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 mircocontroller.

Vi har konkluderet at den skrevne kode implementerer det ønskede kredsløb på Mircrocontrolleren, 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 Mircrocontroller.

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 a 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.

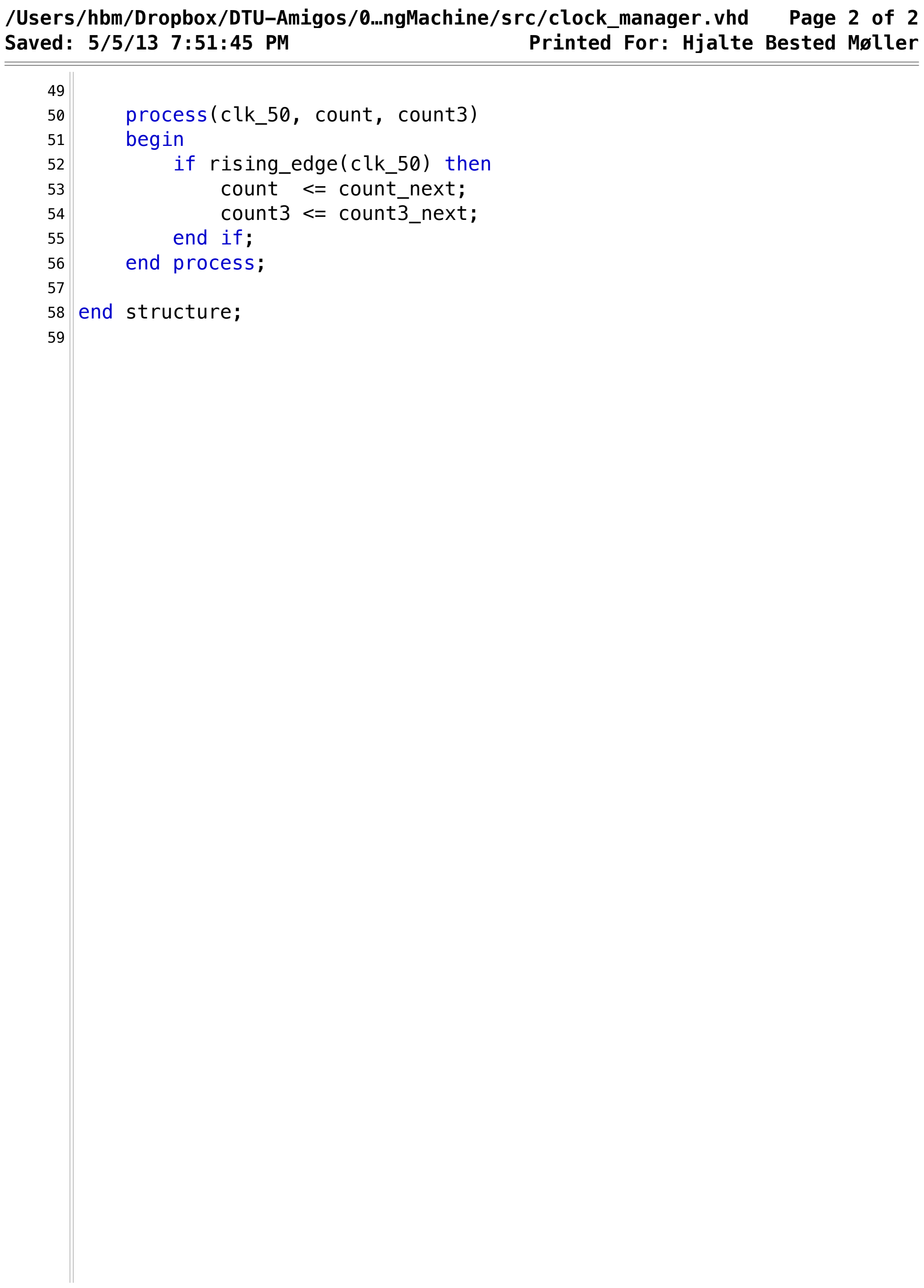
A Appendiks 1

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

/Users/hbm/Dropbox/DTU-Amigos/0…ngMachine/src/clock_manager.vhd   Page 1 of 2
Saved: 5/5/13 7:51:45 PM   Printed For: Hjalte Bested Møller

```vhdl
----------------------------------------------------------------------
-- This component divides the 50 MHz clock signal down to a clk signal
-- of 762Hz and a clk_3 signal of 3Hz.
----------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clock_manager is
    port(clk_50  : in  std_logic;
         clk_man : in  std_logic;
         sel_man : in  std_logic;
         clk     : out std_logic;
         clk_3   : out std_logic);
end clock_manager;

architecture structure of clock_manager is
    signal count, count_next   : unsigned(15 downto 0) := (others => '0');
    signal count3, count3_next : unsigned(23 downto 0) := (others => '0');
    attribute clock_signal : string;
    attribute clock_signal of clk : signal is "yes";

begin
    count_next  <= count + 1;
    count3_next <= count3 + 1;

-------------------------------------------------------------------------
-- this process generates the clk and clk3 signal, which are derived from
-- the 50MHz clock. When MSB of count = '1' clk will go high, when MSB of
-- count = '0' clk will go low. Same with count3 and clk_3. If sel_man is
-- set to 1, clk will be set to the input signal clk_man.
-------------------------------------------------------------------------

    process(sel_man, clk_man, count, count3)
    begin
        if sel_man = '1' then
            clk <= clk_man;
        else
            clk <= count(15);
        end if;
        clk_3 <= count3(23);
    end process;

--------------------------------------------
-- on the rising edge of the 50MHz clock
-- count and count3 will be increased by 1
--------------------------------------------
```

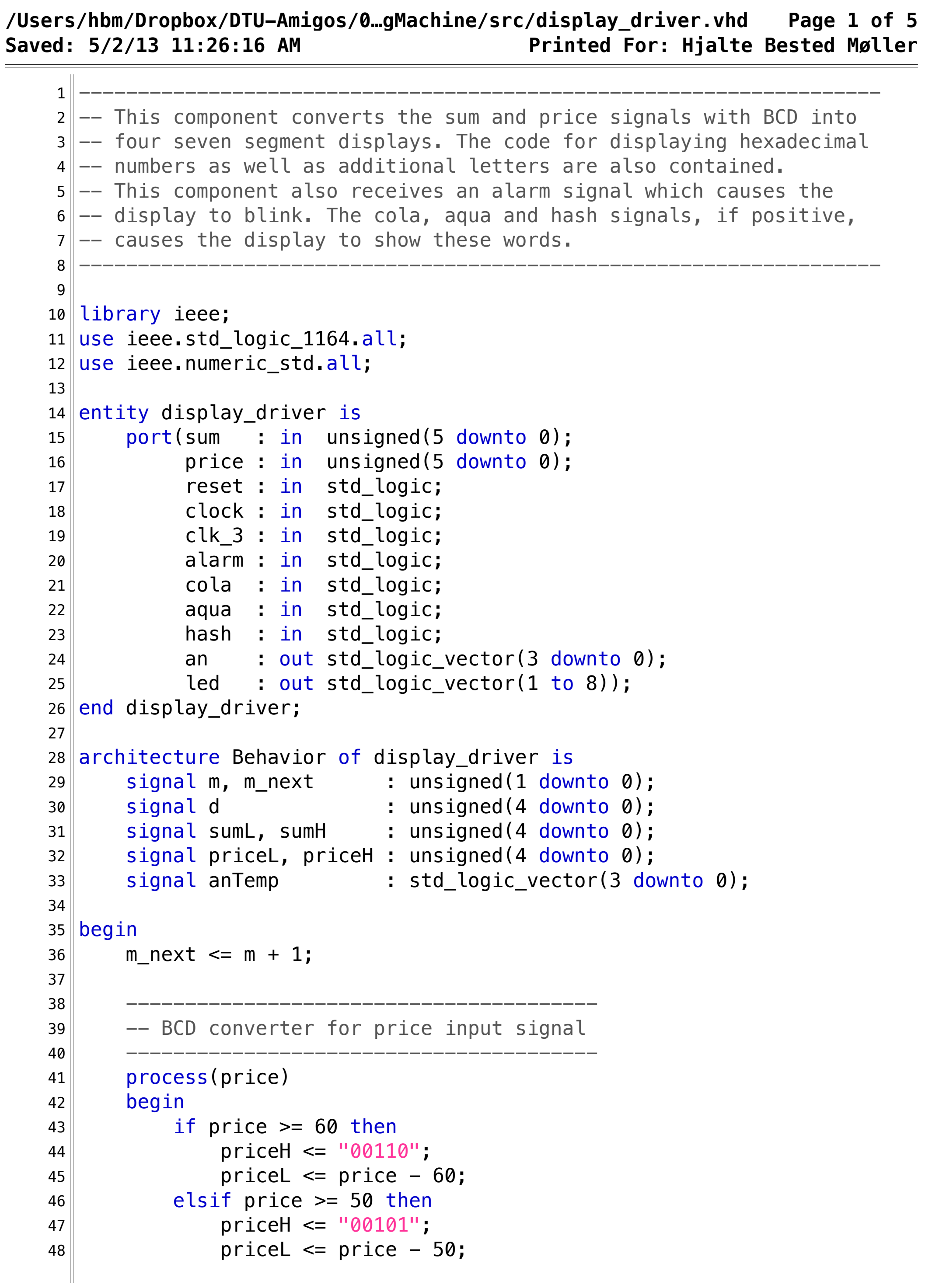
Figure 2: The clock manager, include our own 3 Hz clock for use in the alarm signal



/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

```vhdl

    process(clk_50, count, count3)
    begin
        if rising_edge(clk_50) then
            count  <= count_next;
            count3 <= count3_next;
        end if;
    end process;

end structure;

