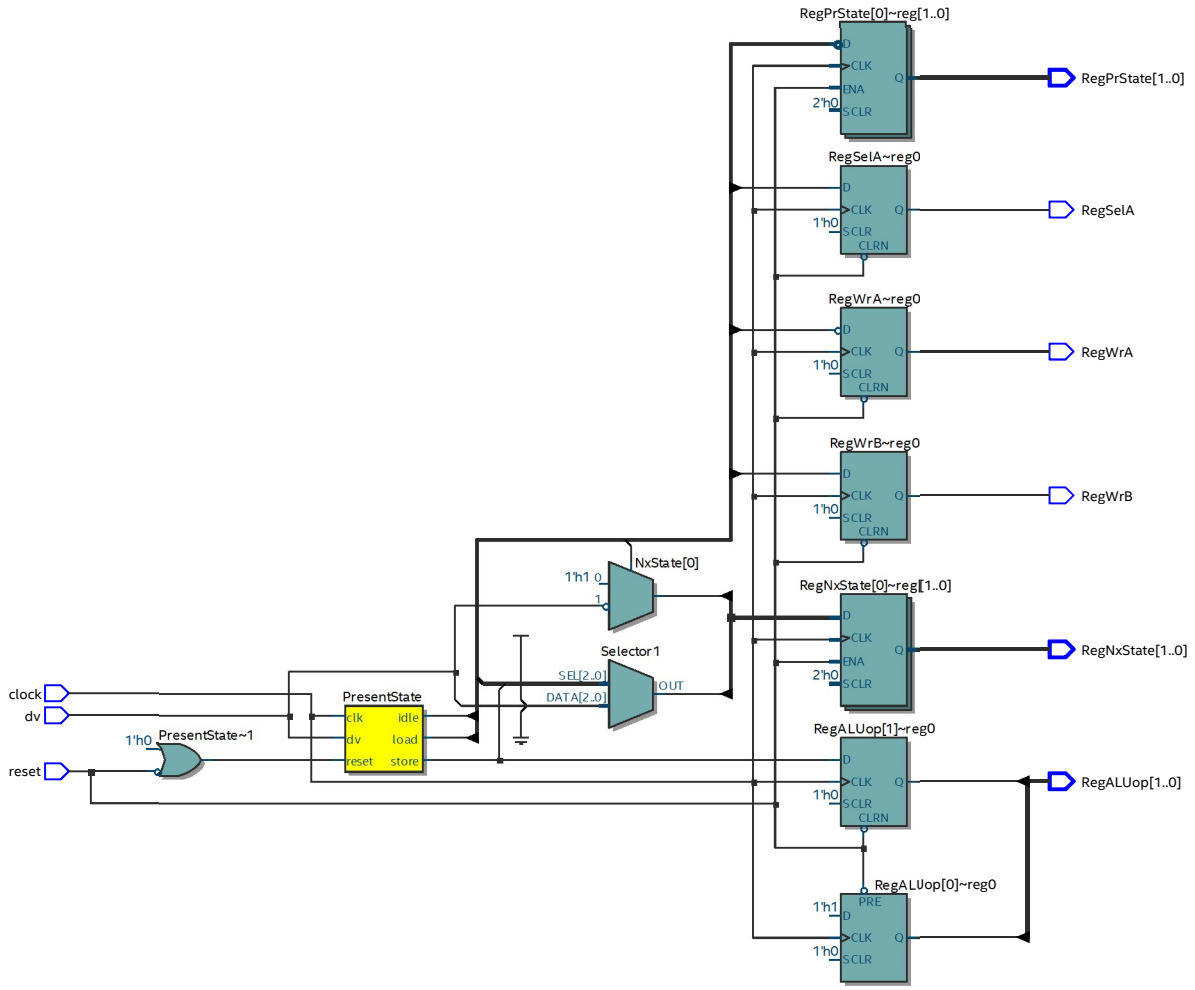

RegPrState[0]~reg[1..0]
D
CLK
ENA
SCLR
Q
2'h0
RegPrState[1..0]
RegSelA~reg0
D
CLK
SCLR
CLRN
Q
1'h0
RegSelA
RegWrA~reg0
D
CLK
SCLR
CLRN
Q
1'h0
RegWrA
RegWrB~reg0
D
CLK
SCLR
CLRN
Q
1'h0
RegWrB
1'h1
0
1
NxState[0]
Selector1
SEL[2..0]
DATA[2..0]
OUT
RegNxState[0]~reg[1..0]
D
CLK
ENA
SCLR
Q
2'h0
RegNxState[1..0]
clock
dv
reset
1'h0
PresentState~1
PresentState
clk
dv
reset
idle
load
store
RegALUop[1]~reg0
D
CLK
SCLR
CLRN
Q
1'h0
RegALUop[1..0]
RegALUop[0]~reg0
PRE
D
CLK
SCLR
Q
1'h1
1'h0

