

# Appendix A

## The Powers of 2

As we are dealing with binary values, it is useful to remember the values of the most significant powers of 2. Below are the smallest:

$2^0$	$2^1$	$2^2$	$2^3$	$2^4$	$2^5$	$2^6$	$2^7$	$2^8$	$2^9$	$2^{10}$
1	2	4	8	16	32	64	128	256	512	1024

We see that  $2^{10} = 1024$  is equal to approximately  $10^3 = 1000$ . Let the prefix K (meaning Kilo) indicate the value 1024 even though the *International Electrotechnical Commission* (IEC) standard establishes the prefix Kibi (from Kilo binary):

$2^{10}$	$2^{11}$	$2^{12}$	$2^{13}$	$2^{14}$	$2^{15}$	$2^{16}$	$2^{17}$	$2^{18}$	$2^{19}$	$2^{20}$
1K	2K	4K	8K	16K	32K	64K	128K	256K	512K	1024K

Similarly  $2^{20}$  let M (Mega) indicate (Mebi), equal to about  $10^6$ :

$2^{20}$	$2^{21}$	$2^{22}$	$2^{23}$	$2^{24}$	$2^{25}$	$2^{26}$	$2^{27}$	$2^{28}$	$2^{29}$	$2^{30}$
1M	2M	4M	8M	16M	32M	64M	128M	256M	512M	1024M

Let G (Giga) ( $10^9$ ) indicate (Gibi), which is equal to  $2^{30}$ :

$2^{30}$	$2^{31}$	$2^{32}$	$2^{33}$	$2^{34}$	$2^{35}$	$2^{36}$	$2^{37}$	$2^{38}$	$2^{39}$	$2^{40}$
1G	2G	4G	8G	16G	32G	64G	128G	256G	512G	1024G

Let T (Tera) indicate  $2^{40}$  (Tebi), about  $10^{12}$ .

To be thorough, a table including all the prefixes follows.

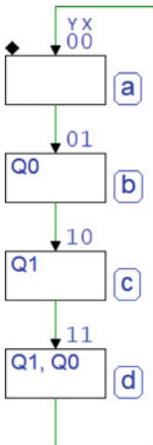
IEC prefix		Representations			Customary prefix	
Name	Symbol	Base 2	Base 1024	Base 10	Name	Symbol
Kibi	Ki	$2^{10}$	$1024^1$	$= 1.024 \times 10^3$	Kilo	k or K
Mebi	Mi	$2^{20}$	$1024^2$	$\approx 1.049 \times 10^6$	Mega	M
Gibi	Gi	$2^{30}$	$1024^3$	$\approx 1.074 \times 10^9$	Giga	G
Tebi	Ti	$2^{40}$	$1024^4$	$\approx 1.100 \times 10^{12}$	Tera	T
Pebi	Pi	$2^{50}$	$1024^5$	$\approx 1.126 \times 10^{15}$	Peta	P
Exbi	Ei	$2^{60}$	$1024^6$	$\approx 1.153 \times 10^{18}$	Exa	E
Zebi	Zi	$2^{70}$	$1024^7$	$\approx 1.181 \times 10^{21}$	Zetta	Z
Yobi	Yi	$2^{80}$	$1024^8$	$\approx 1.209 \times 10^{24}$	Yotta	Y

# Appendix B

## State Diagrams

The “*State Diagram*” is one of several methods for describing the behavior and performing the synthesis of a FSM, alternative to the ASM method developed in the book. It is quite similar to the former and still used in textbooks and documentation materials.

As the ASM, it describes the machine state-to-state transitions and outputs. In the following, we show how to convert *ASM diagrams* into *State Diagrams*, using as examples state machines presented in the book as ASM.

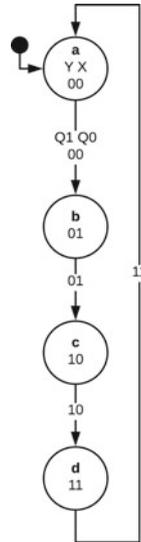


A first example of a *State Diagram* (right) is the module-4 counter described in Sect. 7.2.1 (left).

The circles, or “*bubbles*” represent the states, which are connected by *lines (arches)* with arrows that point to the direction of the transition.

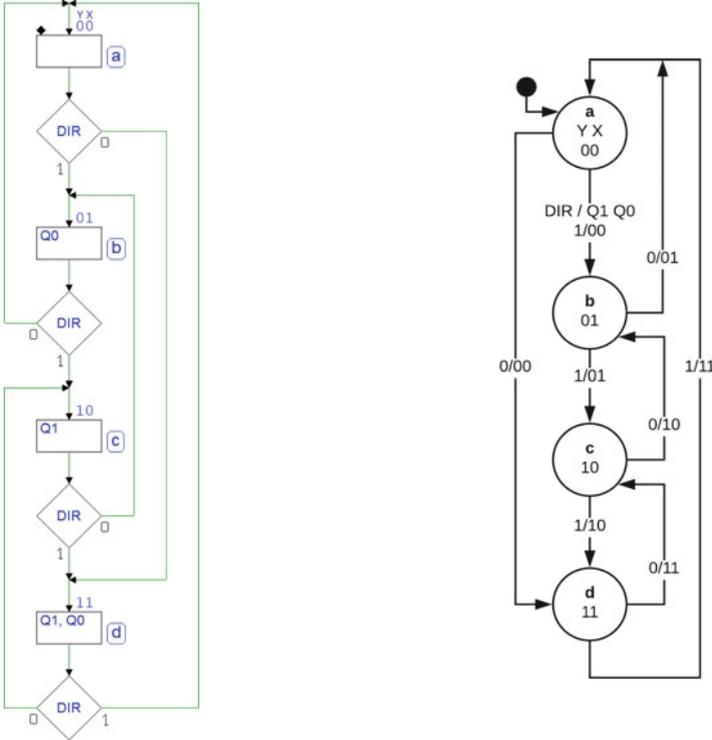
The states are identified by a letter (a, b, c, d); each circle contains also the corresponding binary code. The lines are labelled with the Q1 Q0 output values.

