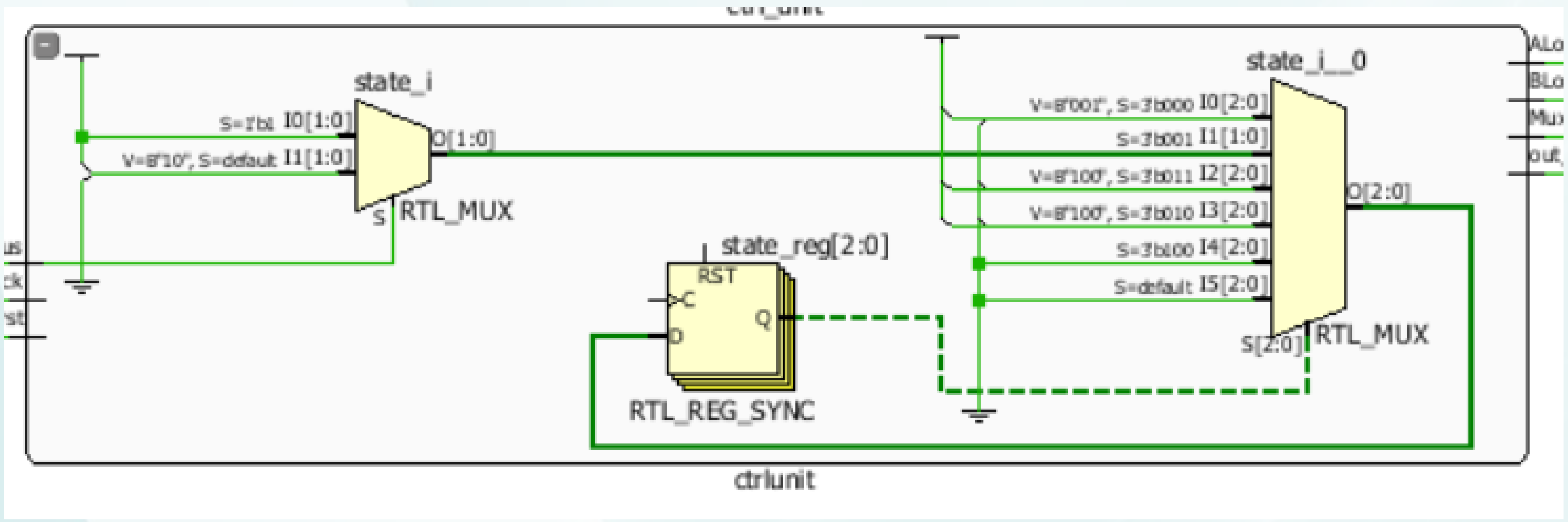




ESRG
Embedded Systems
Research Group

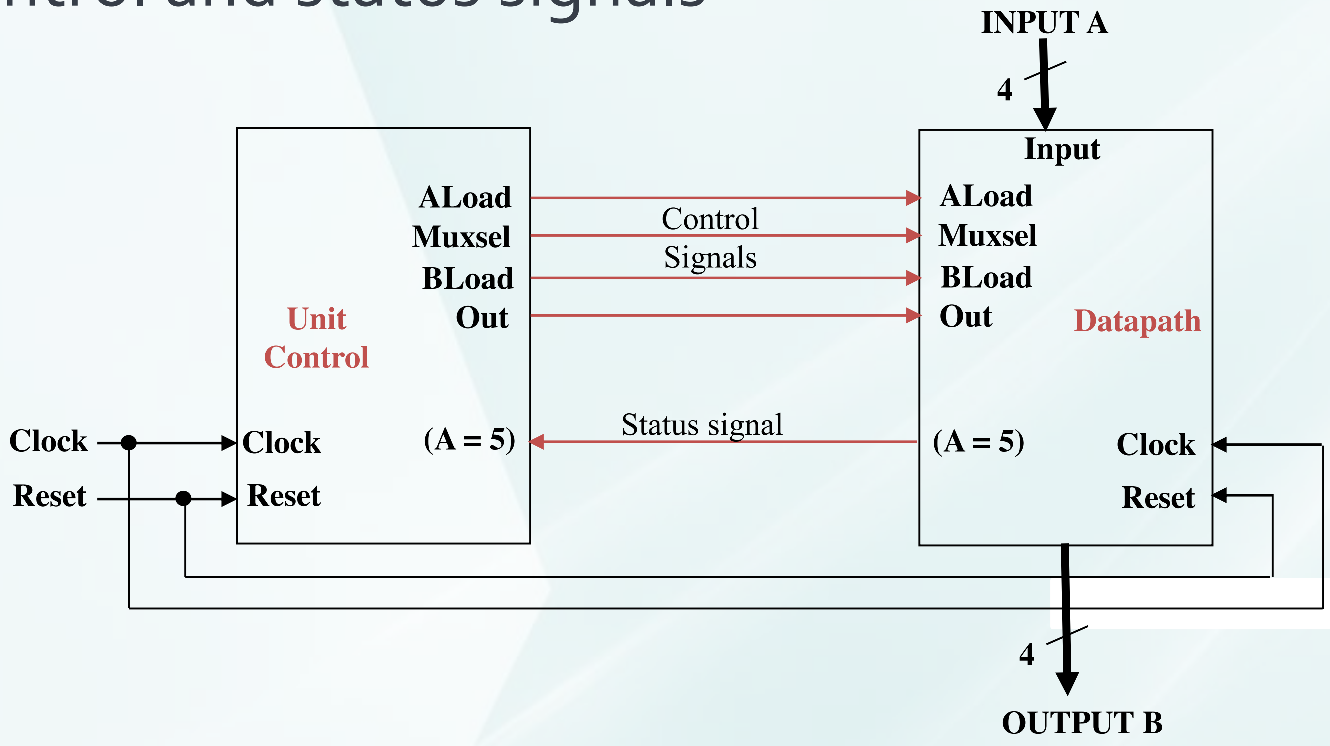
Control unit: internal logic

CENTROALGORITMI



