
30010 Programmeringsprojekt: Reflex Ball



Mads Friis Bornebusch (s123627)



Tobias Tuxen (s120213)



Kristian Sloth Lauszus (s123808)

Reflex Ball
Group Number: 15
DTU Space
June 24, 2013

Resumé

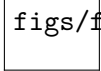
I dette programmeringsprojekt har vi skrevet og designet al koden til spillet Reflex Ball. Koden er skrevet i C i compileren Z8Encore! og implementeret på en Zilog 6403 microcontroller.

Vi har konkluderet at den skrevne kode implementerer det ønskede kredsløb på Microcontrolleren, da alle spillets facetter er blevet gennemtestet og giver det ønskede output. Som slutresultat har vi et fuldt funktionelt ReflexBall-spil med mange udvidelser der bl.a. inkluderer brikker (Arkanoid-stil), power-ups, forskellige levels og styring med ret. Der er indtil videre ikke fundet nogle bugs i slutversionen af spillet.

Abstract

In this programming project we have written and designed all the code to the game Reflex Ball. The code has been written in C in the compiler Z8Encore! and has been implemented on a Zilog 6403 Microcontroller.

We have concluded that the written code implements the desired circuit on the Microcontroller as as all the different facilities of the game has been thoroughly tested and is in compliance with the expected result. In the end we have a fully functional Reflex Ball Game with many expansions, which amongst others, include bricks (Arkanoid style), power-ups, different levels and steering-wheel game controller. So far no bugs has been found in the final version of the game.



figs/forside.png

Figure 1: Screenshot from our implementation of ReflexBall

Fordord

Denne rapport er skrevet som en del af eksaminationen i DTU kursus 30010 Programmeringsprojekt. Alle tre, på forsiden nævnte, gruppemedlemmer har bidraget til rapporten på lige vis. Vi har lavet alle forberedelsesøvelser, skrevet koden og afsnittene i rapporten i fællesskab. Denne rapport beskriver vores arbejde og resultater.

Contents

1 Introduktion

This report documents the design and implementation of the control part of a vending-machine for soft-drinks.

The project was divided into three laboratory exercises, in which parts of the system was designed. After the finalization of all three laboratory exercises a complete vending machine control circuit was designed and implemented on a Basys2 FPGA board. The optional parts of the lab exercises is included in the elaboration of each of the exercises, which also redefines some of the mandatory functions, and update/change them as the requirements of each part changes as we move towards the finished control part.

2 Konklusion

Efter spillet et blevet designet, skrevet og uploadet til Zilog 6403 microcontrolleren, er alle spillets facetter blevet gennemtestet og fundet i overensstemmelse med det forventede resultat. Vi kan derfor konkludere at den skrevne kode implementerer spillet Reflex Ball på microcontrolleren og i hyperterminalen korrekt i forhold til specifikationskravene. Derudover kan vi også konkludere at alle udvidelse virker som forventet.

A Appendiks 1

This appendix contains the complete VHDL-code used, and written/modified by us

```

1  -----
2  -- This component divides the 50 MHz clock signal down to a clk signal
3  -- of 762Hz and a clk_3 signal of 3Hz.
4  -----
5
6  library ieee;
7  use ieee.std_logic_1164.all;
8  use ieee.numeric_std.all;
9
10 entity clock_manager is
11     port(clk_50 : in  std_logic;
12          clk_man : in  std_logic;
13          sel_man : in  std_logic;
14          clk     : out std_logic;
15          clk_3   : out std_logic);
16 end clock_manager;
17
18 architecture structure of clock_manager is
19     signal count, count_next : unsigned(15 downto 0) := (others => '0');
20     signal count3, count3_next : unsigned(23 downto 0) := (others => '0');
21     attribute clock_signal : string;
22     attribute clock_signal of clk : signal is "yes";
23
24 begin
25     count_next <= count + 1;
26     count3_next <= count3 + 1;
27
28 -----
29 -- this process generates the clk and clk3 signal, which are derived from
30 -- the 50MHz clock. When MSB of count = '1' clk will go high, when MSB of
31 -- count = '0' clk will go low. Same with count3 and clk_3. If sel_man is
32 -- set to 1, clk will be set to the input signal clk_man.
33 -----
34
35     process(sel_man, clk_man, count, count3)
36     begin
37         if sel_man = '1' then
38             clk <= clk_man;
39         else
40             clk <= count(15);
41         end if;
42         clk_3 <= count3(23);
43     end process;
44
45 -----
46 -- on the rising edge of the 50MHz clock
47 -- count and count3 will be increased by 1
48 -----