In other representations, in the case of a *Moore* machine, the outputs could also be written inside the state circles.

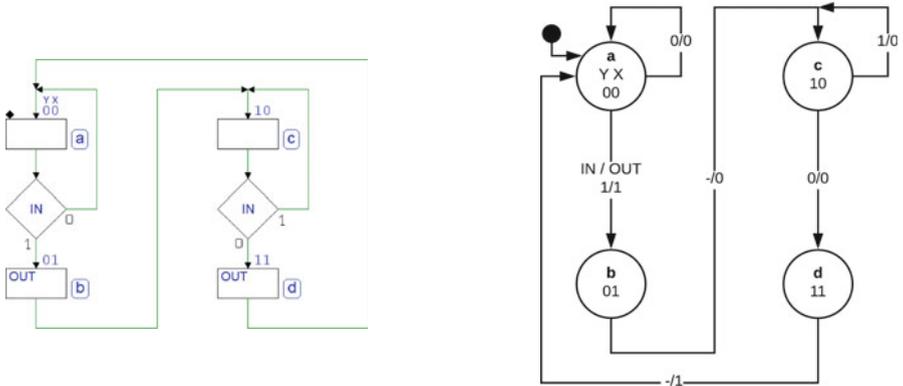


If there are inputs conditioning the state transitions, they are written by the lines, using digits separated by a slash. The first set of digits (before the slash) indicates the value of the inputs: with only one input there will be two lines exiting each state.

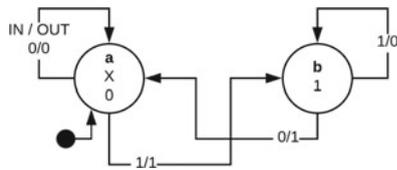
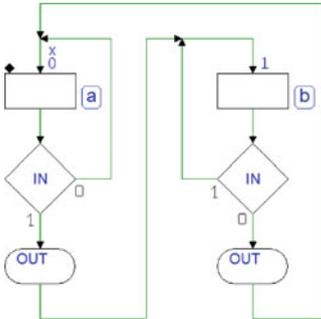
The figure below shows the bidirectional counter introduced in Sect. 7.2.2. There is now an input *DIR* that sets the count direction. If *DIR* = 1 the count is up, if *DIR* = 0 the count is down. The couple of digits after the slash shows the outputs, as in the previous case.



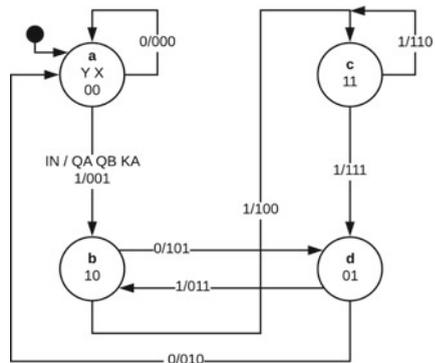
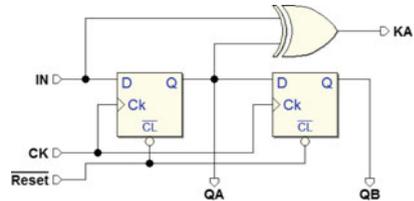
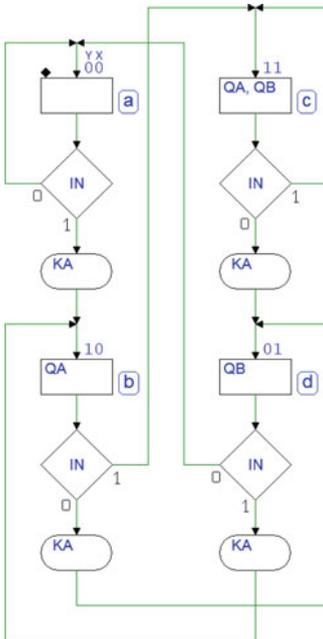
The edge detector (Sect. 7.2.2) is another example of a *Moore* machine. A line that starts and ends in the same state indicates a waiting loop.



In a *Mealy* machine, such as the edge detector with conditioned outputs (Sect. 7.2.3), the outputs are function of both state and inputs. The *State Diagram* indicates, as before, the outputs in the lines after the slash, but in this case, outputs in the same state are different when conditioned by an input.



The figure below is another example of a *Mealy* machine. It represents the algorithm of the circuit seen in Sect. 7.3.9.



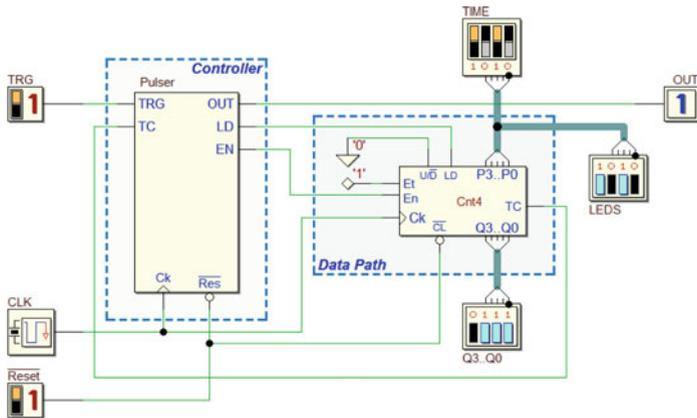
# Appendix C

## VHDL Code

In this pages, we list the VHDL codes described in Chap.9. Listings are almost complete. The omitted parts can be obtained by exporting with *Deeds* the projects available in the digital contents pages (on *Deeds* Web site).

