**Written Project 2-Washing Machine**

**ECEN 424**

**Dr.Doss**

**By: Freddie Boadu**

**Table of Content**

Introduction……………………………………………………………………………………………………………………………………………3

Block diagram……………………………………………………………………………………………….…………………………………………3

Component description ………………………………………………………………………………………………………………………3-4

Explanation of block diagram changes………………………………………….………………………………………………………4-5

State diagram……………………………………………………………………………………………………………………………………….6

Pitfall and achievements…………………………………………………………………………………………………………………………6

Vhdl code and usf file…………………………………………………………………………………………………………………………7-23

Reference and conclusion………………………………………………………………………………………………………………………24

**Introduction**

This report is pertaining to the second project done in ECEN 424. This project was based around simulating a washing machine. The washing machine is to wash, soak, rinse, and spin. This is the process if the washing is doing a normal washing. If the ultra-washing is taking place, the device needs to be washing for an extra rinse and soak cycle before the washing is concerned done. The washing machine also has a lid. When the lid is closed the washing should stop washing and countdown until there is an error. The washing is also supposed to take in a coin input. The coin input is supposed to a start the machine in the first place as well.

**Block Diagram**

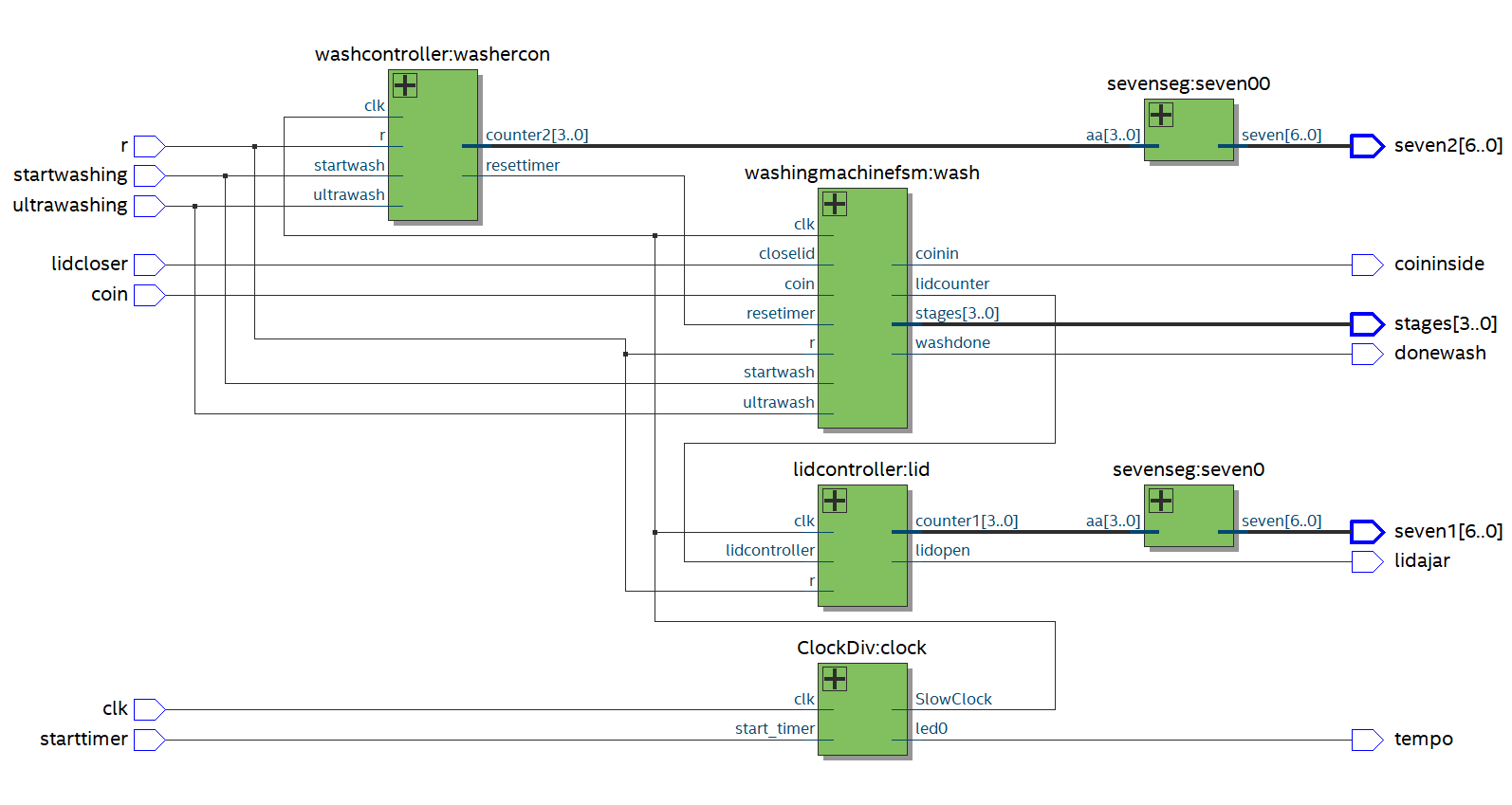
****

Figure 1.1 Block Diagram of the washing machine design

**Component Description**

Clock divider: The clock divider has a clock signal, a user timer, led and a out signal to distribute a slow, medium or fast signal. The clock signal is tied to the board. The user timer will start the clock divider when called upon. This is connected to a switch for the design. The Led is used to time the clock and see the frequency at which the clock signal is creating an event.

Central Fsm or the washing machine fsm: The finite state machine is the brains of the whole design. The finite state machine is taking in button presses that the user is doing externally on the board. This takes in the buttons to know what washing mode the machine should be on, if a coin is in the device and if the lid has been closed or not. This is the brains of the operation and makes sure it receieves signals to do its job right and effectively.

Seven coder: This is the seven segments of the device. When it gets a 3-bit code from the finite state machine to tell the coder what timer the lid is on and the countdown of the washer to what washing cycle the machine will go next into. This will display all this information on the two seven segment displays. There are two seven segments in my code that handle each function

Lidcontroller: This component keeps track of the lid. If the lid is closed the timer will stop. If the lid is open it will trigger this device and count down until 10 to which it will display an error. The controller is connected to the fsm to make sure those signals reach their point of interest.

Washcontroller: This component controls the washer machine count down. And when it gets a signal to resent the timer it will do that.

**Explanation of block diagram changes**

My block diagram changed with me including another counter. For the most part my design is pretty much the same.