```

Figure 2: The clock manager, include our own 3 Hz clock for use in the alarm signal

/Users/hbm/Dropbox/DTU-Amigos/0...ngMachine/src/clock_manager.vhd Page 2 of 2
Saved: 5/5/13 7:51:45 PM Printed For: Hjalte Bested Møller

```
49  
50     process(clk_50, count, count3)  
51     begin  
52         if rising_edge(clk_50) then  
53             count <= count_next;  
54             count3 <= count3_next;  
55         end if;  
56     end process;  
57  
58 end structure;  
59
```

Figure 3: Page 2

/Users/hbm/Dropbox/DTU-Amigos/0...gMachine/src/display_driver.vhd Page 1 of 5
 Saved: 5/2/13 11:26:16 AM Printed For: Hjalte Bested Møller

```

1  -----
2  -- This component converts the sum and price signals with BCD into
3  -- four seven segment displays. The code for displaying hexadecimal
4  -- numbers as well as additional letters are also contained.
5  -- This component also receives an alarm signal which causes the
6  -- display to blink. The cola, aqua and hash signals, if positive,
7  -- causes the display to show these words.
8  -----
9
10 library ieee;
11 use ieee.std_logic_1164.all;
12 use ieee.numeric_std.all;
13
14 entity display_driver is
15     port(sum : in unsigned(5 downto 0);
16          price : in unsigned(5 downto 0);
17          reset : in std_logic;
18          clock : in std_logic;
19          clk_3 : in std_logic;
20          alarm : in std_logic;
21          cola : in std_logic;
22          aqua : in std_logic;
23          hash : in std_logic;
24          an : out std_logic_vector(3 downto 0);
25          led : out std_logic_vector(1 to 8));
26 end display_driver;
27
28 architecture Behavior of display_driver is
29     signal m, m_next : unsigned(1 downto 0);
30     signal d : unsigned(4 downto 0);
31     signal sumL, sumH : unsigned(4 downto 0);
32     signal priceL, priceH : unsigned(4 downto 0);
33     signal anTemp : std_logic_vector(3 downto 0);
34
35 begin
36     m_next <= m + 1;
37
38     -----
39     -- BCD converter for price input signal
40     -----
41     process(price)
42     begin
43         if price >= 60 then
44             priceH <= "00110";
45             priceL <= price - 60;
46         elsif price >= 50 then
47             priceH <= "00101";
48             priceL <= price - 50;

```

Figure 4: Display driver for displaying symbols/numbers on each of the four seven segment displays

/Users/hbm/Dropbox/DTU-Amigos/0...gMachine/src/display_driver.vhd Page 2 of 5
 Saved: 5/2/13 11:26:16 AM Printed For: Hjalte Bested Møller

```

49     elsif price >= 40 then
50         priceH <= "00100";
51         priceL <= price - 40;
52     elsif price >= 30 then
53         priceH <= "00011";
54         priceL <= price - 30;
55     elsif price >= 20 then
56         priceH <= "00010";
57         priceL <= price - 20;
58     elsif price >= 10 then
59         priceH <= "00001";
60         priceL <= price - 10;
61     else
62         priceH <= "00000";
63         priceL <= price(4 downto 0);
64     end if;
65 end process;
66
67 -----
68 -- BCD converter for sum input signal
69 -----
70 process(sum)
71 begin
72     if sum >= 60 then
73         sumH <= "00110";
74         sumL <= sum - 60;
75     elsif sum >= 50 then
76         sumH <= "00101";
77         sumL <= sum - 50;
78     elsif sum >= 40 then
79         sumH <= "00100";
80         sumL <= sum - 40;
81     elsif sum >= 30 then
82         sumH <= "00011";
83         sumL <= sum - 30;
84     elsif sum >= 20 then
85         sumH <= "00010";
86         sumL <= sum - 20;
87     elsif sum >= 10 then
88         sumH <= "00001";
89         sumL <= sum - 10;
90     else
91         sumH <= "00000";
92         sumL <= sum(4 downto 0);
93     end if;
94 end process;
95
96 -----