### C.1 Code from the Pulse Generator Example

#### C.1.1 Schematic



#### C.1.2 Top Entity

```

1 -----
2 -- Deeds (Digital Electronics Education and Design Suite)
3 -- VHDL Code generated on (10/03/2018, 14:50:44)
4 -- by the Deeds (Digital Circuit Simulator) (Deeds-DcS)

```

```

5  -- Ver. 2.10.300 (Jan 19, 2018)
6  -- Copyright(c)2002-2018 University of Genoa, Italy
7  -- Web Site: https://www.digitalelectronicsdeeds.com
8  -----
9  -- Code compiled for: "DE0-CV Board"
10 -- Chip FPGA: Intel/Altera Cyclone(r) V (5C8BA4F23C7)
11 -- Proprietary EDA Tool: Quartus(r) II (Ver = 12.1spl)
12 -----
13
14 LIBRARY ieee;
15 USE ieee.std_logic_1164.ALL;
16 USE ieee.numeric_std.all;
17
18 ENTITY Pulser_DE0CV IS
19     PORT (
20
21         iCLOCK_50MHz: IN    std_logic;    --> PIN_V15 --> "iCLK": 2 Hz (Sw[09], LEDR[09]
22                                         --> Key[01] for Slow Clock Mode)
23
24         -----> Inputs:
25         iTRG:          IN    std_logic;    --> PIN_M6,   PButton: Key[03] H (if pressed)
26         inReset:      IN    std_logic;    --> PIN_P22, PButton: Reset L (if pressed)
27         iTIME_03:     IN    std_logic;    --> PIN_T12, Switch: Sw[03]
28         iTIME_02:     IN    std_logic;    --> PIN_T13, Switch: Sw[02]
29         iTIME_01:     IN    std_logic;    --> PIN_V13, Switch: Sw[01]
30         iTIME_00:     IN    std_logic;    --> PIN_U13, Switch: Sw[00]
31
32         -----> Outputs:
33         oOUT:         OUT   std_logic;    --> PIN_L2,   Red Led: LEDR[08]
34         oLEDS_03:    OUT   std_logic;    --> PIN_Y3,   Red Led: LEDR[03]
35         oLEDS_02:    OUT   std_logic;    --> PIN_W2,   Red Led: LEDR[02]
36         oLEDS_01:    OUT   std_logic;    --> PIN_AA1,  Red Led: LEDR[01]
37         oLEDS_00:    OUT   std_logic;    --> PIN_AA2,  Red Led: LEDR[00]
38         oQ3_Q0_03:   OUT   std_logic;    --> PIN_U1,   Red Led: LEDR[07]
39         oQ3_Q0_02:   OUT   std_logic;    --> PIN_U2,   Red Led: LEDR[06]
40         oQ3_Q0_01:   OUT   std_logic;    --> PIN_N1,   Red Led: LEDR[05]
41         oQ3_Q0_00:   OUT   std_logic;    --> PIN_N2,   Red Led: LEDR[04]
42
43         -----> Added Inputs:
44         iSLOW_Sw09:  IN    std_logic;    --> PIN_AB12 --> "iCLK" Sw[09]
45         iPULSE_Key01: IN   std_logic;    --> PIN_W9   --> "iCLK" Key[01]
46
47         -----> Added Outputs:
48         oCLOCK_ledR09: OUT std_logic;    --> PIN_L1,   Red Led: LEDR[09]
49
50         -----> Default Outputs:
51         oPIN_U21:    OUT   std_logic;    --> PIN_U21,  Seven Segm. Display: HEX 0 0 (a)
52         oPIN_V21:    OUT   std_logic;    --> PIN_V21,  Seven Segm. Display: HEX 0 1 (b)
53         oPIN_W22:    OUT   std_logic;    --> PIN_W22,  Seven Segm. Display: HEX 0 2 (c)
54         oPIN_W21:    OUT   std_logic;    --> PIN_W21,  Seven Segm. Display: HEX 0 3 (d)
55         oPIN_Y22:    OUT   std_logic;    --> PIN_Y22,  Seven Segm. Display: HEX 0 4 (e)
56         oPIN_Y21:    OUT   std_logic;    --> PIN_Y21,  Seven Segm. Display: HEX 0 5 (f)
57         oPIN_AA22:   OUT   std_logic;    --> PIN_AA22, Seven Segm. Display: HEX 0 6 (g)
58         oPIN_AA20:   OUT   std_logic;    --> PIN_AA20, Seven Segm. Display: HEX 1 0 (a)
59         oPIN_AB20:   OUT   std_logic;    --> PIN_AB20, Seven Segm. Display: HEX 1 1 (b)
60         oPIN_AA19:   OUT   std_logic;    --> PIN_AA19, Seven Segm. Display: HEX 1 2 (c)
61         oPIN_AA18:   OUT   std_logic;    --> PIN_AA18, Seven Segm. Display: HEX 1 3 (d)
62         oPIN_AB18:   OUT   std_logic;    --> PIN_AB18, Seven Segm. Display: HEX 1 4 (e)
63         oPIN_AA17:   OUT   std_logic;    --> PIN_AA17, Seven Segm. Display: HEX 1 5 (f)
64         oPIN_U22:    OUT   std_logic;    --> PIN_U22,  Seven Segm. Display: HEX 1 6 (g)
65         oPIN_Y19:    OUT   std_logic;    --> PIN_Y19,  Seven Segm. Display: HEX 2 0 (a)
66         oPIN_AB17:   OUT   std_logic;    --> PIN_AB17, Seven Segm. Display: HEX 2 1 (b)
67         oPIN_AA10:   OUT   std_logic;    --> PIN_AA10, Seven Segm. Display: HEX 2 2 (c)
68         oPIN_Y14:    OUT   std_logic;    --> PIN_Y14,  Seven Segm. Display: HEX 2 3 (d)
69         oPIN_V14:    OUT   std_logic;    --> PIN_V14,  Seven Segm. Display: HEX 2 4 (e)
70         oPIN_AB22:   OUT   std_logic;    --> PIN_AB22, Seven Segm. Display: HEX 2 5 (f)
71         oPIN_AB21:   OUT   std_logic;    --> PIN_AB21, Seven Segm. Display: HEX 2 6 (g)
72         oPIN_Y16:    OUT   std_logic;    --> PIN_Y16,  Seven Segm. Display: HEX 3 0 (a)
73         oPIN_W16:    OUT   std_logic;    --> PIN_W16,  Seven Segm. Display: HEX 3 1 (b)
74         oPIN_Y17:    OUT   std_logic;    --> PIN_Y17,  Seven Segm. Display: HEX 3 2 (c)
75         oPIN_V16:    OUT   std_logic;    --> PIN_V16,  Seven Segm. Display: HEX 3 3 (d)
76         oPIN_U17:    OUT   std_logic;    --> PIN_U17,  Seven Segm. Display: HEX 3 4 (e)
77         oPIN_V18:    OUT   std_logic;    --> PIN_V18,  Seven Segm. Display: HEX 3 5 (f)
78         oPIN_V19:    OUT   std_logic;    --> PIN_V19,  Seven Segm. Display: HEX 3 6 (g)
79         oPIN_U20:    OUT   std_logic;    --> PIN_U20,  Seven Segm. Display: HEX 4 0 (a)
80         oPIN_Y20:    OUT   std_logic;    --> PIN_Y20,  Seven Segm. Display: HEX 4 1 (b)
81         oPIN_V20:    OUT   std_logic;    --> PIN_V20,  Seven Segm. Display: HEX 4 2 (c)
82         oPIN_U16:    OUT   std_logic;    --> PIN_U16,  Seven Segm. Display: HEX 4 3 (d)
83         oPIN_U15:    OUT   std_logic;    --> PIN_U15,  Seven Segm. Display: HEX 4 4 (e)
84         oPIN_Y15:    OUT   std_logic;    --> PIN_Y15,  Seven Segm. Display: HEX 4 5 (f)
85         oPIN_P9:     OUT   std_logic;    --> PIN_P9,   Seven Segm. Display: HEX 4 6 (g)
86         oPIN_N9:     OUT   std_logic;    --> PIN_N9,   Seven Segm. Display: HEX 5 0 (a)
87         oPIN_M8:     OUT   std_logic;    --> PIN_M8,   Seven Segm. Display: HEX 5 1 (b)
88         oPIN_T14:    OUT   std_logic;    --> PIN_T14,  Seven Segm. Display: HEX 5 2 (c)
89         oPIN_P14:    OUT   std_logic;    --> PIN_P14,  Seven Segm. Display: HEX 5 3 (d)
90         oPIN_C1:     OUT   std_logic;    --> PIN_C1,   Seven Segm. Display: HEX 5 4 (e)
91         oPIN_C2:     OUT   std_logic;    --> PIN_C2,   Seven Segm. Display: HEX 5 5 (f)
92         oPIN_W19:    OUT   std_logic;    --> PIN_W19,  Seven Segm. Display: HEX 5 6 (g)
93     );