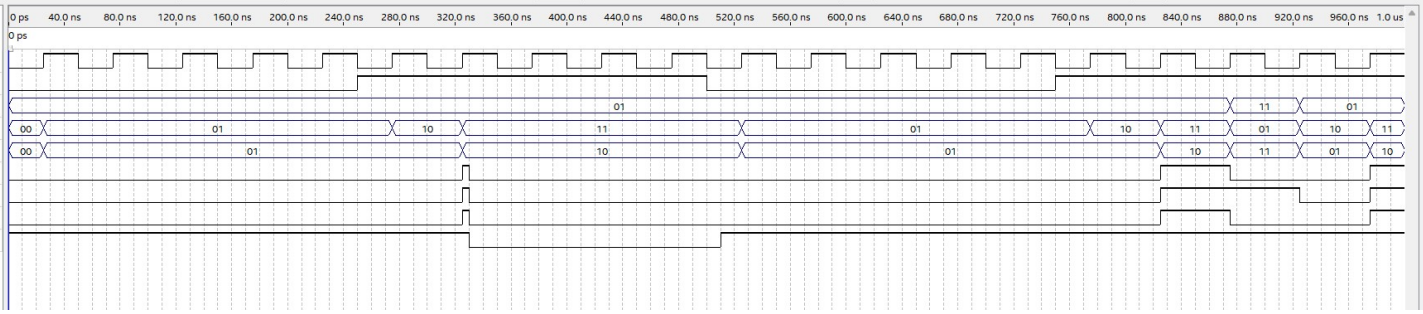
Simulation Waveform Editor - C:/Users/welly/Documents/GitHub/LT39A/pratica02/atv02 - atv02 - [atv02_20240613181600.sim.vwf (Read-Only)]

File Edit View Simulation Help

Search altera.com

Master Time Bar: 0 ps Pointer: 8.3 ns Interval: 8.3 ns Start: End:

Name	Value at 0 ps
clock	B 0
dv	B 0
RegALU...	B 01
RegNxSt...	B 00
RegPrSt...	B 00
RegSelA	B 0
RegWrA	B 0
RegWrB	B 0
reset	B 1



```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3
4   entity StateMachine_ControlUnit is
5       port
6       (
7           clock,reset: in std_logic;
8           dv: in std_logic;
9
10          RegSelA, RegWrA, RegWrB: out std_logic;
11          RegALUop: out std_logic_vector(1 downto 0);
12          RegPrState,RegNxState: out std_logic_vector(1 downto 0)
13      );
14  end entity;
15
16  architecture behavior of StateMachine_ControlUnit is
17
18  type states is (idle, load, store);
19  signal PresentState, NextState: states;
20
21
22  signal selA, wrA, wrB: std_logic;
23  signal ALUop: std_logic_vector(1 downto 0);
24  signal PrState,NxState: std_logic_vector(1 downto 0);
25
26  begin
27
28      process(clock,reset)
29      begin
30
31          if reset = '0' then
32              PresentState <= idle;
33          elsif rising_edge(clock) then
34              PresentState <= NextState;
35          end if;
36
37      end process;
38
39      process(PresentState,dv)
40      begin
41          case PresentState is
42              when idle =>
43                                  selA <= '0';
44                                  wrA <= '0';
45                                  wrB <= '0';
46                                  ALUop <= "01";
47                                  PrState <= "01";
48
49                                  if dv = '1' then
50                                      NextState <= load;
51                                      NxState <= "10";
52                                  else
53                                      NextState <= idle;
54                                      NxState <= "01";
55                                  end if;
56
57
58
59              when load =>        selA <= '1';
60                                  wrA <= '1';
61                                  wrB <= '1';
62                                  ALUop <= "01";
63                                  PrState <= "10";
64
65                                  NextState <= store;
66                                  NxState <= "11";
67
68              when store =>       selA <= '0';
69                                  wrA <= '1';
70                                  wrB <= '0';
71                                  ALUop <= "11";
72                                  PrState <= "11";
73
74                                  NextState <= idle;
75                                  NxState <= "01";
76          end case;
77      end process;
78
79      process(clock, reset)
80      begin
81
```

```
82              if reset = '0' then
83                  RegSelA <= '0';
84                  RegWrA <= '0';
85                  RegWrB <= '0';
86                  RegALUop <= "01";
87              elsif rising_edge(clock) then
88                  RegSelA <= selA;
89                  RegWrA <= wrA;
90                  RegWrB <= wrB;
91                  RegALUop <= ALUop;
92
93                  RegPrState <= PrState;
94                  RegNxState <= NxState;
95
96              end if;
97
98              end process;
99
100         end architecture;
```