```

Figure 5: Page 2

```

97      -- m is the multiplexer select signal which selects which of the sever
98      -- segment displays are turned on at any given time. There is a sequer
99      -- hierarchy in which cola overrules aqua, which overrules hash. If or
100     -- these signals = '1', the word is shown on the display, if none of t
101     -- signals = '1', price is shown on the two first seven seg displays w
102     -- sum is shown on the last two.
103     -----
104     process(m)
105     begin
106         if cola = '1' then
107             case m is
108                 when "00" =>
109                     anTemp <= NOT "0001";
110                     d      <= "01010";
111                 when "01" =>
112                     anTemp <= NOT "0010";
113                     d      <= "10010";
114                 when "10" =>
115                     anTemp <= NOT "0100";
116                     d      <= "00000";
117                 when "11" =>
118                     anTemp <= NOT "1000";
119                     d      <= "01100";
120                 when others =>
121                     anTemp <= NOT "0000";
122                     d      <= "00000";
123             end case;
124         elsif aqua = '1' then
125             case m is
126                 when "00" =>
127                     anTemp <= NOT "0001";
128                     d      <= "01010";
129                 when "01" =>
130                     anTemp <= NOT "0010";
131                     d      <= "10011";
132                 when "10" =>
133                     anTemp <= NOT "0100";
134                     d      <= "01001";
135                 when "11" =>
136                     anTemp <= NOT "1000";
137                     d      <= "01010";
138                 when others =>
139                     anTemp <= NOT "0000";
140                     d      <= "00000";
141             end case;
142         elsif hash = '1' then
143             case m is
144                 when "00" =>

```

Figure 6: Page 3

/Users/hbm/Dropbox/DTU-Amigos/0...gMachine/src/display_driver.vhd Page 4 of 5
 Saved: 5/2/13 11:26:16 AM Printed For: Hjalte Bested Møller

```

145         anTemp <= NOT "0001";
146         d      <= "10000";
147     when "01" =>
148         anTemp <= NOT "0010";
149         d      <= "10001";
150     when "10" =>
151         anTemp <= NOT "0100";
152         d      <= "01010";
153     when "11" =>
154         anTemp <= NOT "1000";
155         d      <= "10000";
156     when others =>
157         anTemp <= NOT "0000";
158         d      <= "00000";
159     end case;
160 else
161     case m is
162     when "00" =>
163         anTemp <= NOT "0001";
164         d      <= sumL;
165     when "01" =>
166         anTemp <= NOT "0010";
167         d      <= sumH;
168     when "10" =>
169         anTemp <= NOT "0100";
170         d      <= priceL;
171     when "11" =>
172         anTemp <= NOT "1000";
173         d      <= priceH;
174     when others =>
175         anTemp <= NOT "0000";
176         d      <= "00000";
177     end case;
178     end if;
179 end process;
180
181 -----
182 -- if the alarm signal is asserted, the whole display will blink on the
183 -- 3Hz clock. This process runs parallel with the above process, meaning
184 -- that e.g. the alarm signal can be asserted while 'cola' is displayed
185 -----
186 process(alarm, clk_3)
187 begin
188     an <= anTemp;
189     if (alarm = '1' AND clk_3 = '0') then
190         an <= NOT "0000";
191     end if;
192 end process;