```

```

89 END Pulser_DE0CV;
90 ARCHITECTURE structural OF Pulser_DE0CV IS
91 ----->>> Components:
92 COMPONENT ClockScaler IS
93     PORT( iMclk: IN std_logic; -- Master Clock
94           iH4: IN std_logic; -- iH4..iH0 = "high" frequency selection
95           iH3: IN std_logic;
96           iH2: IN std_logic;
97           iH1: IN std_logic;
98           iH0: IN std_logic;
99           iL3: IN std_logic; -- iL3..iL0 = "low" frequency selection
100          iL2: IN std_logic; -- and Button Modes
101          iL1: IN std_logic;
102          iL0: IN std_logic;
103          iSwch: IN std_logic; -- Switch
104          iBut: IN std_logic; -- Button for manual pulsed Clock
105          oSclk: OUT std_logic; -- Output Clock
106          oled: OUT std_logic -- Slow "Clock Pulse" Led
107          );
108 END COMPONENT;
109 --
110 COMPONENT Counter4b IS
111     PORT( Ck : IN std_logic;
112           nCL: IN std_logic;
113           LD : IN std_logic;
114           ENP: IN std_logic;
115           ENT: IN std_logic;
116           UD : IN std_logic;
117           P3 : IN std_logic;
118           P2 : IN std_logic;
119           P1 : IN std_logic;
120           P0 : IN std_logic;
121           Q3 : OUT std_logic;
122           Q2 : OUT std_logic;
123           Q1 : OUT std_logic;
124           Q0 : OUT std_logic;
125           Tc : OUT std_logic );
126 END COMPONENT;
127 ----->>> Finite State Machine(s):
128 COMPONENT Pulser_EN IS
129     PORT ( ----->>>Clock & Reset:
130           Ck: IN std_logic;
131           Reset: IN std_logic;
132           ----->>>Inputs:
133           i_TRG: IN std_logic;
134           i_TC: IN std_logic;
135           ----->>>Outputs:
136           o_OUT: OUT std_logic;
137           o_LD: OUT std_logic;
138           o_EN: OUT std_logic
139           );
140 END COMPONENT;
141 ----->>> Signals:
142 SIGNAL S001: std_logic;
143 SIGNAL S002: std_logic;
144 SIGNAL S003: std_logic;
145 SIGNAL S004: std_logic;
146 SIGNAL S005: std_logic;
147 SIGNAL S006: std_logic;
148 SIGNAL S007: std_logic;
149 SIGNAL S008: std_logic;
150 SIGNAL S009: std_logic;
151 SIGNAL S010: std_logic;
152 SIGNAL S011: std_logic;
153 SIGNAL S012: std_logic;
154 SIGNAL S013: std_logic;
155 SIGNAL S014: std_logic;
156 SIGNAL S015: std_logic;
157 SIGNAL S016: std_logic;
158 SIGNAL S017: std_logic;
159 ----->>> Added Signals:
160 SIGNAL SLOW_Sw09: std_logic;
161 SIGNAL SPULSE_Key01: std_logic;
162 SIGNAL iCLK: std_logic;
163 SIGNAL SCLOCK_LedR09: std_logic;
164
165 BEGIN -- structural
166 ----->>> Input:
167 S002 <= NOT iTRG;
168 S005 <= iCLK;
169 S007 <= inReset;
170 S014 <= iTIME_00;
171 S015 <= iTIME_01;
172