```

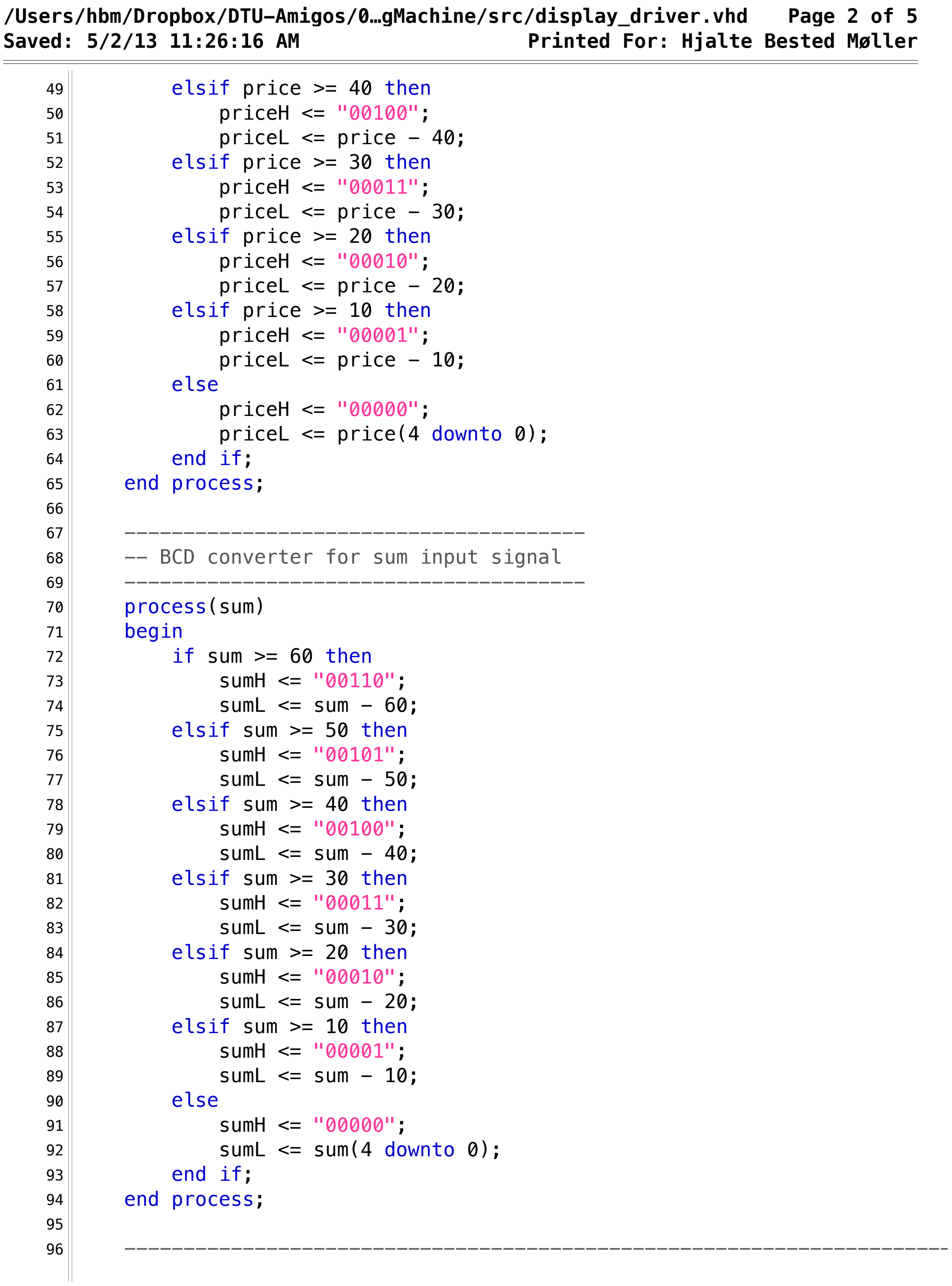
Figure 3: Page 2



/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

```
--------------------------------------------------------------------
-- This component converts the sum and price signals with BCD into
-- four seven segment displays. The code for displaying hexadecimal
-- numbers as well as additional letters are also contained.
-- This component also receives an alarm signal which causes the
-- display to blink. The cola, aqua and hash signals, if positive,
-- causes the display to show these words.
--------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity display_driver is
    port(sum   : in  unsigned(5 downto 0);
         price : in  unsigned(5 downto 0);
         reset : in  std_logic;
         clock : in  std_logic;
         clk_3 : in  std_logic;
         alarm : in  std_logic;
         cola  : in  std_logic;
         aqua  : in  std_logic;
         hash  : in  std_logic;
         an    : out std_logic_vector(3 downto 0);
         led   : out std_logic_vector(1 to 8));
end display_driver;

architecture Behavior of display_driver is
    signal m, m_next      : unsigned(1 downto 0);
    signal d              : unsigned(4 downto 0);
    signal sumL, sumH     : unsigned(4 downto 0);
    signal priceL, priceH : unsigned(4 downto 0);
    signal anTemp         : std_logic_vector(3 downto 0);

begin
    m_next <= m + 1;

    ----------------------------------------
    -- BCD converter for price input signal
    ----------------------------------------
    process(price)
    begin
        if price >= 60 then
            priceH <= "00110";
            priceL <= price - 60;
        elsif price >= 50 then
            priceH <= "00101";
            priceL <= price - 50;
```

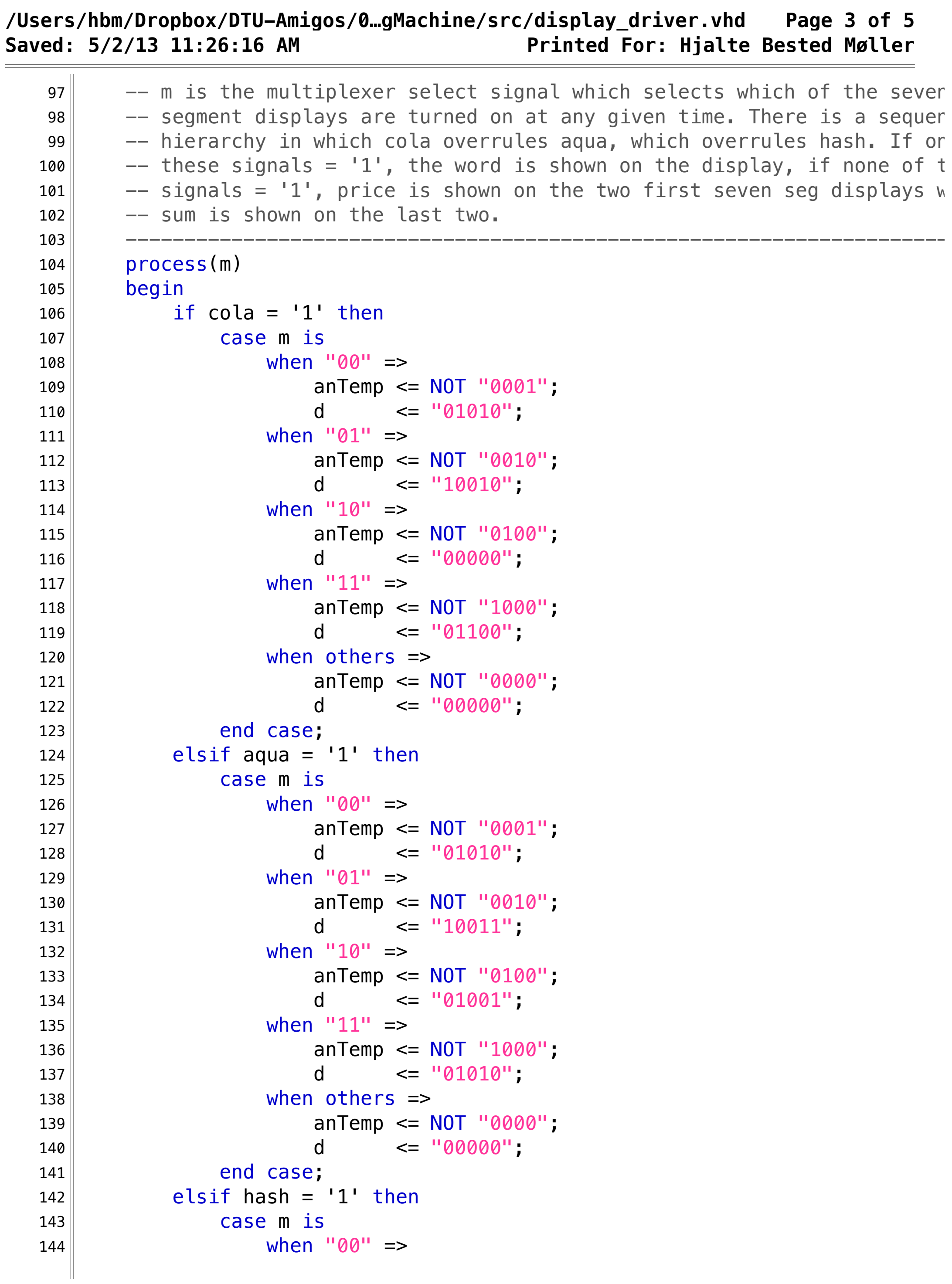
Figure 4: Display driver for displaying symbols/numbers on each of the four seven segment displays



/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

```vhdl
        elsif price >= 40 then
            priceH <= "00100";
            priceL <= price - 40;
        elsif price >= 30 then
            priceH <= "00011";
            priceL <= price - 30;
        elsif price >= 20 then
            priceH <= "00010";
            priceL <= price - 20;
        elsif price >= 10 then
            priceH <= "00001";
            priceL <= price - 10;
        else
            priceH <= "00000";
            priceL <= price(4 downto 0);
        end if;
    end process;

    ---------------------------------------
    -- BCD converter for sum input signal
    ---------------------------------------
    process(sum)
    begin
        if sum >= 60 then
            sumH <= "00110";
            sumL <= sum - 60;
        elsif sum >= 50 then
            sumH <= "00101";
            sumL <= sum - 50;
        elsif sum >= 40 then
            sumH <= "00100";
            sumL <= sum - 40;
        elsif sum >= 30 then
            sumH <= "00011";
            sumL <= sum - 30;
        elsif sum >= 20 then
            sumH <= "00010";
            sumL <= sum - 20;
        elsif sum >= 10 then
            sumH <= "00001";
            sumL <= sum - 10;
        else
            sumH <= "00000";
            sumL <= sum(4 downto 0);
        end if;
    end process;

    ----------------------------------------------------------------------
```