Control unit + datapath

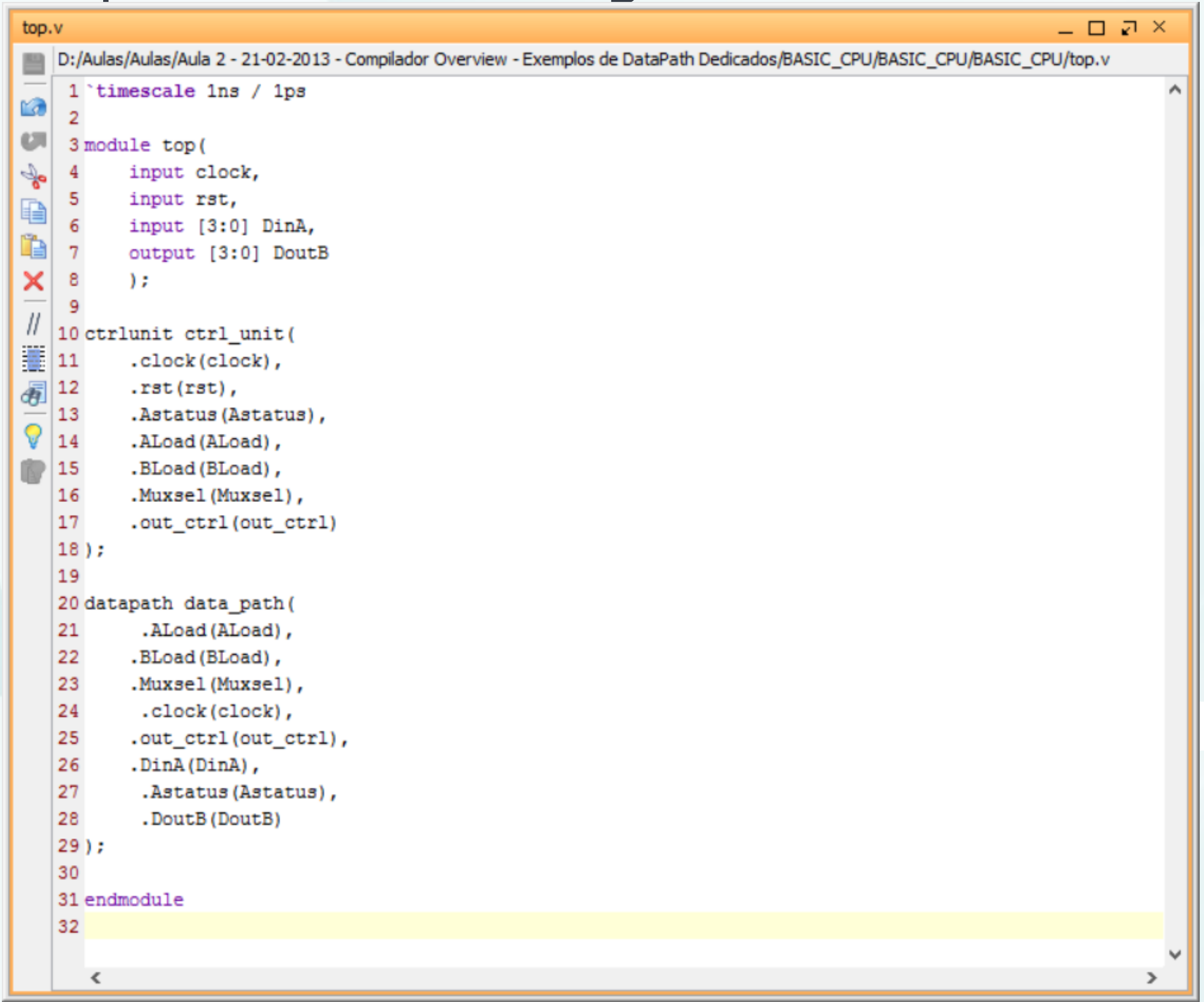
- Finally, the microprocessor will be build by connecting the control unit to the datapath through control and status signals