```

```

173 S016 <= iTIME_02;
174 S017 <= iTIME_03;
175 -----> Output:
176 oOUT <= S008;
177 oLEDS_00 <= S014;
178 oLEDS_01 <= S015;
179 oLEDS_02 <= S016;
180 oLEDS_03 <= S017;
181 oQ3_Q0_00 <= S010;
182 oQ3_Q0_01 <= S011;
183 oQ3_Q0_02 <= S012;
184 oQ3_Q0_03 <= S013;
185 -----> Constants:
186 S004 <= '0';
187 S001 <= '1';
188 oPIN_U21 <= '1';
189 oPIN_V21 <= '1';
190 oPIN_W22 <= '1';
191 oPIN_W21 <= '1';
192 oPIN_Y22 <= '1';
193 oPIN_Y21 <= '1';
194 oPIN_AA22 <= '1';
195 oPIN_AA20 <= '1';
196 oPIN_AB20 <= '1';
197 oPIN_AA19 <= '1';
198 oPIN_AA18 <= '1';
199 oPIN_AB18 <= '1';
200 oPIN_AA17 <= '1';
201 oPIN_U22 <= '1';
202 oPIN_Y19 <= '1';
203 oPIN_AB17 <= '1';
204 oPIN_AA10 <= '1';
205 oPIN_Y14 <= '1';
206 oPIN_V14 <= '1';
207 oPIN_AB22 <= '1';
208 oPIN_AB21 <= '1';
209 oPIN_Y16 <= '1';
210 oPIN_W16 <= '1';
211 oPIN_Y17 <= '1';
212 oPIN_V16 <= '1';
213 oPIN_U17 <= '1';
214 oPIN_V18 <= '1';
215 oPIN_V19 <= '1';
216 oPIN_U20 <= '1';
217 oPIN_Y20 <= '1';
218 oPIN_V20 <= '1';
219 oPIN_U16 <= '1';
220 oPIN_U15 <= '1';
221 oPIN_Y15 <= '1';
222 oPIN_P9 <= '1';
223 oPIN_N9 <= '1';
224 oPIN_M8 <= '1';
225 oPIN_T14 <= '1';
226 oPIN_P14 <= '1';
227 oPIN_C1 <= '1';
228 oPIN_C2 <= '1';
229 oPIN_W19 <= '1';
230 -----> Component Mapping:
231 SLOW_Sw09 <= iSLOW_Sw09;
232 SPULSE_Key01 <= iPULSE_Key01;
233 oCLOCK_LedR09 <= sCLOCK_LedR09;
234 ClockScaler_iCLK: ClockScaler PORT MAP (
235     iCLOCK_50MHz, '1', '0', '1', '0', '0', '1', '1', '1', '1',
236     SSLOW_Sw09, SPULSE_Key01, iCLK, sCLOCK_LedR09 );
237
238 C680: Counter4b PORT MAP ( S005, S007, S003, S006, S001, S004, S017, S016,
239     S015, S014, S013, S012, S011, S010, S009 );
240 C704: Pulser_EN PORT MAP ( S005, S007, S002, S009, S008, S003, S006 );
241 END structural;

```

### C.1.3 Components

```

1 -----
2 -- Deeds (Digital Electronics Education and Design Suite)
3 -- VHDL Code generated on (10/03/2018, 14:50:44)
4 -- by the Deeds (Digital Circuit Simulator) (Deeds-DcS)
5 -- Ver. 2.10.300 (Jan 19, 2018)
6 -- Copyright(c)2002-2018 University of Genoa, Italy
7 -- Web Site: https://www.digitalelectronicsdeeds.com
8 -----
9 -- Code compiled for: "DE0-CV Board"

```

```

10 -- Chip FPGA: Intel/Altera Cyclone(r) V (5CEBA4F23C7)
11 -- Proprietary EDA Tool: Quartus(r) II (Ver = 12.1sp1)
12 -----
13
14 library ieee;
15 use ieee.std_logic_1164.all;
16 use ieee.numeric_std.all;
17
18 ENTITY Counter4b IS
19     PORT( Ck : IN std_logic;
20           nCL: IN std_logic;
21           LD : IN std_logic;
22           ENP: IN std_logic;
23           ENT: IN std_logic;
24           UD : IN std_logic;
25           P3 : IN std_logic;
26           P2 : IN std_logic;
27           P1 : IN std_logic;
28           P0 : IN std_logic;
29           Q3 : OUT std_logic;
30           Q2 : OUT std_logic;
31           Q1 : OUT std_logic;
32           Q0 : OUT std_logic;
33           Tc : OUT std_logic );
34 END Counter4b;
35
36 -----
37 ARCHITECTURE behavioral OF Counter4b IS
38 BEGIN
39     Count4b: PROCESS( Ck, nCL, ENP, ENT, UD )
40         variable aCnt: unsigned( 3 downto 0 );
41     BEGIN
42         if (nCL = '0') then aCnt := (others =>'0');
43         elsif (nCL = '1') then
44             if (Ck'event) AND (Ck='1') then
45                 if (LD = '1') then aCnt := (P3 & P2 & P1 & P0); -- Load
46                 elsif (LD = '0') then
47                     if (ENP = '1') and (ENT = '1') then
48                         if (UD = '1') then
49                             if (aCnt < "1111") then aCnt := aCnt + 1;
50                             else aCnt := (others =>'0');
51                             end if;
52                         elsif (UD = '0') then
53                             if (aCnt > "0000") then aCnt := aCnt - 1;
54                             else aCnt := (others =>'1');
55                             end if;
56                         else aCnt := (others =>'X'); -- (UD: Unknown)
57                         END IF;
58                     elsif not ((ENP='0') or
59                                (ENT='0')) then aCnt := (others =>'X'); -- (ENP: Unknown)
60                     END IF;
61                 else aCnt := (others =>'X'); -- (LD: Unknown)
62                 END IF;
63             END IF;
64         else aCnt := (others =>'X'); -- (nCL: Unknown)
65         END IF;
66         --
67         Tc <= ENT and ( (aCnt(3) and aCnt(2) and aCnt(1) and aCnt(0) and UD) or
68                       (not(aCnt(3) or aCnt(2) or aCnt(1) or aCnt(0) or UD)) );
69         --
70         Q3 <= aCnt(3);
71         Q2 <= aCnt(2);
72         Q1 <= aCnt(1);
73         Q0 <= aCnt(0);
74     END PROCESS;
75 END behavioral;
76 -----
77 library ieee;
78 use ieee.std_logic_1164.all;
79 use ieee.numeric_std.all;
80
81 -- Clock Scaler (Altera DE1, DE2 and DE2-115 version, master clock = 50 MHz)
82 ENTITY ClockScaler IS
83     PORT( iMClk: IN std_logic; -- Master Clock
84           iH4: IN std_logic; -- iH4..iH0 = "high" fr. sel.
85           iH3: IN std_logic;
86           iH2: IN std_logic;
87           iH1: IN std_logic;
88           iH0: IN std_logic;
89           iL3: IN std_logic; -- iL3..iL0 = "low" freq. sel.
90           iL2: IN std_logic; --
91           iL1: IN std_logic; -- and Button Modes
92           iL0: IN std_logic;
93           iSwch: IN std_logic; -- Switch