```

Figure 7: Page 4

```

193
194 -----
195 -- this process assigns each hexadecimal number (0-F) as well
196 -- as some additional letters to different values of d.
197 -----
198 process(d)
199 begin
200   case d is
201     when "00000" => led <= NOT "11111100"; -- 0
202     when "00001" => led <= NOT "01100000"; -- 1
203     when "00010" => led <= NOT "11011010"; -- 2
204     when "00011" => led <= NOT "11110010"; -- 3
205     when "00100" => led <= NOT "01100110"; -- 4
206     when "00101" => led <= NOT "10110110"; -- 5
207     when "00110" => led <= NOT "10111110"; -- 6
208     when "00111" => led <= NOT "11100000"; -- 7
209     when "01000" => led <= NOT "11111110"; -- 8
210     when "01001" => led <= NOT "11100110"; -- 9
211     when "01010" => led <= NOT "11101110"; -- A
212     when "01011" => led <= NOT "00111110"; -- b
213     when "01100" => led <= NOT "10011100"; -- C
214     when "01101" => led <= NOT "01111010"; -- d
215     when "01110" => led <= NOT "10011110"; -- E
216     when "01111" => led <= NOT "10001110"; -- F
217     when "10000" => led <= NOT "01101110"; -- H
218     when "10001" => led <= NOT "10110110"; -- S
219     when "10010" => led <= NOT "00011100"; -- L
220     when "10011" => led <= NOT "01111100"; -- U
221     when others => led <= (others => '0');
222   end case;
223 end process;
224
225 -----
226 -- the two bit vector m is increased by one on
227 -- each positive clock edge of the 762Hz clock
228 -----
229 process(clock)
230 begin
231   if rising_edge(clock) then
232     m <= m_next;
233   end if;
234 end process;
235 end Behavior;
236

```

Figure 8: Page 5

/Users/hbm/Dropbox/DTU-Amigos/0...hine/src/input_synchronizer.vhd Page 1 of 4
 Saved: 5/2/13 11:53:51 AM Printed For: Hjalte Bested Møller

```

1  -----
2  -- This component synchronizes input signals produced from outside the
3  -- board with the clock signal on the board. This is done to avoid
4  -- metastability. Also this component makes sure that these signals will
5  -- only be asserted for one clock cycle each, even if the input signals
6  -- are asserted from outside the board for several clock cycles.
7  -----
8
9  library ieee;
10 use ieee.std_logic_1164.all;
11
12 entity input_synchronizer is
13   port(clock      : in  std_logic;
14         buy_btn   : in  std_logic;
15         buy_out    : out std_logic;
16         coin1_btn  : in  std_logic;
17         coin1_out  : out std_logic;
18         coin2_btn  : in  std_logic;
19         coin2_out  : out std_logic;
20         coin5_btn  : in  std_logic;
21         coin5_out  : out std_logic;
22         cola_sw    : in  std_logic;
23         cola_out   : out std_logic;
24         hash_sw    : in  std_logic;
25         hash_out   : out std_logic;
26         aqua_sw    : in  std_logic;
27         aqua_out   : out std_logic;
28         Reset      : in  std_logic);
29 end input_synchronizer;
30
31 architecture Structure of input_synchronizer is
32   signal buy_btn_synk, buy_btn_synk_new : std_logic;
33   signal coin1_btn_synk, coin1_btn_synk_new : std_logic;
34   signal coin2_btn_synk, coin2_btn_synk_new : std_logic;
35   signal coin5_btn_synk, coin5_btn_synk_new : std_logic;
36   signal cola_sw_synk, cola_sw_synk_new : std_logic;
37   signal hash_sw_synk, hash_sw_synk_new : std_logic;
38   signal aqua_sw_synk, aqua_sw_synk_new : std_logic;
39
40 begin
41
42   -----
43   -- the seven processes below asserts the signal_out only when
44   -- the synkd version of the input signal is '1' and the synk_new
45   -- version of the input signal is '0'. This makes signal_out
46   -- last for exactly one clock cycle.
47   -----
48