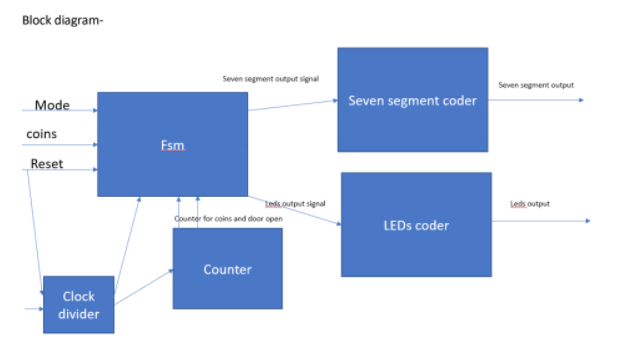


Figure 2- The first block diagram of the washer machine design

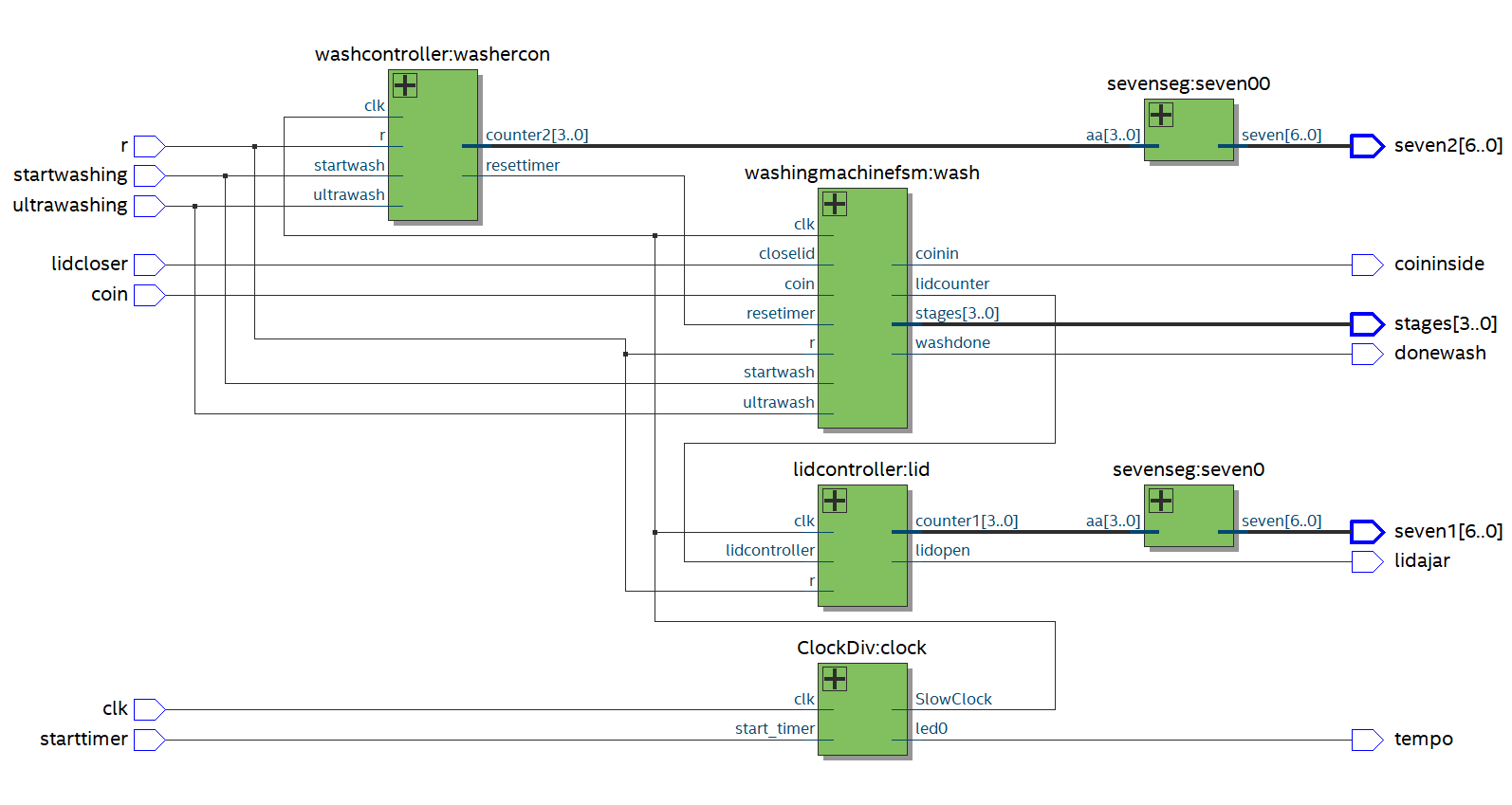
****

Figure 3- A repeat of figure 1

**State Diagram**

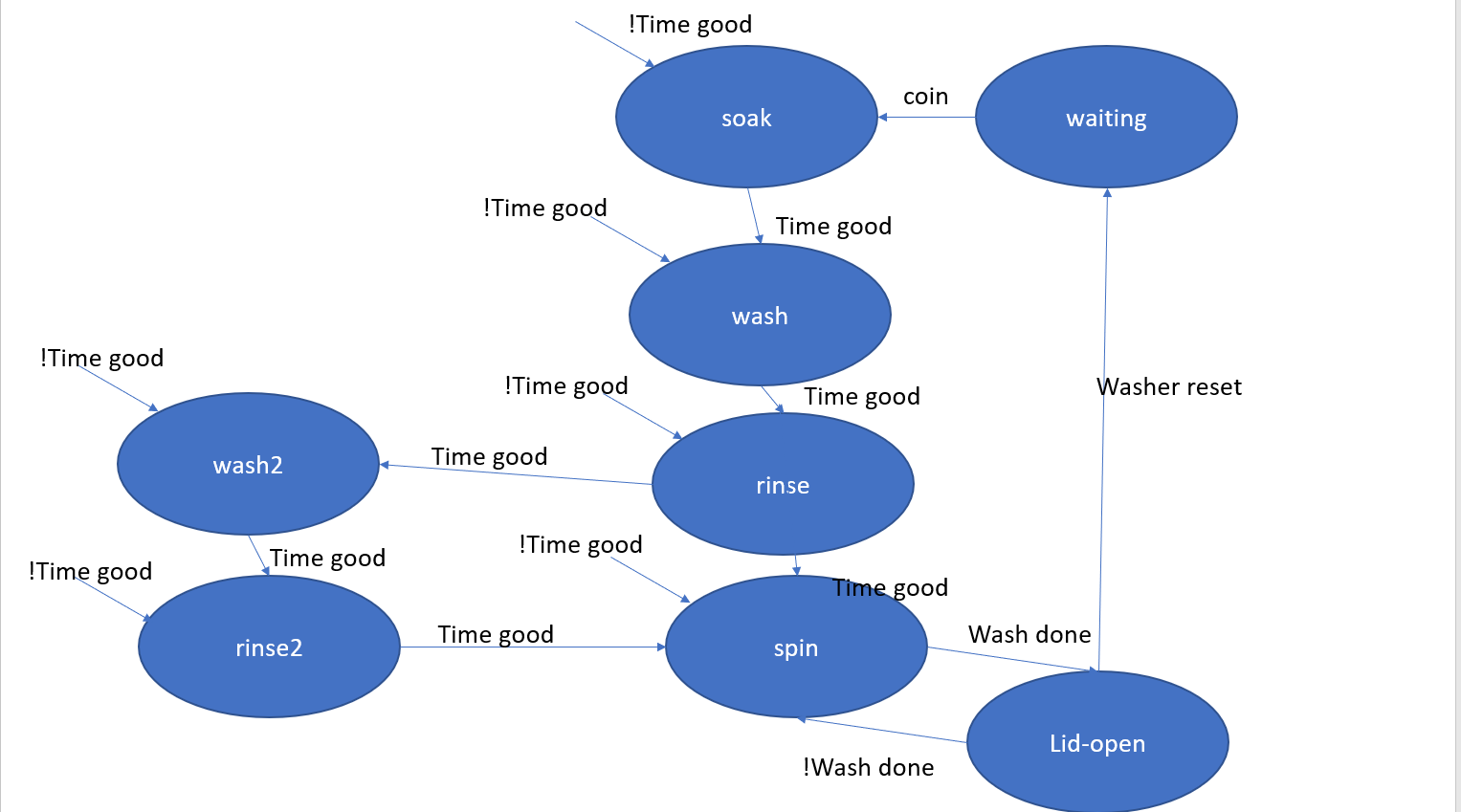


Figure 4- state diagram of the whole washing machine system

The state machine will start at a waiting state. It will then go through the washing states as long is there is no timer reset. If there is timer reset from opening the lid, then should reset that timer. The state machine also accounts for if their ultra-wash becomes a factor in the washing as well.

**Pitfalls and achievements**

During the design of this system, the idea of top down design was a huge achievement. The importance of learning and planning out details before fully implementing them is so important. It is an important component of systems engineering and exactly what is