```

```

94     iBut: IN std_logic;    -- Button for manual pulsed Clock
95     oCk:  OUT std_logic;   -- Output Clock
96     oLed: OUT std_logic   -- Slow "Clock Pulse" Led
97     );
98     END ClockScaler;
99
100 -----
101 ARCHITECTURE behavioral OF ClockScaler IS
102 BEGIN
103     -- ...omissis... (see the complete code generated by Deeds)
104 END behavioral;

```

## C.1.4 Finite State Machine

```

1  -----
2  -- Deeds (Digital Electronics Education and Design Suite)
3  -- VHDL Code generated on (10/03/2018, 14:50:44)
4  -- by the Deeds (Finite State Machine Simulator)(Deeds-FsM)
5  -- Ver. 2.10.300 (Jan 19, 2018)
6  -- Copyright (c)2002-2018 University of Genoa, Italy
7  -- Web Site: https://www.digitalelectronicsdeeds.com
8  -----
9
10 LIBRARY ieee;
11 USE ieee.std_logic_1164.ALL;
12
13 ENTITY Pulser_EN IS
14     PORT ( ----->Clock & Reset:
15         Ck:      IN std_logic;
16         Reset:  IN std_logic;
17         ----->Inputs:
18         i_TRG:  IN std_logic;
19         i_TC:   IN std_logic;
20         ----->Outputs:
21         o_OUT:  OUT std_logic;
22         o_LD:   OUT std_logic;
23         o_EN:   OUT std_logic );
24 END Pulser_EN;
25
26 ARCHITECTURE behavioral OF Pulser_EN IS      -- (Behavioral Description)
27     TYPE states is ( state_rs,
28                     state_wt,
29                     state_ps,
30                     dummy_ll );
31     SIGNAL State,
32            Next_State: states;
33 BEGIN
34     -- Next State Combinational Logic -----
35     FSM: process( State, i_TRG, i_TC )
36     begin
37         CASE State IS
38             when state_wt =>
39                 if (i_TRG = '1') then
40                     Next_State <= state_ps;
41                 else
42                     Next_State <= state_wt;
43                 end if;
44             when state_ps =>
45                 if (i_TC = '1') then
46                     Next_State <= state_rs;
47                 else
48                     Next_State <= state_ps;
49                 end if;
50             when state_rs =>
51                 if (i_TRG = '1') then
52                     Next_State <= state_rs;
53                 else
54                     Next_State <= state_wt;
55                 end if;
56             when OTHERS =>
57                 Next_State <= state_rs;
58         END case;
59     end process;
60
61     -- State Register -----
62     REG: process( Ck, Reset )
63     begin
64         if (Reset = '0') then
65             State <= state_rs;
66         elsif rising_edge(Ck) then
67             State <= Next_State;

```

```

68         end if;
69     end process;
70
71     -- Outputs Combinational Logic -----
72     OUTPUTS: process( State, i_TRG, i_TC )
73     begin
74         -- Set output defaults:
75         o_OUT <= '0';
76         o_LD <= '0';
77         o_EN <= '0';
78
79         -- Set output as function of current state and input:
80         CASE State IS
81             when state_wt =>
82                 o_LD <= '1';
83             when state_ps =>
84                 o_OUT <= '1';
85                 if (i_TC = '0') then
86                     o_EN <= '1';
87                 end if;
88             when OTHERS =>
89                 o_OUT <= '0';
90                 o_LD <= '0';
91                 o_EN <= '0';
92         END case;
93     end process;
94 END behavioral;

```

## C.2 Other VHDL Examples

### C.2.1 Decoder

```

1  -----
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  ENTITY Decoder_2_4 IS
6      PORT( A1: IN    std_logic;
7            A0: IN    std_logic;
8            EN: IN    std_logic;
9            Y0: OUT   std_logic;
10           Y1: OUT   std_logic;
11           Y2: OUT   std_logic;
12           Y3: OUT   std_logic );
13 END Decoder_2_4;
14
15 ARCHITECTURE behavioral OF Decoder_2_4 IS
16     SIGNAL aNumber: std_logic_vector( 2 downto 0 );
17 BEGIN
18     aNumber <= EN & A1 & A0;
19     with aNumber select
20         Y0 <= '0' when "000", '0' when "001",
21             '0' when "010", '0' when "011",
22             '1' when "100", '0' when "101",
23             '0' when "110", '0' when "111", 'X' when others;
24     with aNumber select
25         Y1 <= '0' when "000", '0' when "001",
26             '0' when "010", '0' when "011",
27             '0' when "100", '1' when "101",
28             '0' when "110", '0' when "111", 'X' when others;
29     with aNumber select
30         Y2 <= '0' when "000", '0' when "001",
31             '0' when "010", '0' when "011",
32             '0' when "100", '0' when "101",
33             '1' when "110", '0' when "111", 'X' when others;
34     with aNumber select
35         Y3 <= '0' when "000", '0' when "001",
36             '0' when "010", '0' when "011",
37             '0' when "100", '0' when "101",
38             '0' when "110", '1' when "111", 'X' when others;
39 END behavioral;