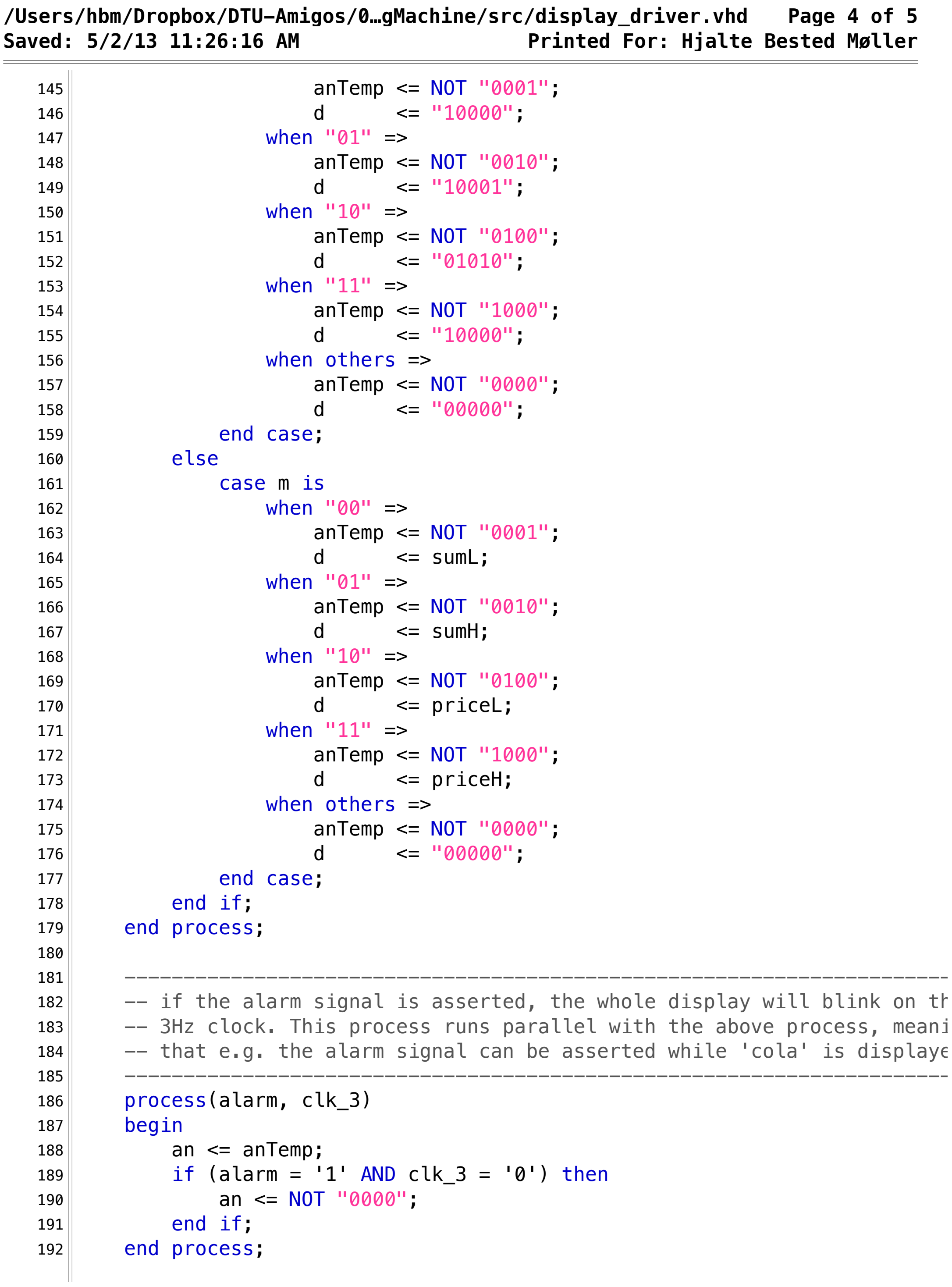
Figure 5: Page 2



/Users/hbm/Dropbox/DTU-Amigos/0…gMachine/src/display_driver.vhd   Page 3 of 5
Saved: 5/2/13 11:26:16 AM   Printed For: Hjalte Bested Møller

```
-- m is the multiplexer select signal which selects which of the sever
-- segment displays are turned on at any given time. There is a sequer
-- hierarchy in which cola overrules aqua, which overrules hash. If or
-- these signals = '1', the word is shown on the display, if none of t
-- signals = '1', price is shown on the two first seven seg displays w
-- sum is shown on the last two.
-------------------------------------------------------------------
process(m)
begin
    if cola = '1' then
        case m is
            when "00" =>
                anTemp <= NOT "0001";
                d      <= "01010";
            when "01" =>
                anTemp <= NOT "0010";
                d      <= "10010";
            when "10" =>
                anTemp <= NOT "0100";
                d      <= "00000";
            when "11" =>
                anTemp <= NOT "1000";
                d      <= "01100";
            when others =>
                anTemp <= NOT "0000";
                d      <= "00000";
        end case;
    elsif aqua = '1' then
        case m is
            when "00" =>
                anTemp <= NOT "0001";
                d      <= "01010";
            when "01" =>
                anTemp <= NOT "0010";
                d      <= "10011";
            when "10" =>
                anTemp <= NOT "0100";
                d      <= "01001";
            when "11" =>
                anTemp <= NOT "1000";
                d      <= "01010";
            when others =>
                anTemp <= NOT "0000";
                d      <= "00000";
        end case;
    elsif hash = '1' then
        case m is
            when "00" =>
```

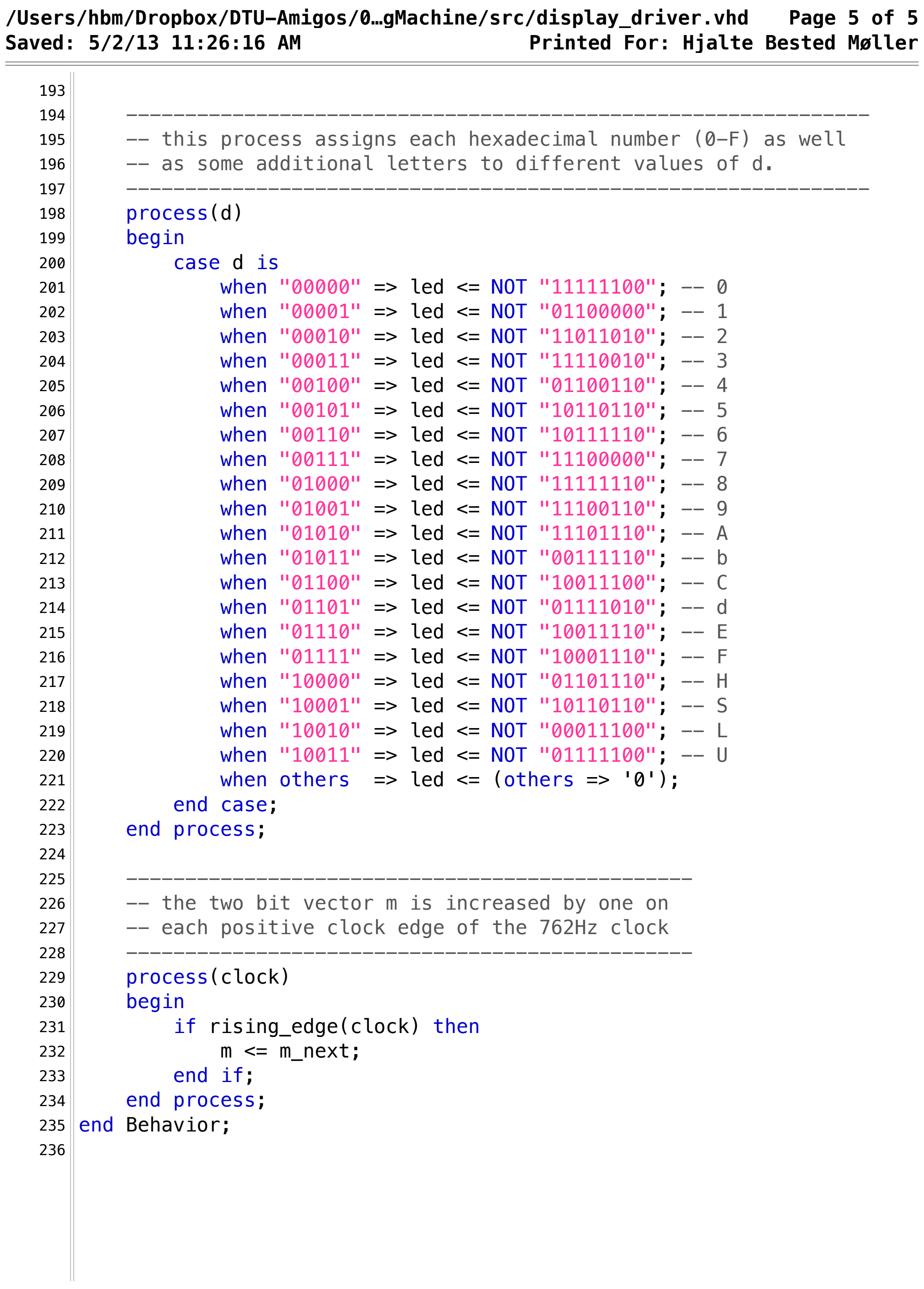
Figure 6: Page 3



/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

```vhdl
                    anTemp <= NOT "0001";
                    d      <= "10000";
                when "01" =>
                    anTemp <= NOT "0010";
                    d      <= "10001";
                when "10" =>
                    anTemp <= NOT "0100";
                    d      <= "01010";
                when "11" =>
                    anTemp <= NOT "1000";
                    d      <= "10000";
                when others =>
                    anTemp <= NOT "0000";
                    d      <= "00000";
            end case;
        else
            case m is
                when "00" =>
                    anTemp <= NOT "0001";
                    d      <= sumL;
                when "01" =>
                    anTemp <= NOT "0010";
                    d      <= sumH;
                when "10" =>
                    anTemp <= NOT "0100";
                    d      <= priceL;
                when "11" =>
                    anTemp <= NOT "1000";
                    d      <= priceH;
                when others =>
                    anTemp <= NOT "0000";
                    d      <= "00000";
            end case;
        end if;
    end process;

    ----------------------------------------------------------------------
    -- if the alarm signal is asserted, the whole display will blink on th
    -- 3Hz clock. This process runs parallel with the above process, meani
    -- that e.g. the alarm signal can be asserted while 'cola' is displaye
    ----------------------------------------------------------------------
    process(alarm, clk_3)
    begin
        an <= anTemp;
        if (alarm = '1' AND clk_3 = '0') then
            an <= NOT "0000";
        end if;
    end process;
```