```

Figure 9: VHDL code for the input synchronizer which synchronizes the input signals to the clock

/Users/hbm/Dropbox/DTU-Amigos/0...hine/src/input_synchronizer.vhd Page 2 of 4
Saved: 5/2/13 11:53:51 AM Printed For: Hjalte Bested Møller

```
49  -- buy button
50  process(buy_btn_synk)
51  begin
52      buy_out <= '0';
53      if buy_btn_synk = '1' AND buy_btn_synk_new = '0' then
54          buy_out <= '1';
55      end if;
56  end process;
57
58  -- coin1 button
59  process(coin1_btn_synk)
60  begin
61      coin1_out <= '0';
62      if coin1_btn_synk = '1' AND coin1_btn_synk_new = '0' then
63          coin1_out <= '1';
64      end if;
65  end process;
66
67  -- coin2 button
68  process(coin2_btn_synk)
69  begin
70      coin2_out <= '0';
71      if (coin2_btn_synk = '1' AND coin2_btn_synk_new = '0') then
72          coin2_out <= '1';
73      end if;
74  end process;
75
76  -- coin5 button
77  process(coin5_btn_synk)
78  begin
79      coin5_out <= '0';
80      if coin5_btn_synk = '1' AND coin5_btn_synk_new = '0' then
81          coin5_out <= '1';
82      end if;
83  end process;
84
85  -- cola switch
86  process(cola_sw_synk)
87  begin
88      cola_out <= '0';
89      if cola_sw_synk = '1' AND cola_sw_synk_new = '0' then
90          cola_out <= '1';
91      end if;
92  end process;
93
94  -- hash switch
95  process(hash_sw_synk)
96  begin
```

Figure 10: Page 2

/Users/hbm/Dropbox/DTU-Amigos/0...hine/src/input_synchronizer.vhd Page 3 of 4
 Saved: 5/2/13 11:53:51 AM Printed For: Hjalte Bested Møller

```

97     hash_out <= '0';
98     if hash_sw_synk = '1' AND hash_sw_synk_new = '0' then
99         hash_out <= '1';
100    end if;
101 end process;
102
103 -- aqua switch
104 process(aqua_sw_synk)
105 begin
106     aqua_out <= '0';
107     if aqua_sw_synk = '1' AND aqua_sw_synk_new = '0' then
108         aqua_out <= '1';
109     end if;
110 end process;
111
112 -----
113 -- This process keeps updating registers
114 -- on every rising clock edge
115 -----
116 process(clock)
117 begin
118     if rising_edge(clock) then
119         -- buy button
120         buy_btn_synk      <= buy_btn;
121         buy_btn_synk_new <= buy_btn_synk;
122
123         -- coin1 button
124         coin1_btn_synk    <= coin1_btn;
125         coin1_btn_synk_new <= coin1_btn_synk;
126
127         -- coin2 button
128         coin2_btn_synk    <= coin2_btn;
129         coin2_btn_synk_new <= coin2_btn_synk;
130
131         -- coin5 button
132         coin5_btn_synk    <= coin5_btn;
133         coin5_btn_synk_new <= coin5_btn_synk;
134
135         -- cola switch
136         cola_sw_synk      <= cola_sw;
137         cola_sw_synk_new <= cola_sw_synk;
138
139         -- hash switch
140         hash_sw_synk      <= hash_sw;
141         hash_sw_synk_new <= hash_sw_synk;
142
143         -- aqua switch
144         aqua_sw_synk      <= aqua_sw;

```

Figure 11: Page 3

/Users/hbm/Dropbox/DTU-Amigos/0...hine/src/input_synchronizer.vhd Page 4 of 4
Saved: 5/2/13 11:53:51 AM Printed For: Hjalte Bested Møller

```
145         aqua_sw_synk_new <= aqua_sw_synk;  
146         end if;  
147     end process;  
148  
149 end Structure;
```