needed in the work world. Learning how to design the system according to the requirements before coding was extremely helpful. There was a little bit of a failure with timing. The lid does what it supposes to, but the washing machine part doesn’t count down and switch states like intended. It is a work in progress.

**VHDL Code**

**Top level**

--top level

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity toplevel is

port (clk,r: in std\_logic;

coin: in std\_logic;

starttimer: in std\_logic;

startwashing: in std\_logic;

ultrawashing: in std\_logic;

lidcloser: in std\_logic;

tempo: out std\_logic;

coininside: out std\_logic;

donewash: out std\_logic;

lidajar: out std\_logic;

stages: out std\_logic\_vector(3 downto 0);

seven1: out std\_logic\_vector(6 downto 0);

seven2: out std\_logic\_vector(6 downto 0));

end;

architecture beh of toplevel is

signal clockfeeder:std\_logic;

signal lidfeeder: std\_logic;

signal sevenfeeder: std\_logic\_vector(3 downto 0);

signal sevenfeeder2: std\_logic\_vector(3 downto 0);

component clockdiv is

port(clk: in STD\_logic;

start\_timer: in STD\_logic;

FastClock , MediumClock , SlowClock , led0 : out STD\_LOGIC );

end component ;

component washingmachinefsm is

port(clk,r: in std\_logic;

coin: in std\_logic;

startwash: in std\_logic;

ultrawash: in std\_logic;

closelid: in std\_logic;

coinin: out std\_logic;

stages: out std\_logic\_vector(3 downto 0);

washdone: out std\_logic;

washcounter: out std\_logic\_vector( 3 downto 0);

lidcounter: out std\_logic);

end component;

component lidcontroller is

port(clk,r, lidcontroller: in std\_logic;

lidopen: out std\_logic;

counter1: out std\_logic\_vector(3 downto 0));

end component;

component sevenseg is

Port ( aa : in STD\_LOGIC\_VECTOR(3 downto 0);