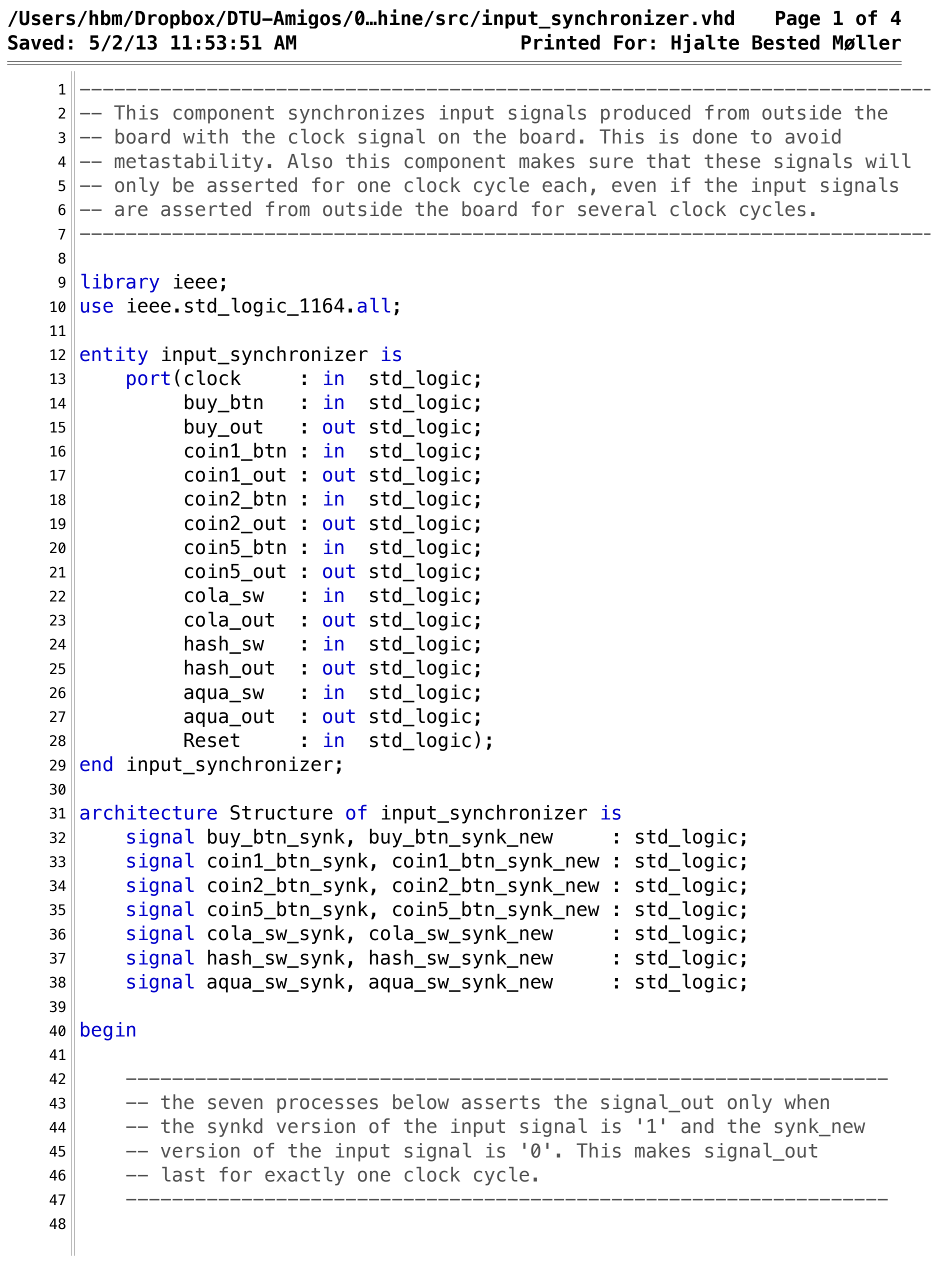
Figure 7: Page 4



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

```vhdl

    ---------------------------------------------------------------
    -- this process assigns each hexadecimal number (0-F) as well
    -- as some additional letters to different values of d.
    ---------------------------------------------------------------
    process(d)
    begin
        case d is
            when "00000" => led <= NOT "11111100"; -- 0
            when "00001" => led <= NOT "01100000"; -- 1
            when "00010" => led <= NOT "11011010"; -- 2
            when "00011" => led <= NOT "11110010"; -- 3
            when "00100" => led <= NOT "01100110"; -- 4
            when "00101" => led <= NOT "10110110"; -- 5
            when "00110" => led <= NOT "10111110"; -- 6
            when "00111" => led <= NOT "11100000"; -- 7
            when "01000" => led <= NOT "11111110"; -- 8
            when "01001" => led <= NOT "11100110"; -- 9
            when "01010" => led <= NOT "11101110"; -- A
            when "01011" => led <= NOT "00111110"; -- b
            when "01100" => led <= NOT "10011100"; -- C
            when "01101" => led <= NOT "01111010"; -- d
            when "01110" => led <= NOT "10011110"; -- E
            when "01111" => led <= NOT "10001110"; -- F
            when "10000" => led <= NOT "01101110"; -- H
            when "10001" => led <= NOT "10110110"; -- S
            when "10010" => led <= NOT "00011100"; -- L
            when "10011" => led <= NOT "01111100"; -- U
            when others  => led <= (others => '0');
        end case;
    end process;

    ------------------------------------------------
    -- the two bit vector m is increased by one on
    -- each positive clock edge of the 762Hz clock
    ------------------------------------------------
    process(clock)
    begin
        if rising_edge(clock) then
            m <= m_next;
        end if;
    end process;
end Behavior;

```

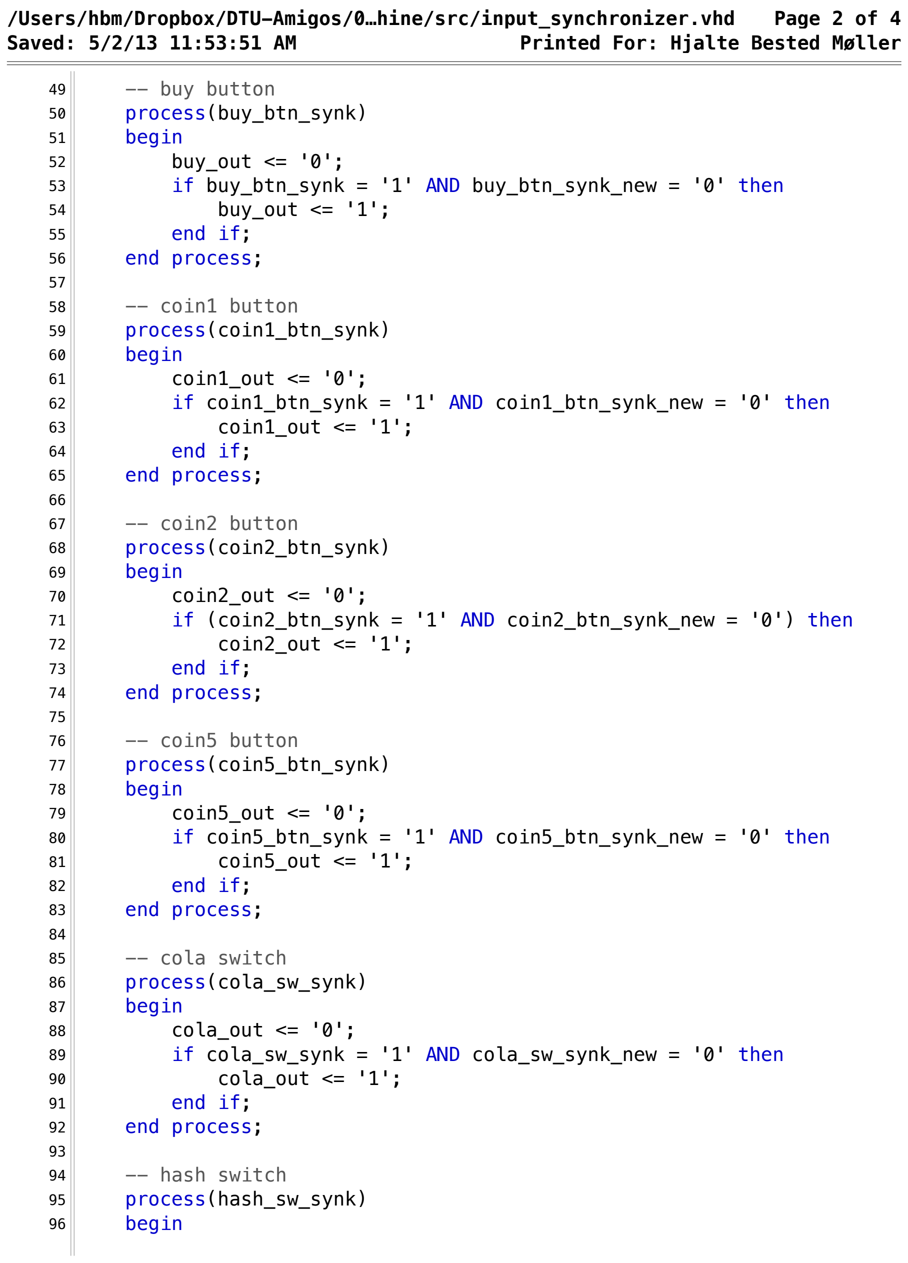
Figure 8: Page 5



/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

```vhdl
--------------------------------------------------------------------------
-- This component synchronizes input signals produced from outside the
-- board with the clock signal on the board. This is done to avoid
-- metastability. Also this component makes sure that these signals will
-- only be asserted for one clock cycle each, even if the input signals
-- are asserted from outside the board for several clock cycles.
--------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity input_synchronizer is
    port(clock      : in  std_logic;
         buy_btn    : in  std_logic;
         buy_out    : out std_logic;
         coin1_btn  : in  std_logic;
         coin1_out  : out std_logic;
         coin2_btn  : in  std_logic;
         coin2_out  : out std_logic;
         coin5_btn  : in  std_logic;
         coin5_out  : out std_logic;
         cola_sw    : in  std_logic;
         cola_out   : out std_logic;
         hash_sw    : in  std_logic;
         hash_out   : out std_logic;
         aqua_sw    : in  std_logic;
         aqua_out   : out std_logic;
         Reset      : in  std_logic);
end input_synchronizer;

architecture Structure of input_synchronizer is
    signal buy_btn_synk, buy_btn_synk_new     : std_logic;
    signal coin1_btn_synk, coin1_btn_synk_new : std_logic;
    signal coin2_btn_synk, coin2_btn_synk_new : std_logic;
    signal coin5_btn_synk, coin5_btn_synk_new : std_logic;
    signal cola_sw_synk, cola_sw_synk_new     : std_logic;
    signal hash_sw_synk, hash_sw_synk_new     : std_logic;
    signal aqua_sw_synk, aqua_sw_synk_new     : std_logic;

begin

    ------------------------------------------------------------------
    -- the seven processes below asserts the signal_out only when
    -- the synkd version of the input signal is '1' and the synk_new
    -- version of the input signal is '0'. This makes signal_out
    -- last for exactly one clock cycle.
    ------------------------------------------------------------------

```

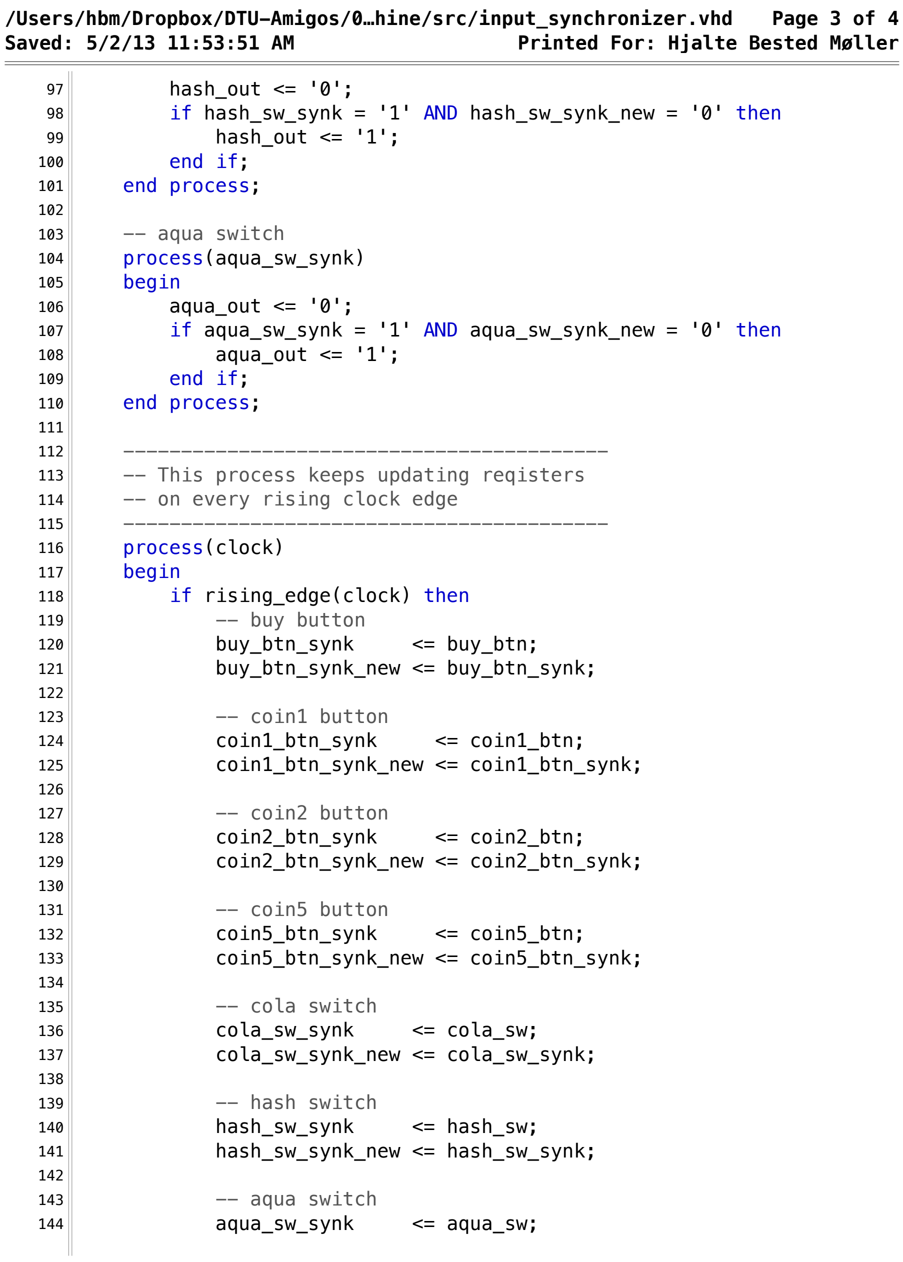
Figure 9: VHDL code for the input synchronizer which synchronizes the input signals to the clock