ESRG
Embedded Systems
Research Group

Control unit + datapath

- Map two modules together:



top.v

D:/Aulas/Aulas/Aula 2 - 21-02-2013 - Compilador Overview - Exemplos de DataPath Dedicados/BASIC_CPU/BASIC_CPU/BASIC_CPU/top.v

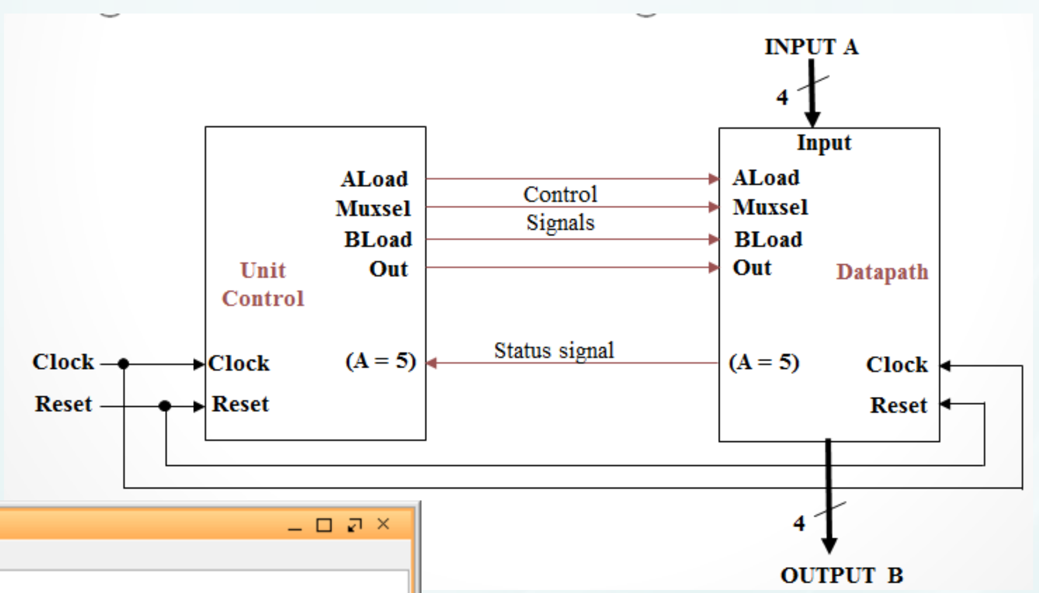
```
`timescale 1ns / 1ps

module top(
    input clock,
    input rst,
    input [3:0] DinA,
    output [3:0] DoutB
    );

ctrlunit ctrl_unit(
    .clock(clock),
    .rst(rst),
    .Astatus(Astatus),
    .ALoad(ALoad),
    .BLoad(BLoad),
    .Muxsel(Muxsel),
    .out_ctrl(out_ctrl)
);

datapath data_path(
     .ALoad(ALoad),
    .BLoad(BLoad),
    .Muxsel(Muxsel),
     .clock(clock),
    .out_ctrl(out_ctrl),
    .DinA(DinA),
     .Astatus(Astatus),
     .DoutB(DoutB)
);

endmodule
```