Figure 12: Page 4

/Users/hbm/Dropbox/DTU-Amigos/0...Machine/src/processing_unit.vhd Page 1 of 3
 Saved: 5/5/13 8:27:12 PM Printed For: Hjalte Bested Møller

```

1  -----
2  -- This component is the central processing unit of the vending machine.
3  -- This includes the following operations:
4  --     When one of the coin inputs are asserted, sum is updated
5  --     When one of the price_product inputs are asserted, price is update
6  --     If a buy is attempted:
7  --         sum will be deducted from price if sum >= price
8  --         alarm signal will be asserted if sum < price
9  -----
10
11 library ieee;
12 use ieee.std_logic_1164.all;
13 use ieee.numeric_std.all;
14
15 entity processing_unit is
16     port(clock      : in  std_logic;    -- Clock signal in 762Hz
17           clk_3      : in  std_logic;    -- Clock signal in 3Hz
18           buy        : in  std_logic;
19           coin1      : in  std_logic;
20           coin2      : in  std_logic;
21           coin5      : in  std_logic;
22           price_cola : in  std_logic;
23           price_hash : in  std_logic;
24           price_aqua : in  std_logic;
25           Reset      : in  std_logic;
26           sum_out    : out unsigned(5 downto 0);
27           price_out   : out unsigned(5 downto 0);
28           alarm_out   : out std_logic;
29           cola_out    : out std_logic;
30           hash_out    : out std_logic;
31           aqua_out    : out std_logic);
32 end processing_unit;
33
34 architecture Behavioral of processing_unit is
35     signal sum, price : unsigned(5 downto 0);
36     signal alarm_count, alarm_count_next : unsigned(10 downto 0);
37     signal alarm, cola, hash, aqua : std_logic;
38     signal cola_count, cola_count_next : unsigned(10 downto 0);
39     signal hash_count, hash_count_next : unsigned(10 downto 0);
40     signal aqua_count, aqua_count_next : unsigned(10 downto 0);
41
42 begin
43     alarm_count_next <= alarm_count + 1;
44     cola_count_next <= cola_count + 1;
45     hash_count_next <= hash_count + 1;
46     aqua_count_next <= aqua_count + 1;
47
48     -----

```

Figure 13: Processing unit, for calculating/setting price/sum

/Users/hbm/Dropbox/DTU-Amigos/0...Machine/src/processing_unit.vhd Page 2 of 3
Saved: 5/5/13 8:27:12 PM Printed For: Hjalte Bested Møller

```
49 -- This process sets the price of the 3 products. When one of the
50 -- price_'product' signals are asserted, the 'product' signal is set
51 -- to '1', which will cause 'product'_count to keep adding + 1 on
52 -- every clock, until its MSB = '1' and the 'product' signal will be
53 -- set back to '0'. For both the price_'product' and the 'product'
54 -- signals, cola overrules hash, which overrules aqua.
55 -----
56
57 process(price_cola, price_hash, price_aqua, clock)
58 begin
59     if rising_edge(clock) then
60         if price_cola = '1' then
61             cola <= '1';
62             price <= "010010";
63         elsif price_hash = '1' then
64             hash <= '1';
65             price <= "110111";
66         elsif price_aqua = '1' then
67             aqua <= '1';
68             price <= "001100";
69         elsif cola = '1' then
70             cola_count <= cola_count_next;
71             if cola_count(10) = '1' then
72                 cola <= '0';
73                 cola_count <= "0000000000";
74             end if;
75         elsif hash = '1' then
76             hash_count <= hash_count_next;
77             if hash_count(10) = '1' then
78                 hash <= '0';
79                 hash_count <= "0000000000";
80             end if;
81         elsif aqua = '1' then
82             aqua_count <= aqua_count_next;
83             if aqua_count(10) = '1' then
84                 aqua <= '0';
85                 aqua_count <= "0000000000";
86             end if;
87         end if;
88     end if;
89     cola_out <= cola;
90     hash_out <= hash;
91     aqua_out <= aqua;
92 end process;
93
94 -----
95 -- This process adds the coin value to sum when a coin input is
96 -- asserted. If buy is asserted sum will be deducted from price,
```