/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

```
    -- buy button
    process(buy_btn_synk)
    begin
        buy_out <= '0';
        if buy_btn_synk = '1' AND buy_btn_synk_new = '0' then
            buy_out <= '1';
        end if;
    end process;

    -- coin1 button
    process(coin1_btn_synk)
    begin
        coin1_out <= '0';
        if coin1_btn_synk = '1' AND coin1_btn_synk_new = '0' then
            coin1_out <= '1';
        end if;
    end process;

    -- coin2 button
    process(coin2_btn_synk)
    begin
        coin2_out <= '0';
        if (coin2_btn_synk = '1' AND coin2_btn_synk_new = '0') then
            coin2_out <= '1';
        end if;
    end process;

    -- coin5 button
    process(coin5_btn_synk)
    begin
        coin5_out <= '0';
        if coin5_btn_synk = '1' AND coin5_btn_synk_new = '0' then
            coin5_out <= '1';
        end if;
    end process;

    -- cola switch
    process(cola_sw_synk)
    begin
        cola_out <= '0';
        if cola_sw_synk = '1' AND cola_sw_synk_new = '0' then
            cola_out <= '1';
        end if;
    end process;

    -- hash switch
    process(hash_sw_synk)
    begin
```

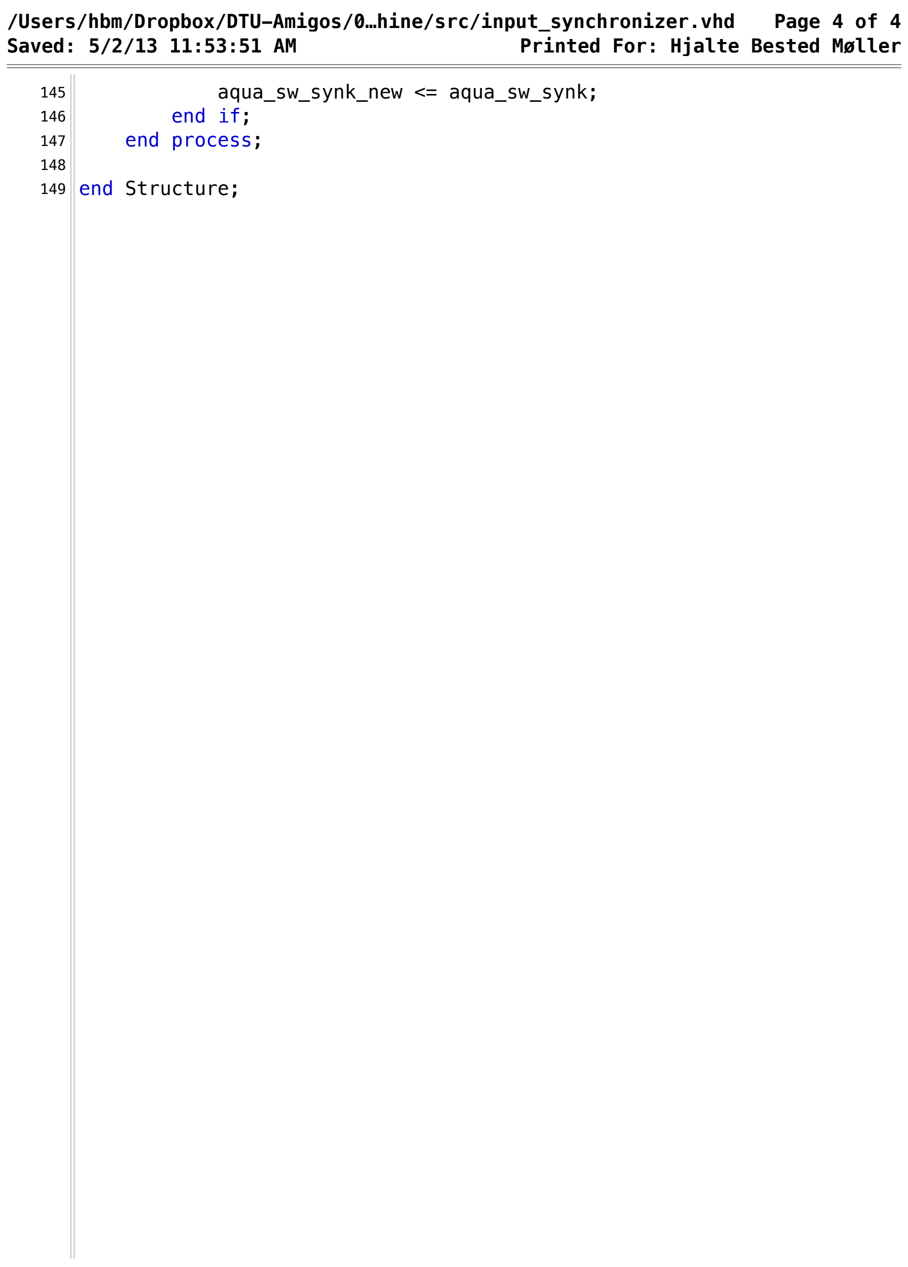
Figure 10: Page 2



/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

```
        hash_out <= '0';
        if hash_sw_synk = '1' AND hash_sw_synk_new = '0' then
            hash_out <= '1';
        end if;
    end process;

    -- aqua switch
    process(aqua_sw_synk)
    begin
        aqua_out <= '0';
        if aqua_sw_synk = '1' AND aqua_sw_synk_new = '0' then
            aqua_out <= '1';
        end if;
    end process;

    ------------------------------------------
    -- This process keeps updating reqisters
    -- on every rising clock edge
    ------------------------------------------
    process(clock)
    begin
        if rising_edge(clock) then
            -- buy button
            buy_btn_synk     <= buy_btn;
            buy_btn_synk_new <= buy_btn_synk;

            -- coin1 button
            coin1_btn_synk     <= coin1_btn;
            coin1_btn_synk_new <= coin1_btn_synk;

            -- coin2 button
            coin2_btn_synk     <= coin2_btn;
            coin2_btn_synk_new <= coin2_btn_synk;

            -- coin5 button
            coin5_btn_synk     <= coin5_btn;
            coin5_btn_synk_new <= coin5_btn_synk;

            -- cola switch
            cola_sw_synk     <= cola_sw;
            cola_sw_synk_new <= cola_sw_synk;

            -- hash switch
            hash_sw_synk     <= hash_sw;
            hash_sw_synk_new <= hash_sw_synk;

            -- aqua switch
            aqua_sw_synk     <= aqua_sw;
```

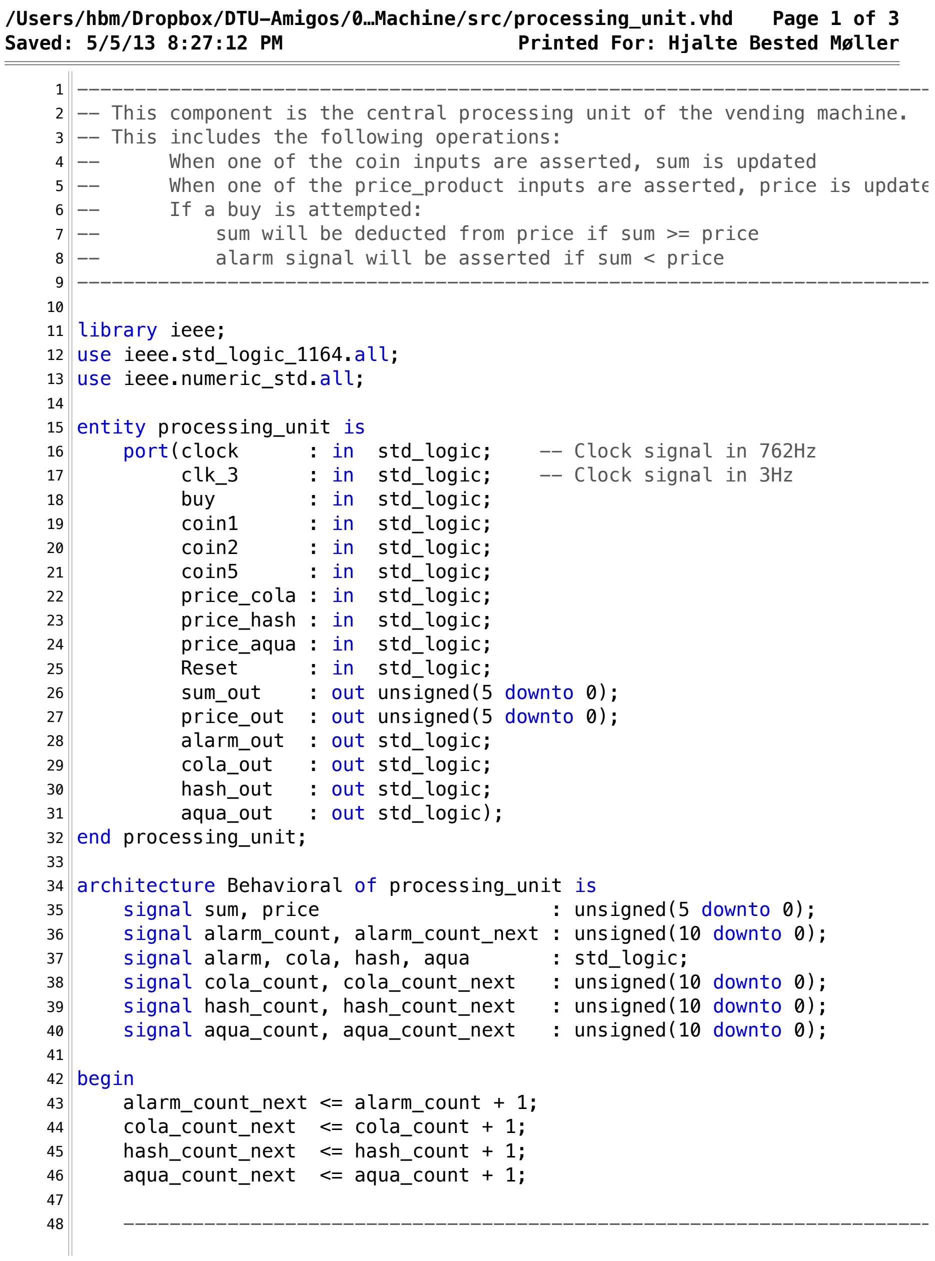
Figure 11: Page 3



/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

```vhdl
            aqua_sw_synk_new <= aqua_sw_synk;
        end if;
    end process;

end Structure;
```