seven : out STD\_LOGIC\_VECTOR(6 downto 0));

end component;

begin

clock: clockdiv port map(clk=>clk, start\_timer=> starttimer, slowclock=>clockfeeder,led0=>tempo);

wash: washingmachinefsm port map(clk=>clockfeeder, r=>r, coin=>coin,startwash=>startwashing,ultrawash=>ultrawashing,closelid=>lidcloser,coinin=>coininside,stages=>stages,washcounter=>sevenfeeder2,washdone=>donewash,lidcounter=>lidfeeder);

lid : lidcontroller port map(clk=>clockfeeder, r=>r, lidcontroller=>lidfeeder,lidopen=>lidajar, counter1=>sevenfeeder);

seven0: sevenseg port map(aa=>sevenfeeder,seven=>seven1);

seven00: sevenseg port map(aa=>sevenfeeder2,seven=>seven2);

end;

**Door timer**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Doortimer is

port (clk,timego: in std\_logic;--clk from the divider and the signal for the timer to start for the door

counter: inout std\_logic\_vector(1 downto 0));--counter that is used to determine the counter getting to 3

end;

architecture beh of doortimer is

begin

process(clk,timego)

begin

if (clk' event and clk='1') then

if (timego='1') then

counter<=counter +1;--count to 3

elsif(timego='0') then

counter<="00";

end if;

end if;

end process;

end;

**Central fsm**

--central fsm for the elevator

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity washingmachinefsm is

port(clk,r: in std\_logic;

coin: in std\_logic;

startwash: in std\_logic;

ultrawash: in std\_logic;

closelid: in std\_logic;

coinin: out std\_logic;

stages: out std\_logic\_vector(3 downto 0);

washdone: out std\_logic;

washcounter: out std\_logic\_vector( 3 downto 0);

lidcounter: out std\_logic);

end;

architecture beh of washingmachinefsm is

type state is ( done,s0,soak,soak1,soak2,soak3, wash,wash1,wash2,wash3, rinse,rinse1,rinse2,rinse3, spin,spin1,spin2,spin3,soaker,soaker1,soaker2,soaker3, rinser,rinser1,rinser2,rinser3);

signal cs, ns: state;

signal stagestmp: std\_logic\_vector(3 downto 0);

signal washcountertmp: std\_logic\_vector( 3 downto 0);

begin

process(clk,r)

begin

if( r='1') then

cs<=s0;

elsif(clk'event and clk='1') then

cs<=ns;

end if;

end process;

process(coin,cs,startwash, ultrawash,closelid)

begin

stagestmp<="0000";

washcountertmp<="0000";

coinin<='0';

lidcounter<='0';

washdone<='0';

case cs is

when done=>

if(coin='1') then

ns<=s0;

elsif (coin='0') then

ns<=cs;

end if;

stagestmp<="0000";

washcountertmp<="0000";

when s0=>

if (closelid='0') then

lidcounter<='0';

else

lidcounter<='1';

end if;

if ( startwash='1' and closelid='1' ) then

ns<=soak;

coinin<='1';

elsif( ultrawash='1' and closelid='1' ) then

ns<=soak;

coinin<='1';

end if;

stagestmp<="0000";

washcountertmp<="0000";

when soak=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=soak1;

coinin<='1';

elsif( ultrawash='1' and closelid='1') then

ns<=soak1;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0100";

when soak1=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=soak2;

coinin<='1';

elsif( ultrawash='1' and closelid='1') then

ns<=soak2;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0011";

when soak2=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=soak3;

coinin<='1';

elsif( ultrawash='1' and closelid='1') then

ns<=soak3;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0010";

when soak3=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=wash;

coinin<='1';

elsif( ultrawash='1' and closelid='1') then

ns<=wash;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0000";

when wash=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=wash1;

coinin<='1';

elsif( ultrawash='1' and closelid='1' ) then

ns<=wash1;

coinin<='1';

end if;

stagestmp<="0010";

washcountertmp<="0100";

when wash1=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=wash2;

coinin<='1';

elsif( ultrawash='1' and closelid='1' ) then

ns<=wash2;

coinin<='1';

end if;

stagestmp<="0010";

washcountertmp<="0011";

when wash2=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=wash1;

coinin<='1';

elsif( ultrawash='1' and closelid='1') then

ns<=wash1;

coinin<='1';

end if;

stagestmp<="0010";

washcountertmp<="0010";

when wash3=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=rinse;