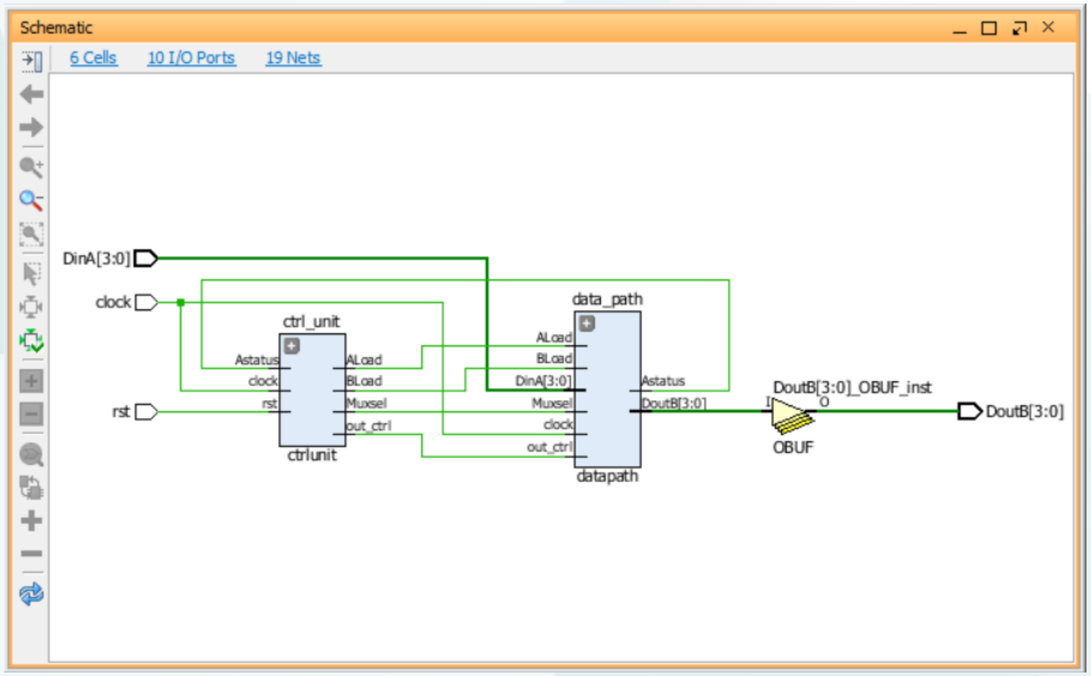
ESRG
Embedded Systems
Research Group

Control unit + datapath

CENTROALGORITMI

- Result:



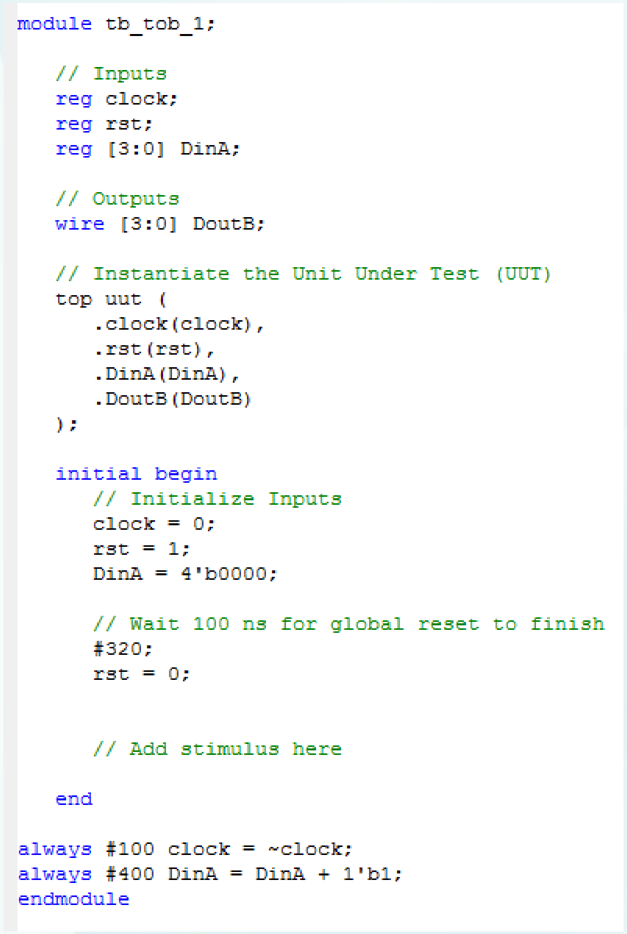


ESRG
Embedded Systems
Research Group

Simulation

CENTROALGORITMI

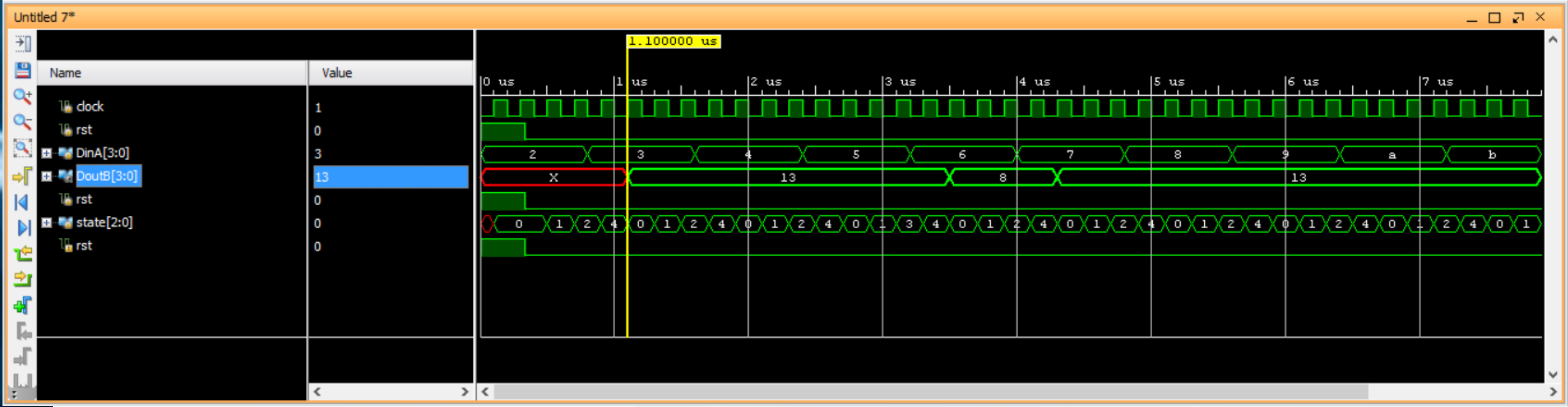
- Add the following testbench to your project and then verify its behavior:



```
module tb_tob_1;

   // Inputs
   reg clock;
   reg rst;
   reg [3:0] DinA;

   // Outputs
   wire [3:0] DoutB;

   // Instantiate the Unit Under Test (UUT)
   top uut (
      .clock(clock),
      .rst(rst),
      .DinA(DinA),
      .DoutB(DoutB)
   );

   initial begin
      // Initialize Inputs
      clock = 0;
      rst = 1;
      DinA = 4'b0000;

      // Wait 100 ns for global reset to finish
      #320;
      rst = 0;


      // Add stimulus here

   end

always #100 clock = ~clock;
always #400 DinA = DinA + 1'b1;
endmodule
```

Simulation

- Run simulation and observe the outputted waveform:



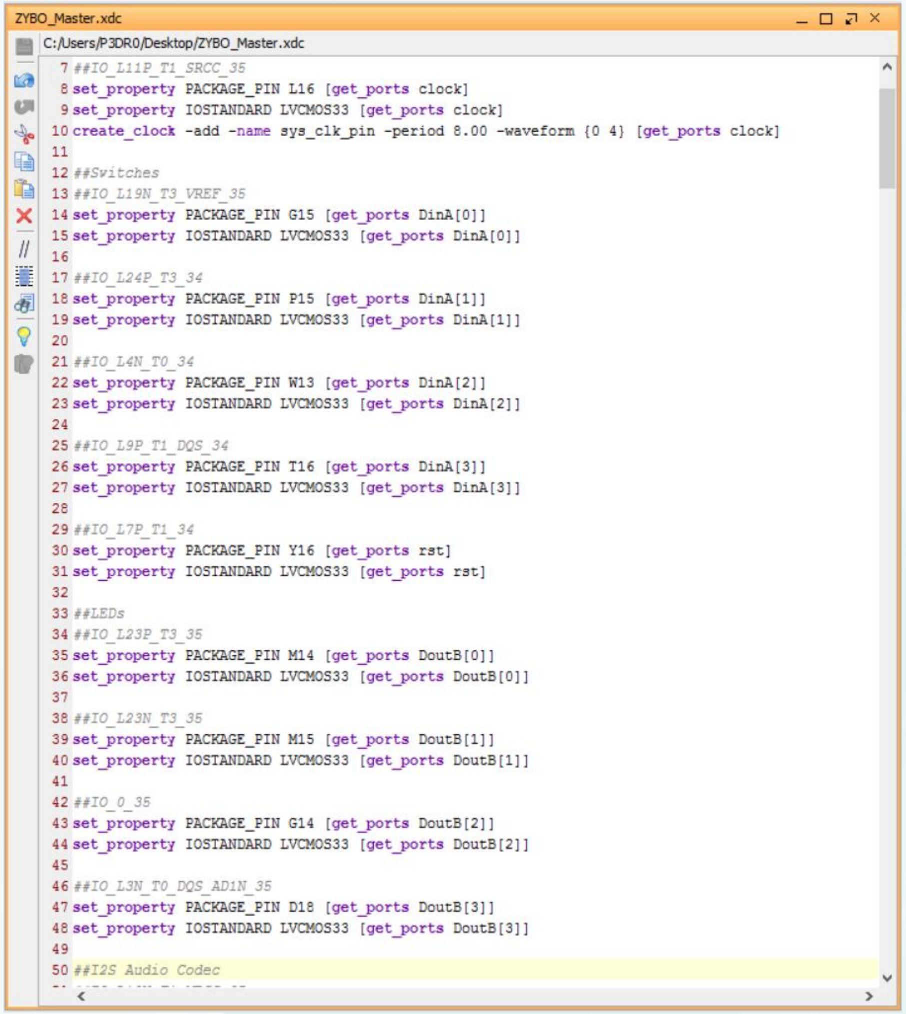
Name	Value
clock	1
rst	0
DinA[3:0]	3
DoutB[3:0]	13
rst	0
state[2:0]	0
rst	0

ESRG
Embedded Systems
Research Group

Xilinx description file

CENTROALGORITMI

- Check the *.xdc file:



ZYBO_Master.xdc

C:/Users/P3DR0/Desktop/ZYBO_Master.xdc

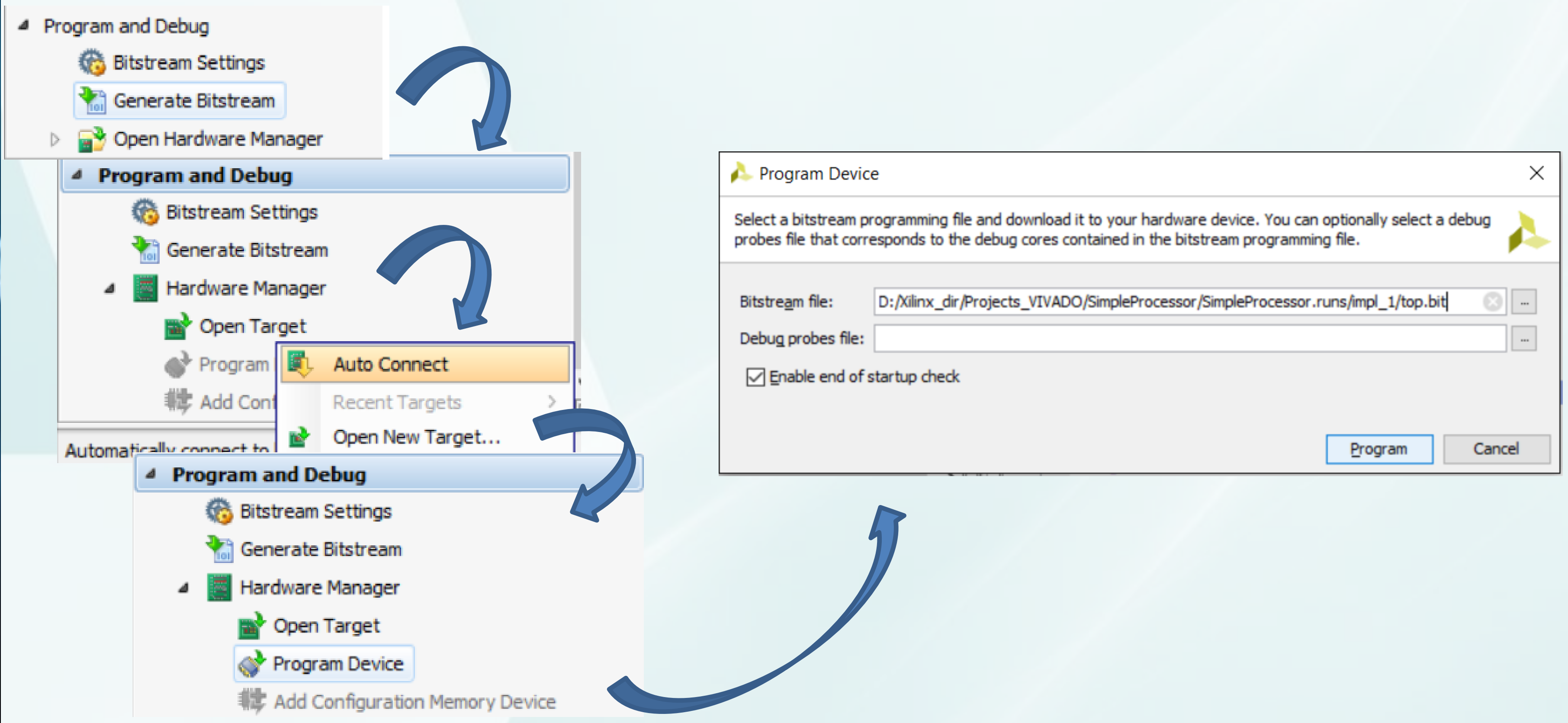
```
##IO_L11P_T1_SRCC_35
set_property PACKAGE_PIN L16 [get_ports clock]
set_property IOSTANDARD LVCMOS33 [get_ports clock]
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports clock]

##Switches
##IO_L19N_T3_VREF_35
set_property PACKAGE_PIN G15 [get_ports DinA[0]]
set_property IOSTANDARD LVCMOS33 [get_ports DinA[0]]

##IO_L24P_T3_34
set_property PACKAGE_PIN P15 [get_ports DinA[1]]
set_property IOSTANDARD LVCMOS33 [get_ports DinA[1]]

##IO_L4N_T0_34
set_property PACKAGE_PIN W13 [get_ports DinA[2]]
set_property IOSTANDARD LVCMOS33 [get_ports DinA[2]]

##IO_L9P_T1_DQS_34
set_property PACKAGE_PIN T16 [get_ports DinA[3]]
set_property IOSTANDARD LVCMOS33 [get_ports DinA[3]]

##IO_L7P_T1_34
set_property PACKAGE_PIN Y16 [get_ports rst]
set_property IOSTANDARD LVCMOS33 [get_ports rst]

##LEDs
##IO_L23P_T3_35
set_property PACKAGE_PIN M14 [get_ports DoutB[0]]
set_property IOSTANDARD LVCMOS33 [get_ports DoutB[0]]

##IO_L23N_T3_35
set_property PACKAGE_PIN M15 [get_ports DoutB[1]]
set_property IOSTANDARD LVCMOS33 [get_ports DoutB[1]]

##IO_0_35
set_property PACKAGE_PIN G14 [get_ports DoutB[2]]
set_property IOSTANDARD LVCMOS33 [get_ports DoutB[2]]

##IO_L3N_T0_DQS_AD1N_35
set_property PACKAGE_PIN D18 [get_ports DoutB[3]]
set_property IOSTANDARD LVCMOS33 [get_ports DoutB[3]]

##I2S Audio Codec
```

ESRG
Embedded Systems
Research Group

Bitstream

CENTROALGORITMI

- Create the bitstream and burn it into your Zybo device;
- Check the functionality on the hardware;



Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file:	D:/Xilinx_dir/Projects_VIVADO/SimpleProcessor/SimpleProcessor.runs/impl_1/top.bit
Debug probes file:	

☑ Enable end of startup check

ESRG
Embedded Systems
Research Group

Any doubts ????

CENTROALGORITMI