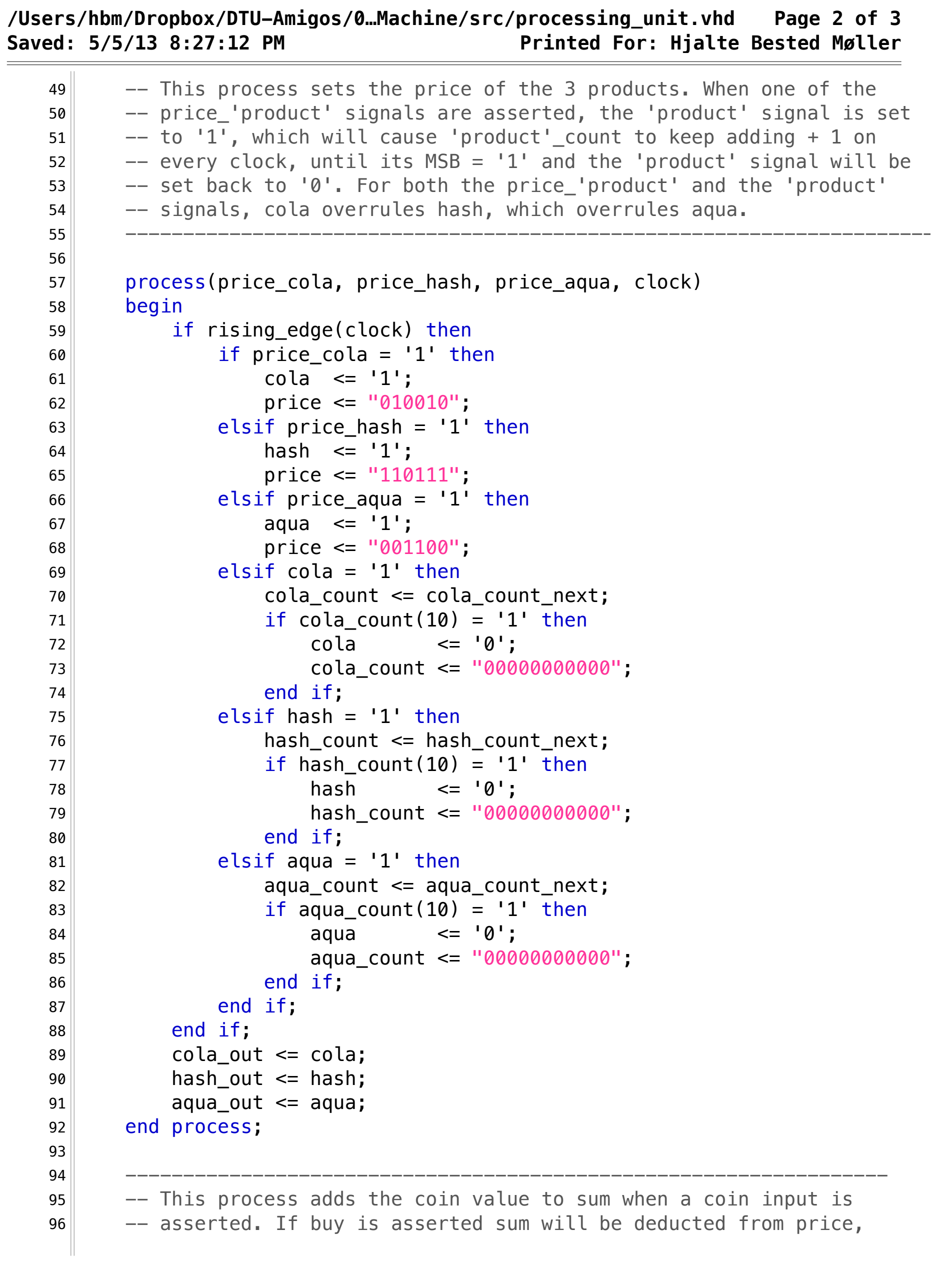
Figure 12: Page 4



/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

```vhdl
------------------------------------------------------------------------
-- This component is the central processing unit of the vending machine.
-- This includes the following operations:
--      When one of the coin inputs are asserted, sum is updated
--      When one of the price_product inputs are asserted, price is update
--      If a buy is attempted:
--          sum will be deducted from price if sum >= price
--          alarm signal will be asserted if sum < price
------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity processing_unit is
    port(clock      : in  std_logic;    -- Clock signal in 762Hz
         clk_3      : in  std_logic;    -- Clock signal in 3Hz
         buy        : in  std_logic;
         coin1      : in  std_logic;
         coin2      : in  std_logic;
         coin5      : in  std_logic;
         price_cola : in  std_logic;
         price_hash : in  std_logic;
         price_aqua : in  std_logic;
         Reset      : in  std_logic;
         sum_out    : out unsigned(5 downto 0);
         price_out  : out unsigned(5 downto 0);
         alarm_out  : out std_logic;
         cola_out   : out std_logic;
         hash_out   : out std_logic;
         aqua_out   : out std_logic);
end processing_unit;

architecture Behavioral of processing_unit is
    signal sum, price                  : unsigned(5 downto 0);
    signal alarm_count, alarm_count_next : unsigned(10 downto 0);
    signal alarm, cola, hash, aqua       : std_logic;
    signal cola_count, cola_count_next   : unsigned(10 downto 0);
    signal hash_count, hash_count_next   : unsigned(10 downto 0);
    signal aqua_count, aqua_count_next   : unsigned(10 downto 0);

begin
    alarm_count_next <= alarm_count + 1;
    cola_count_next  <= cola_count + 1;
    hash_count_next  <= hash_count + 1;
    aqua_count_next  <= aqua_count + 1;

    --------------------------------------------------------------------
```

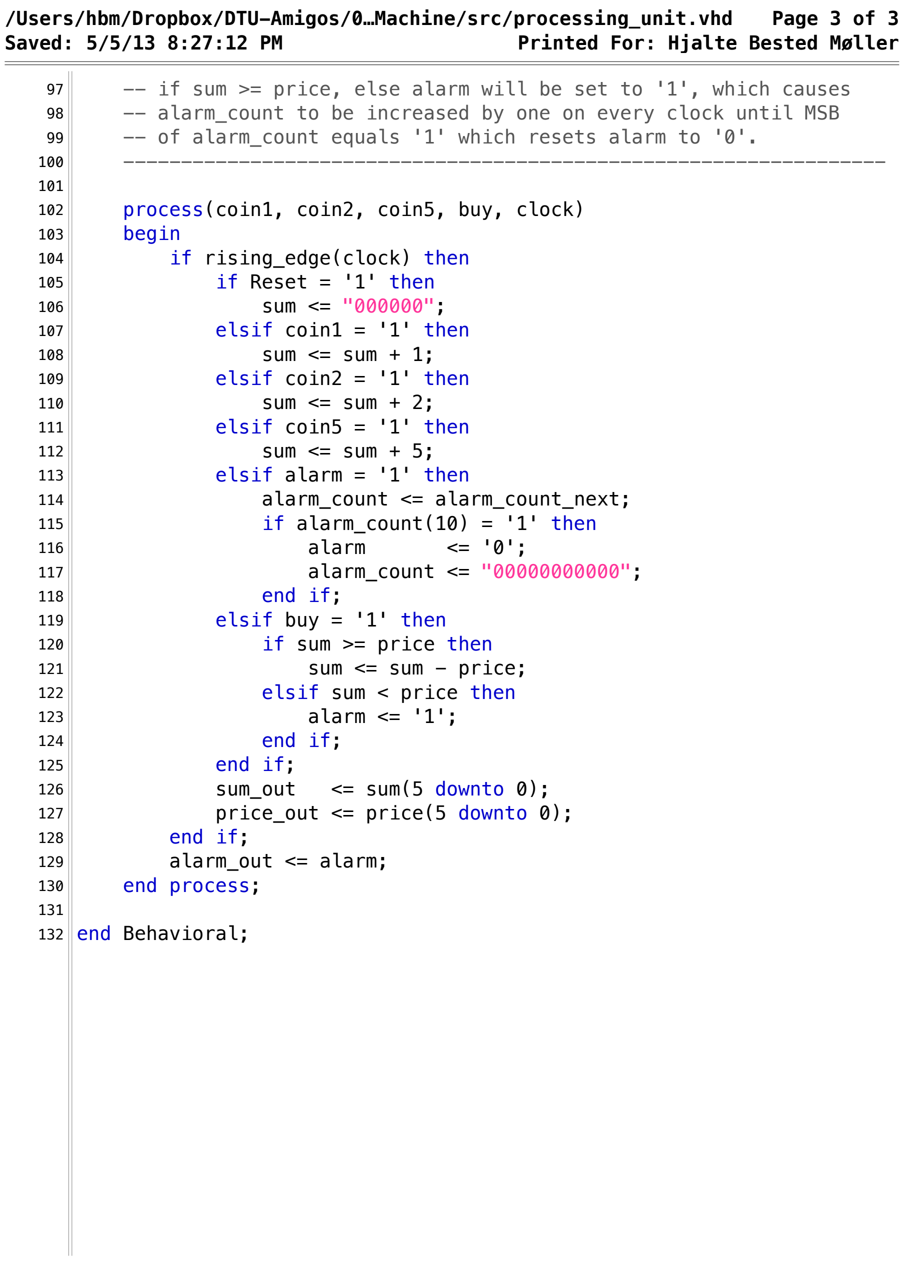
Figure 13: Processing unit, for calculating/setting price/sum