coinin<='1';

elsif( ultrawash='1' and closelid='1' ) then

ns<=wash1;

coinin<='1';

end if;

stagestmp<="0010";

washcountertmp<="0001";

when rinse=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(startwash='1' and closelid='1' ) then

ns<=rinse1;

coinin<='1';

elsif(ultrawash='1' and closelid='1' ) then

ns<=rinse1;

coinin<='1';

end if;

stagestmp<="0100";

washcountertmp<="0100";

when rinse1=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(startwash='1' and closelid='1' ) then

ns<=rinse2;

coinin<='1';

elsif(ultrawash='1' and closelid='1' ) then

ns<=rinse2;

coinin<='1';

end if;

stagestmp<="0100";

washcountertmp<="0011";

when rinse2=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(startwash='1' and closelid='1' ) then

ns<=rinse3;

coinin<='1';

elsif(ultrawash='1' and closelid='1' ) then

ns<=rinse3;

coinin<='1';

end if;

stagestmp<="0100";

washcountertmp<="0010";

when rinse3=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(startwash='1' and closelid='1' ) then

ns<=spin;

coinin<='1';

elsif(ultrawash='1' and closelid='1' ) then

ns<=spin;

coinin<='1';

end if;

stagestmp<="0100";

washcountertmp<="0001";

when spin=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=spin1;

washdone<='1';

coinin<='0';

elsif( ultrawash='1' and closelid='1' ) then

ns<=spin1;

coinin<='1';

end if;

stagestmp<="1000";

washcountertmp<="0100";

when spin1=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=spin2;

washdone<='1';

coinin<='0';

elsif( ultrawash='1' and closelid='1' ) then

ns<=spin2;

coinin<='1';

end if;

stagestmp<="1000";

washcountertmp<="0011";

when spin2=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=spin3;

washdone<='1';

coinin<='0';

elsif( ultrawash='1' and closelid='1' ) then

ns<=spin3;

coinin<='1';

end if;

stagestmp<="1000";

washcountertmp<="0010";

when spin3=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( startwash='1' and closelid='1' ) then

ns<=done;

washdone<='1';

coinin<='0';

elsif( ultrawash='1' and closelid='1' ) then

ns<=soaker;

coinin<='1';

end if;

stagestmp<="1000";

washcountertmp<="0001";

when soaker=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( ultrawash='1' and closelid='1') then

ns<=soaker1;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0100";

when soaker1=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( ultrawash='1' and closelid='1' ) then

ns<=soaker2;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0011";

when soaker2=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( ultrawash='1' and closelid='1' ) then

ns<=soaker3;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0010";

when soaker3=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if( ultrawash='1' and closelid='1' ) then

ns<=rinser;

coinin<='1';

end if;

stagestmp<="0001";

washcountertmp<="0001";

when rinser=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(ultrawash='1' and closelid='1' ) then

ns<=rinser1;

washdone<='1';

coinin<='0';

end if;

stagestmp<="0100";

washcountertmp<="0100";

when rinser1=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(ultrawash='1' and closelid='1' ) then

ns<=rinser2;

washdone<='1';

coinin<='0';

end if;

stagestmp<="0100";

washcountertmp<="0011";

when rinser2=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(ultrawash='1' and closelid='1' ) then

ns<=rinser3;

washdone<='1';

coinin<='0';

end if;

stagestmp<="0100";

washcountertmp<="0010";

when rinser3=>

if (closelid='0') then

lidcounter<='1';

else

lidcounter<='0';

end if;

if(ultrawash='1' and closelid='1' ) then

ns<=done;

washdone<='1';

coinin<='0';

end if;

stagestmp<="0100";

washcountertmp<="0001";

when others=>

null;

end case;

end process;

washcounter<=washcountertmp;

stages<=stagestmp;

end;