Figure 14: Page 2

/Users/hbm/Dropbox/DTU-Amigos/0...Machine/src/processing_unit.vhd Page 3 of 3
Saved: 5/5/13 8:27:12 PM Printed For: Hjalte Bested Møller

```
97      -- if sum >= price, else alarm will be set to '1', which causes
98      -- alarm_count to be increased by one on every clock until MSB
99      -- of alarm_count equals '1' which resets alarm to '0'.
100      -----
101
102      process(coin1, coin2, coin5, buy, clock)
103      begin
104          if rising_edge(clock) then
105              if Reset = '1' then
106                  sum <= "000000";
107              elsif coin1 = '1' then
108                  sum <= sum + 1;
109              elsif coin2 = '1' then
110                  sum <= sum + 2;
111              elsif coin5 = '1' then
112                  sum <= sum + 5;
113              elsif alarm = '1' then
114                  alarm_count <= alarm_count_next;
115                  if alarm_count(10) = '1' then
116                      alarm <= '0';
117                      alarm_count <= "0000000000";
118                  end if;
119              elsif buy = '1' then
120                  if sum >= price then
121                      sum <= sum - price;
122                  elsif sum < price then
123                      alarm <= '1';
124                  end if;
125              end if;
126              sum_out <= sum(5 downto 0);
127              price_out <= price(5 downto 0);
128          end if;
129          alarm_out <= alarm;
130      end process;
131
132      end Behavioral;
```

Figure 15: Page 3

/Users/hbm/Dropbox/DTU-Amigos/0...Machine/src/vending_machine.vhd Page 1 of 4
Saved: 5/5/13 7:54:15 PM Printed For: Hjalte Bested Møller

```
1  -----
2  -- Top component of the vending machine for the course:
3  -- 02139 Digital electronics 2 at the Technical University of Denmark
4  --
5  -- This component declares and instantiates all the components of the
6  -- vending machine.
7  -----
8
9  library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.numeric_std.all;
12
13 entity vending_machine is
14     port(clk_50      : in  std_logic;
15          clk_man     : in  std_logic;
16          sel_man     : in  std_logic;
17          reset       : in  std_logic;
18          coin1_btn   : in  std_logic;
19          coin2_btn   : in  std_logic;
20          coin5_btn   : in  std_logic;
21          buy_btn     : in  std_logic;
22          cola_sw     : in  std_logic;
23          hash_sw     : in  std_logic;
24          aqua_sw     : in  std_logic;
25          an          : out std_logic_vector(3 downto 0);
26          led         : out std_logic_vector(1 to 8));
27 end vending_machine;
28
29 architecture struct of vending_machine is
30     signal clk                : std_logic;
31     signal clk_3              : std_logic;
32     signal buy                : std_logic;
33     signal coin1, coin2, coin5 : std_logic;
34     signal price_cola, price_hash, price_aqua : std_logic;
35     signal alarm, cola, aqua, hash : std_logic;
36     signal sum, price          : unsigned(5 downto 0);
37
38     -----
39     -- Component declarations
40     -----
41     component clock_manager is
42         port(clk_50 : in  std_logic;
43              clk_man : in  std_logic;
44              sel_man : in  std_logic;
45              clk     : out std_logic;
46              clk_3   : out std_logic);
47     end component;
48
```