/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

```
-- This process sets the price of the 3 products. When one of the
-- price_'product' signals are asserted, the 'product' signal is set
-- to '1', which will cause 'product'_count to keep adding + 1 on
-- every clock, until its MSB = '1' and the 'product' signal will be
-- set back to '0'. For both the price_'product' and the 'product'
-- signals, cola overrules hash, which overrules aqua.
----------------------------------------------------------------------

process(price_cola, price_hash, price_aqua, clock)
begin
    if rising_edge(clock) then
        if price_cola = '1' then
            cola  <= '1';
            price <= "010010";
        elsif price_hash = '1' then
            hash  <= '1';
            price <= "110111";
        elsif price_aqua = '1' then
            aqua  <= '1';
            price <= "001100";
        elsif cola = '1' then
            cola_count <= cola_count_next;
            if cola_count(10) = '1' then
                cola       <= '0';
                cola_count <= "00000000000";
            end if;
        elsif hash = '1' then
            hash_count <= hash_count_next;
            if hash_count(10) = '1' then
                hash       <= '0';
                hash_count <= "00000000000";
            end if;
        elsif aqua = '1' then
            aqua_count <= aqua_count_next;
            if aqua_count(10) = '1' then
                aqua       <= '0';
                aqua_count <= "00000000000";
            end if;
        end if;
    end if;
    cola_out <= cola;
    hash_out <= hash;
    aqua_out <= aqua;
end process;

------------------------------------------------------------------
-- This process adds the coin value to sum when a coin input is
-- asserted. If buy is asserted sum will be deducted from price,
```

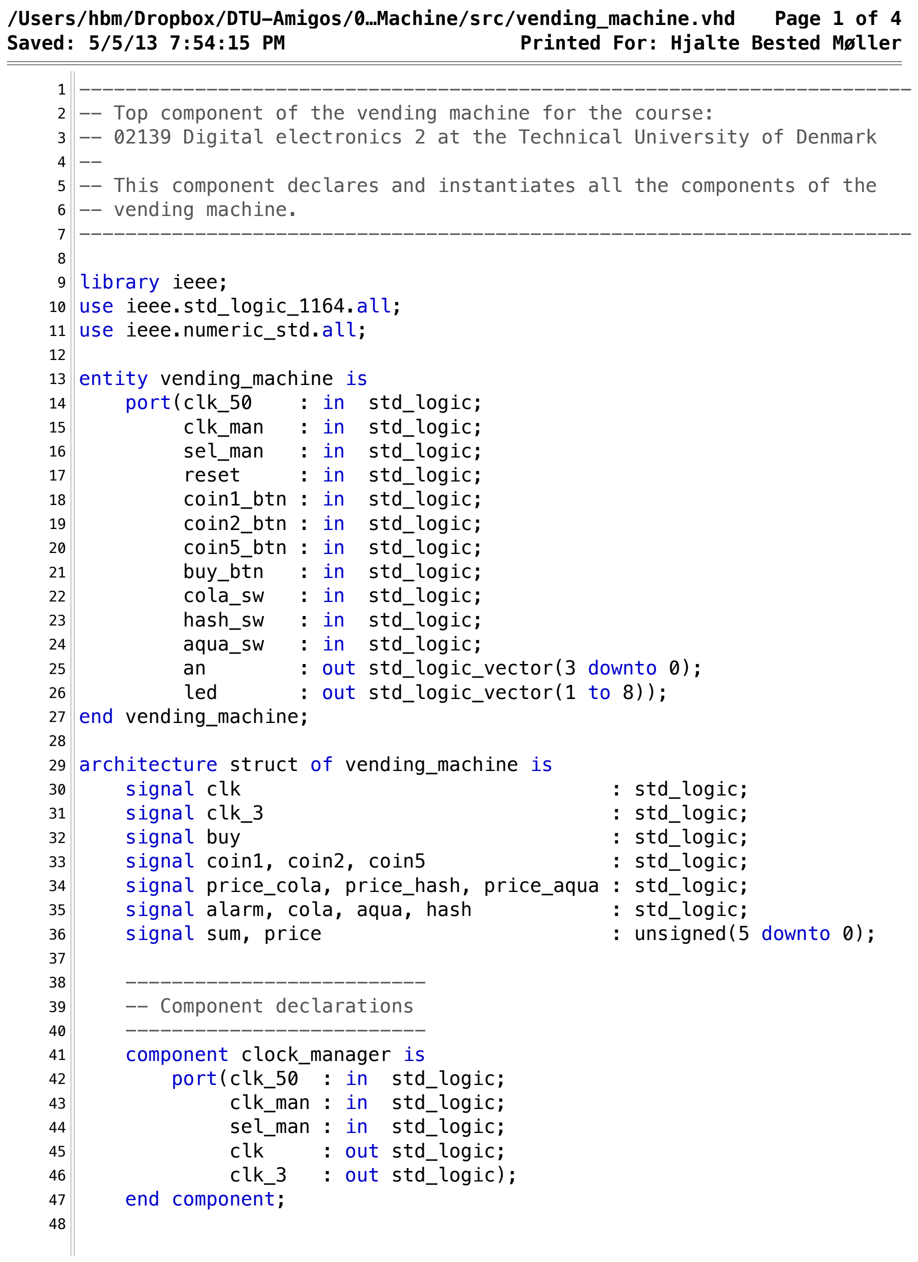
Figure 14: Page 2



/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

```vhdl
    -- if sum >= price, else alarm will be set to '1', which causes
    -- alarm_count to be increased by one on every clock until MSB
    -- of alarm_count equals '1' which resets alarm to '0'.
    ------------------------------------------------------------------

    process(coin1, coin2, coin5, buy, clock)
    begin
        if rising_edge(clock) then
            if Reset = '1' then
                sum <= "000000";
            elsif coin1 = '1' then
                sum <= sum + 1;
            elsif coin2 = '1' then
                sum <= sum + 2;
            elsif coin5 = '1' then
                sum <= sum + 5;
            elsif alarm = '1' then
                alarm_count <= alarm_count_next;
                if alarm_count(10) = '1' then
                    alarm       <= '0';
                    alarm_count <= "00000000000";
                end if;
            elsif buy = '1' then
                if sum >= price then
                    sum <= sum - price;
                elsif sum < price then
                    alarm <= '1';
                end if;
            end if;
            sum_out   <= sum(5 downto 0);
            price_out <= price(5 downto 0);
        end if;
        alarm_out <= alarm;
    end process;

end Behavioral;
```

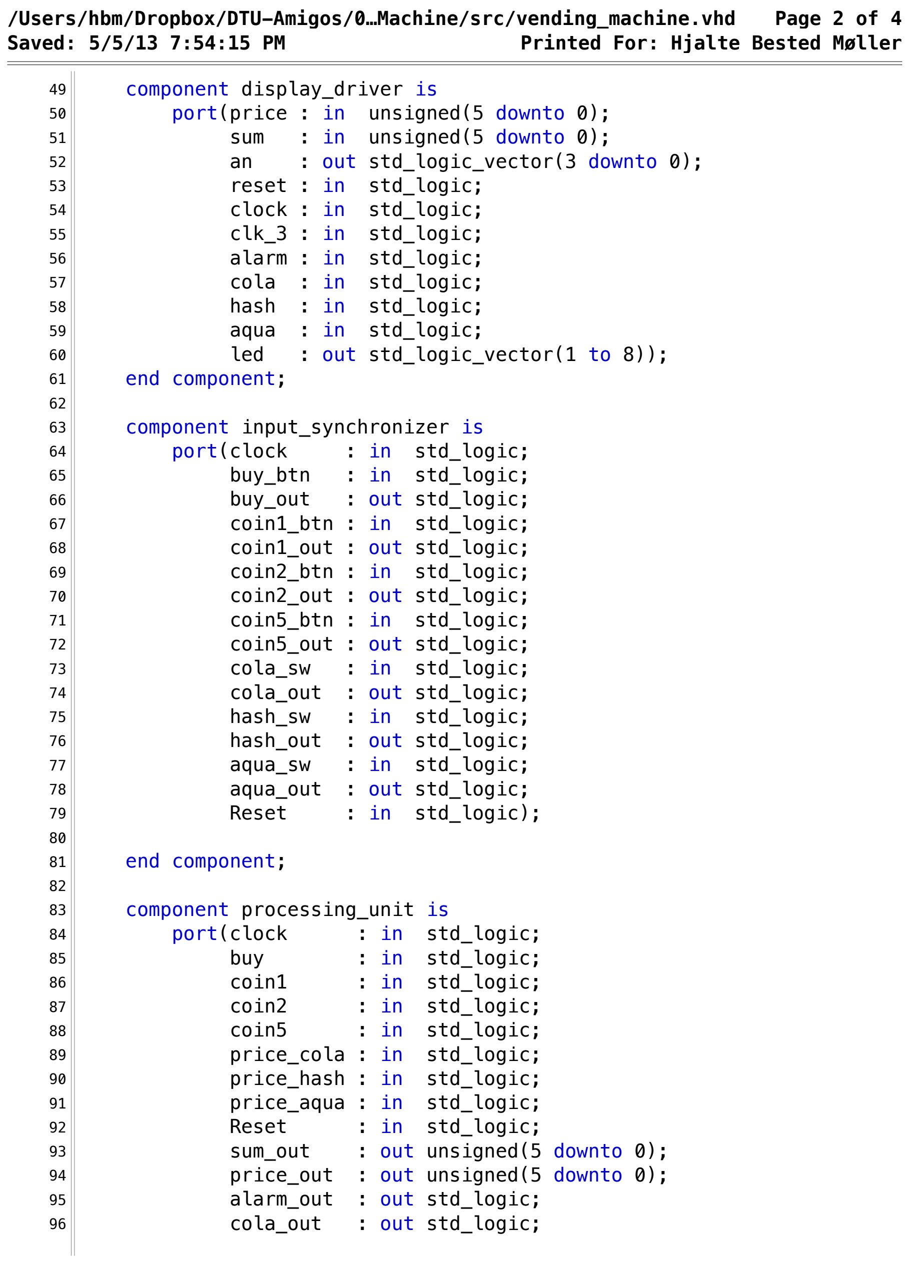
Figure 15: Page 3



/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

```vhdl
------------------------------------------------------------------------
-- Top component of the vending machine for the course:
-- 02139 Digital electronics 2 at the Technical University of Denmark
--
-- This component declares and instantiates all the components of the
-- vending machine.
------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity vending_machine is
    port(clk_50    : in  std_logic;
         clk_man   : in  std_logic;
         sel_man   : in  std_logic;
         reset     : in  std_logic;
         coin1_btn : in  std_logic;
         coin2_btn : in  std_logic;
         coin5_btn : in  std_logic;
         buy_btn   : in  std_logic;
         cola_sw   : in  std_logic;
         hash_sw   : in  std_logic;
         aqua_sw   : in  std_logic;
         an        : out std_logic_vector(3 downto 0);
         led       : out std_logic_vector(1 to 8));
end vending_machine;

architecture struct of vending_machine is
    signal clk                                : std_logic;
    signal clk_3                              : std_logic;
    signal buy                                : std_logic;
    signal coin1, coin2, coin5                : std_logic;
    signal price_cola, price_hash, price_aqua : std_logic;
    signal alarm, cola, aqua, hash            : std_logic;
    signal sum, price                         : unsigned(5 downto 0);

    --------------------------
    -- Component declarations
    --------------------------
    component clock_manager is
        port(clk_50  : in  std_logic;
             clk_man : in  std_logic;
             sel_man : in  std_logic;
             clk     : out std_logic;
             clk_3   : out std_logic);
    end component;