```

## C.2.2 Multiplexer

```

1 -----
2 library ieee;
3 use ieee.std_logic_1164.all;
4
5 ENTITY Multiplexer_4_1 IS
6     PORT( I0: IN    std_logic;
7           I1: IN    std_logic;
8           I2: IN    std_logic;
9           I3: IN    std_logic;
10          S1: IN    std_logic;
11          S0: IN    std_logic;
12          Q: OUT   std_logic );
13 END Multiplexer_4_1;
14
15 ARCHITECTURE behavioral OF Multiplexer_4_1 IS
16 BEGIN
17     Q <= I0 when ((S1 = '0') and (S0 = '0')) else
18          I1 when ((S1 = '0') and (S0 = '1')) else
19          I2 when ((S1 = '1') and (S0 = '0')) else
20          I3 when ((S1 = '1') and (S0 = '1')) else 'X';
21 END behavioral;

```

## C.2.3 Demultiplexer

```

1 -----
2 library ieee;
3 use ieee.std_logic_1164.all;
4 ENTITY Demultiplexer_1_4 IS
5     PORT( I: IN    std_logic;
6           S1: IN    std_logic;
7           S0: IN    std_logic;
8           Q0: OUT   std_logic;
9           Q1: OUT   std_logic;
10          Q2: OUT   std_logic;
11          Q3: OUT   std_logic );
12 END Demultiplexer_1_4;
13
14 ARCHITECTURE behavioral OF Demultiplexer_1_4 IS
15     SIGNAL aNumber: std_logic_vector( 2 downto 0 );
16 BEGIN
17     aNumber <= I & S1 & S0;
18     with aNumber select
19         Q0 <= '0' when "000", '0' when "001",
20              '0' when "010", '0' when "011",
21              '1' when "100", '0' when "101",
22              '0' when "110", '0' when "111", 'X' when others;
23     with aNumber select
24         Q1 <= '0' when "000", '0' when "001",
25              '0' when "010", '0' when "011",
26              '0' when "100", '1' when "101",
27              '0' when "110", '0' when "111", 'X' when others;
28     with aNumber select
29         Q2 <= '0' when "000", '0' when "001",
30              '0' when "010", '0' when "011",
31              '0' when "100", '0' when "101",
32              '1' when "110", '0' when "111", 'X' when others;
33     with aNumber select
34         Q3 <= '0' when "000", '0' when "001",
35              '0' when "010", '0' when "011",
36              '0' when "100", '0' when "101",
37              '0' when "110", '1' when "111", 'X' when others;
38 END behavioral;

```

## C.2.4 Full Adder

```

1  -----
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  ENTITY Adder_Full IS
6      PORT( CIN: IN   std_logic;
7            COUT:OUT std_logic;
8            A:   IN   std_logic;
9            B:   IN   std_logic;
10           S:   OUT  std_logic );
11 END Adder_Full;
12
13 ARCHITECTURE behavioral OF Adder_Full IS
14     SIGNAL ABC: std_logic_vector( 2 downto 0 );
15 BEGIN
16     ABC <= A & B & CIN;
17     with ABC select
18     S <= '0' when "000",
19          '1' when "001",
20          '1' when "010",
21          '0' when "011",
22          '1' when "100",
23          '0' when "101",
24          '0' when "110",
25          '1' when "111",
26          'X' when others;
27     with ABC select
28     COUT <= '0' when "000",
29             '0' when "001",
30             '0' when "010",
31             '1' when "011",
32             '0' when "100",
33             '1' when "101",
34             '1' when "110",
35             '1' when "111",
36             'X' when others;
37 END behavioral;

```

## C.2.5 Comparator

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  ENTITY Compar_4 IS
6      PORT( A3: IN   std_logic;
7            A2: IN   std_logic;
8            A1: IN   std_logic;
9            A0: IN   std_logic;
10           B3: IN   std_logic;
11           B2: IN   std_logic;
12           B1: IN   std_logic;
13           B0: IN   std_logic;
14           MIN: OUT  std_logic;
15           EQU: OUT  std_logic;
16           MAJ: OUT  std_logic );
17 END Compar_4;
18
19 ARCHITECTURE behavioral OF Compar_4 IS
20 BEGIN
21     Cmp4: PROCESS( A3, A2, A1, A0, B3, B2, B1, B0 )
22     variable A: unsigned( 3 downto 0 );
23     variable B: unsigned( 3 downto 0 );
24     BEGIN
25         A := (A3 & A2 & A1 & A0);
26         B := (B3 & B2 & B1 & B0);
27         --
28         if (A > B) then MIN <= '0'; EQU <= '0'; MAJ <= '1';
29         elsif (A < B) then MIN <= '1'; EQU <= '0'; MAJ <= '0';

```