Figure 16: The complete vending machine, assembling all the components

/Users/hbm/Dropbox/DTU-Amigos/0...Machine/src/vending_machine.vhd Page 2 of 4
 Saved: 5/5/13 7:54:15 PM Printed For: Hjalte Bested Møller

```

49     component display_driver is
50     port(price : in unsigned(5 downto 0);
51           sum  : in unsigned(5 downto 0);
52           an   : out std_logic_vector(3 downto 0));
53     reset : in std_logic;
54     clock : in std_logic;
55     clk_3 : in std_logic;
56     alarm : in std_logic;
57     cola  : in std_logic;
58     hash  : in std_logic;
59     aqua  : in std_logic;
60     led   : out std_logic_vector(1 to 8));
61   end component;
62
63   component input_synchronizer is
64   port(clock      : in std_logic;
65         buy_btn   : in std_logic;
66         buy_out    : out std_logic;
67         coin1_btn  : in std_logic;
68         coin1_out  : out std_logic;
69         coin2_btn  : in std_logic;
70         coin2_out  : out std_logic;
71         coin5_btn  : in std_logic;
72         coin5_out  : out std_logic;
73         cola_sw    : in std_logic;
74         cola_out   : out std_logic;
75         hash_sw    : in std_logic;
76         hash_out   : out std_logic;
77         aqua_sw    : in std_logic;
78         aqua_out   : out std_logic;
79         Reset      : in std_logic);
80
81   end component;
82
83   component processing_unit is
84   port(clock      : in std_logic;
85         buy        : in std_logic;
86         coin1      : in std_logic;
87         coin2      : in std_logic;
88         coin5      : in std_logic;
89         price_cola : in std_logic;
90         price_hash : in std_logic;
91         price_aqua : in std_logic;
92         Reset      : in std_logic;
93         sum_out    : out unsigned(5 downto 0);
94         price_out  : out unsigned(5 downto 0);
95         alarm_out  : out std_logic;
96         cola_out   : out std_logic;

```

Figure 17: Page 2

/Users/hbm/Dropbox/DTU-Amigos/0...Machine/src/vending_machine.vhd Page 3 of 4
Saved: 5/5/13 7:54:15 PM Printed For: Hjalte Bested Møller

```

97         hash_out    : out std_logic;
98         aqua_out     : out std_logic);
99     end component;
100
101 -----
102 -- signal assignments
103 -----
104
105 begin
106     clock_manager1 : clock_manager
107         port map(clk_50 => clk_50,
108                 clk_man => clk_man,
109                 sel_man => sel_man,
110                 clk     => clk,
111                 clk_3   => clk_3);
112
113     display_driver1 : display_driver
114         port map(sum  => sum,
115                 price => price,
116                 an    => an,
117                 reset => reset,
118                 clock => clk,
119                 clk_3 => clk_3,
120                 alarm => alarm,
121                 cola  => cola,
122                 hash  => hash,
123                 aqua  => aqua,
124                 led   => led);
125
126     input_synchronizer1 : input_synchronizer
127         port map(clock    => clk,
128                 buy_btn   => buy_btn,
129                 buy_out   => buy,
130                 coin1_btn => coin1_btn,
131                 coin1_out => coin1,
132                 coin2_btn => coin2_btn,
133                 coin2_out => coin2,
134                 coin5_btn => coin5_btn,
135                 coin5_out => coin5,
136                 cola_sw  => cola_sw,
137                 cola_out => price_cola,
138                 hash_sw  => hash_sw,
139                 hash_out => price_hash,
140                 aqua_sw  => aqua_sw,
141                 aqua_out => price_aqua,
142                 Reset    => reset);
143
144     processing_unit1 : processing_unit
```

Figure 18: Page 3

/Users/hbm/Dropbox/DTU-Amigos/0...Machine/src/vending_machine.vhd Page 4 of 4
Saved: 5/5/13 7:54:15 PM Printed For: Hjalte Bested Møller

```
145     port map(clock      => clk,  
146               buy       => buy,  
147               coin1     => coin1,  
148               coin2     => coin2,  
149               coin5     => coin5,  
150               price_cola => price_cola,  
151               price_hash => price_hash,  
152               price_aqua => price_aqua,  
153               Reset     => reset,  
154               sum_out    => sum,  
155               price_out  => price,  
156               alarm_out  => alarm,  
157               cola_out   => cola,  
158               hash_out   => hash,  
159               aqua_out   => aqua);  
160  
161 end struct;  
162
```

Figure 19: Page 4