```

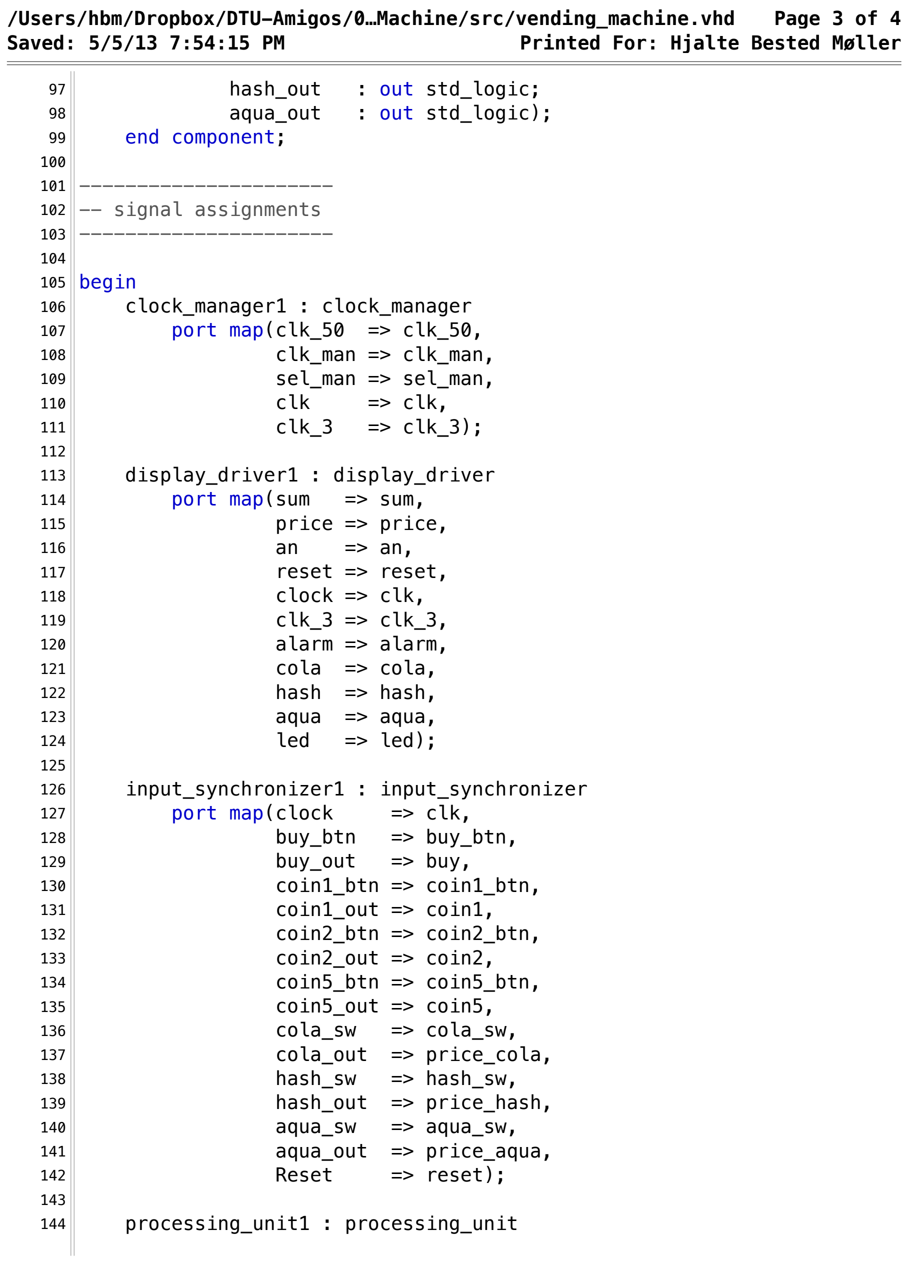
Figure 16: The complete vending machine, assembling all the components



/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

```vhdl
    component display_driver is
        port(price : in  unsigned(5 downto 0);
             sum   : in  unsigned(5 downto 0);
             an    : out std_logic_vector(3 downto 0);
             reset : in  std_logic;
             clock : in  std_logic;
             clk_3 : in  std_logic;
             alarm : in  std_logic;
             cola  : in  std_logic;
             hash  : in  std_logic;
             aqua  : in  std_logic;
             led   : out std_logic_vector(1 to 8));
    end component;

    component input_synchronizer is
        port(clock     : in  std_logic;
             buy_btn   : in  std_logic;
             buy_out   : out std_logic;
             coin1_btn : in  std_logic;
             coin1_out : out std_logic;
             coin2_btn : in  std_logic;
             coin2_out : out std_logic;
             coin5_btn : in  std_logic;
             coin5_out : out std_logic;
             cola_sw   : in  std_logic;
             cola_out  : out std_logic;
             hash_sw   : in  std_logic;
             hash_out  : out std_logic;
             aqua_sw   : in  std_logic;
             aqua_out  : out std_logic;
             Reset     : in  std_logic);

    end component;

    component processing_unit is
        port(clock      : in  std_logic;
             buy        : in  std_logic;
             coin1      : in  std_logic;
             coin2      : in  std_logic;
             coin5      : in  std_logic;
             price_cola : in  std_logic;
             price_hash : in  std_logic;
             price_aqua : in  std_logic;
             Reset      : in  std_logic;
             sum_out    : out unsigned(5 downto 0);
             price_out  : out unsigned(5 downto 0);
             alarm_out  : out std_logic;
             cola_out   : out std_logic;
```

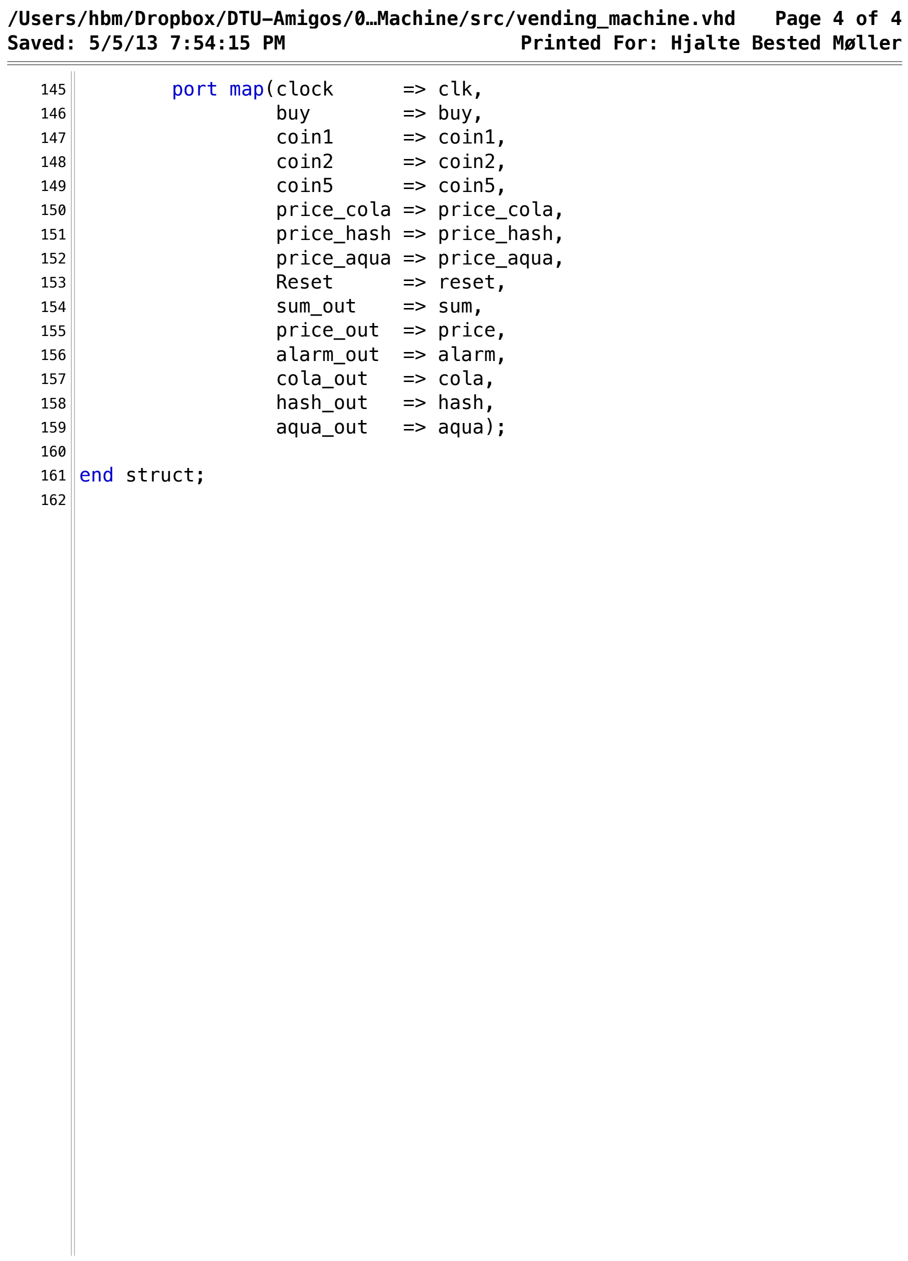
Figure 17: Page 2



/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 Besteded Møller

```
            hash_out   : out std_logic;
            aqua_out   : out std_logic);
    end component;

----------------------
-- signal assignments
----------------------

begin
    clock_manager1 : clock_manager
        port map(clk_50  => clk_50,
                 clk_man => clk_man,
                 sel_man => sel_man,
                 clk     => clk,
                 clk_3   => clk_3);

    display_driver1 : display_driver
        port map(sum   => sum,
                 price => price,
                 an    => an,
                 reset => reset,
                 clock => clk,
                 clk_3 => clk_3,
                 alarm => alarm,
                 cola  => cola,
                 hash  => hash,
                 aqua  => aqua,
                 led   => led);

    input_synchronizer1 : input_synchronizer
        port map(clock     => clk,
                 buy_btn   => buy_btn,
                 buy_out   => buy,
                 coin1_btn => coin1_btn,
                 coin1_out => coin1,
                 coin2_btn => coin2_btn,
                 coin2_out => coin2,
                 coin5_btn => coin5_btn,
                 coin5_out => coin5,
                 cola_sw   => cola_sw,
                 cola_out  => price_cola,
                 hash_sw   => hash_sw,
                 hash_out  => price_hash,
                 aqua_sw   => aqua_sw,
                 aqua_out  => price_aqua,
                 Reset     => reset);

    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

```
        port map(clock      => clk,
                 buy        => buy,
                 coin1      => coin1,
                 coin2      => coin2,
                 coin5      => coin5,
                 price_cola => price_cola,
                 price_hash => price_hash,
                 price_aqua => price_aqua,
                 Reset      => reset,
                 sum_out    => sum,
                 price_out  => price,
                 alarm_out  => alarm,
                 cola_out   => cola,
                 hash_out   => hash,
                 aqua_out   => aqua);

end struct;

```

Figure 19: Page 4