```

30     elsif (A = B) then MIN <= '0'; EQU <= '1'; MAJ <= '0';
31     else MIN <= 'X'; EQU <= 'X'; MAJ <= 'X';
32     END IF;
33     END PROCESS;
34 END behavioral;

```

## C.2.6 Flip-Flop D-PET

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  ENTITY DpetFF IS
5      PORT( D, Ck : IN std_logic;
6            nCL, nPR: IN std_logic;
7            Q, nQ : OUT std_logic );
8  END DpetFF;
9
10 ARCHITECTURE behavioral OF DpetFF IS
11 BEGIN
12     Dff: PROCESS( Ck, nCL, nPR )
13     BEGIN
14         if (nCL='0') and (nPR='0') then Q <= 'X'; nQ <= 'X';
15         elsif (nCL='0') and (nPR='1') then Q <= '0'; nQ <= '1';
16         elsif (nCL='1') and (nPR='0') then Q <= '1'; nQ <= '0';
17         elsif (nCL='1') and (nPR='1') then
18             if (Ck'event) AND (Ck='1') THEN -- Positive Edge
19                 Q <= D; nQ <= not D;
20             END IF;
21         else Q <= 'X'; nQ <= 'X';
22         END IF;
23     END PROCESS;
24 END behavioral;

```

## C.2.7 Flip-Flop E-PET

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  ENTITY EpetFF IS
5      PORT( D, E, Ck: IN std_logic;
6            nCL, nPR: IN std_logic;
7            Q, nQ : OUT std_logic );
8  END EpetFF;
9  ARCHITECTURE behavioral OF EpetFF IS
10 BEGIN
11     Eff: PROCESS( Ck, nCL, nPR )
12     BEGIN
13         if (nCL='0') and (nPR='0') then Q <= 'X'; nQ <= 'X';
14         elsif (nCL='0') and (nPR='1') then Q <= '0'; nQ <= '1';
15         elsif (nCL='1') and (nPR='0') then Q <= '1'; nQ <= '0';
16         elsif (nCL='1') and (nPR='1') then
17             if (Ck'event) AND (Ck='1') THEN -- Positive Edge
18                 if (E = '1') then Q <= D; nQ <= not D;
19                 elsif not(E = '0') then Q <= 'X'; nQ <= 'X';
20             END IF;
21         END IF;
22         else Q <= 'X'; nQ <= 'X';
23         END IF;
24     END PROCESS;
25 END behavioral;

```

## C.2.8 Flip-Flop JK-PET

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  ENTITY JKpetFF IS

```

```

5   PORT( J, K, Ck: IN std_logic;
6         nCL, nPR: IN std_logic;
7         Q, nQ   : OUT std_logic );
8   END JKpetFF;
9
10  ARCHITECTURE behavioral OF JKpetFF IS
11  BEGIN
12    JKff: PROCESS( Ck, nCL, nPR )
13      variable OutQ: STD_LOGIC;
14    BEGIN
15      if (nCL='0') and (nPR='1') then OutQ := '0';
16      elsif (nCL='1') and (nPR='0') then OutQ := '1';
17      elsif (nCL='1') and (nPR='1') then
18        if (Ck'event) AND (Ck='1') THEN
19          -- Positive Edge
20          if (J = '0') AND (K = '1') THEN OutQ := '0';
21          elsif (J = '1') AND (K = '0') THEN OutQ := '1';
22          elsif (J = '1') AND (K = '1') THEN OutQ := not OutQ;
23          elsif not((J='0') AND (K='0')) THEN OutQ := 'X';
24        END IF;
25      END IF;
26    else
27      OutQ := 'X';
28    END IF;
29    Q <= ( OutQ);
30    nQ <= (not OutQ);
31  END PROCESS;
32  END behavioral;

```

## C.2.9 Parallel Register

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  ENTITY PiPoE4 IS
5  PORT( Ck : IN std_logic;
6        nCL: IN std_logic;
7        E  : IN std_logic;
8        P3 : IN std_logic;
9        P2 : IN std_logic;
10       P1 : IN std_logic;
11       P0 : IN std_logic;
12       Q3 : OUT std_logic;
13       Q2 : OUT std_logic;
14       Q1 : OUT std_logic;
15       Q0 : OUT std_logic );
16  END PiPoE4;
17  ARCHITECTURE behavioral OF PiPoE4 IS
18  BEGIN
19    RegPiPoE4: PROCESS( Ck, nCL )
20      variable aReg: std_logic_vector( 3 downto 0 );
21    BEGIN
22      if (nCL = '0') then aReg := (others =>'0');
23      elsif (nCL = '1') then
24        if (Ck'event) AND (Ck='1') THEN -- Positive Edge
25          if (E = '1') then
26            aReg := (P3 & P2 & P1 & P0);
27            elsif not(E = '0') then
28              aReg := (others =>'X');
29          END IF;
30        END IF;
31      else
32        aReg := (others =>'X');
33      END IF;
34      Q3 <= aReg(3);
35      Q2 <= aReg(2);
36      Q1 <= aReg(1);
37      Q0 <= aReg(0);
38    END PROCESS;
39  END behavioral;

```

### C.2.10 Shift Register (S.I.P.O.)

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  ENTITY SiPoE4 IS
5      PORT( I   : IN std_logic;
6            Ck  : IN std_logic;
7            nCL : IN std_logic;
8            E   : IN std_logic;
9            Q3  : OUT std_logic;
10           Q2  : OUT std_logic;
11           Q1  : OUT std_logic;
12           Q0  : OUT std_logic );
13 END SiPoE4;
14
15 ARCHITECTURE behavioral OF SiPoE4 IS
16 BEGIN
17     RegSiPoE4: PROCESS( Ck, nCL )
18         variable aReg: std_logic_vector( 3 downto 0 );
19         BEGIN
20             if (nCL = '0') then aReg := (others =>'0');
21             elsif (nCL = '1') then
22                 if (Ck'event) AND (Ck='1') THEN -- Positive Edge
23                     if (E = '1') then
24                         aReg := (I & aReg(3) & aReg(2) & aReg(1));
25                         elsif not (E = '0') then
26                             aReg := (others =>'X');
27                     END IF;
28                 END IF;
29             else aReg := (others =>'X');
30             END IF;
31             --
32             Q3 <= aReg(3);
33             Q2 <= aReg(2);
34             Q1 <= aReg(1);
35             Q0 <= aReg(0);
36             --
37         END PROCESS;
38     END behavioral;

```