**Lidcontroller**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity lidcontroller is

port (clk,r, lidcontroller: in std\_logic;

lidopen: out std\_logic;--These are all signals that the fsm will send and start the counters

counter1: out std\_logic\_vector(3 downto 0));--this counter will go back to fsm

end;

architecture beh of lidcontroller is

type state is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10);

signal cs,ns:state;

signal lidopentmp:std\_logic;

begin

process(clk,r)

begin

if(r='1') then

cs<=s0;

elsif (clk' event and clk='1') then

cs<=ns;

end if;

end process;

process(cs,lidcontroller)

begin

case cs is

when s0=>

if (lidcontroller='0') then

counter1<="0000";

lidopentmp<='1';

ns<=s1;

else

ns<=s0;

counter1<="0000";

end if;

when s1=>

if (lidcontroller='0') then

counter1<="0001";

lidopentmp<='1';

ns<=s2;

else

ns<=s0;

counter1<="0000";

end if;

when s2=>

if (lidcontroller='0') then

counter1<="0010";

lidopentmp<='1';

ns<=s3;

else

ns<=s0;

counter1<="0000";

end if;

when s3=>

if (lidcontroller='0') then

counter1<="0011";

lidopentmp<='1';

ns<=s4;

else

ns<=s0;

counter1<="0000";

end if;

when s4=>

if (lidcontroller='0') then

counter1<="0100";

lidopentmp<='1';

ns<=s5;

else

ns<=s0;

counter1<="0000";

end if;

when s5=>

if (lidcontroller='0') then

counter1<="0101";

lidopentmp<='1';

ns<=s6;

else

ns<=s0;

counter1<="0000";

end if;

when s6=>

if (lidcontroller='0') then

counter1<="0110";

lidopentmp<='1';

ns<=s7;

else

ns<=s0;

counter1<="0000";

end if;

when s7=>

if (lidcontroller='0') then

counter1<="0111";

lidopentmp<='1';

ns<=s8;

else

ns<=s0;

counter1<="0000";

end if;

when s8=>

if (lidcontroller='0') then

counter1<="1000";

lidopentmp<='1';

ns<=s9;

else

ns<=s0;

counter1<="0000";

end if;

when s9=>

if (lidcontroller='0') then

counter1<="1001";

lidopentmp<='1';

ns<=s10;

else

ns<=s0;

counter1<="0000";

end if;

when s10=>

if (lidcontroller='0') then

counter1<="1111";

lidopentmp<='1';

ns<=s10;

else

ns<=s0;

counter1<="0000";

end if;

when others=>

counter1<="1111";

end case;

end process;

lidopen<=lidopentmp;

end;

**Wash controller**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity washcontroller is

port(clk,r: in std\_logic;

startwash: in std\_logic;

ultrawash: in std\_logic;

resettimer: out std\_logic;

counter2: out std\_logic\_vector(3 downto 0));

end;

architecture beh of washcontroller is

type state is (s0,s1,s2,s3,s4);

signal cs,ns:state;

begin

process(clk,r)

begin

if(r='1') then

cs<=s0;

elsif (clk'event and clk='1') then

cs<=ns;

end if;

end process;

process(cs,startwash,ultrawash)

begin

case cs is

when s0=>

if (startwash='1') then

counter2<="0100";

ns<=s1;

elsif (ultrawash='1') then

counter2<="0100";

ns<=s1;

else

ns<=s0;

counter2<="0000";

end if;

when s1=>

if (startwash='1') then

counter2<="0011";

ns<=s2;

elsif (ultrawash='1') then

counter2<="0011";

ns<=s2;

else

ns<=s0;

counter2<="0000";

end if;

when s2=>

if (startwash='1') then

counter2<="0010";

ns<=s3;

elsif (ultrawash='1') then

counter2<="0010";

ns<=s3;

else

ns<=s0;

counter2<="0000";

end if;

when s3=>

if (startwash='1') then

counter2<="0001";

ns<=s4;

elsif (ultrawash='1') then

counter2<="0001";

ns<=s4;

else

ns<=s0;

counter2<="0000";

end if;

when s4=>

if (startwash='1') then

counter2<="0000";

ns<=s0;

resettimer<='1';

elsif (ultrawash='1') then

counter2<="0000";

ns<=s0;

resettimer<='1';

else

ns<=s0;

counter2<="0000";

end if;

when others=>

null;

end case;

end process;

end;

**Clock divider**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity ClockDiv is

port(clk: in STD\_logic;

start\_timer: in STD\_logic;

FastClock , MediumClock , SlowClock , led0 : out STD\_LOGIC );

end Clockdiv ;

architecture beh of Clockdiv is

signal slowClock\_sig : STD\_LOGIC;

begin

process

variable cnt : STD\_LOGIC\_VECTOR (26 downto 0):= "000000000000000000000000000";

begin

wait until (( clk 'EVENT) AND ( clk = '1'));

if ( start\_timer = '0') then

cnt := "000000000000000000000000000" ;

else

cnt := STD\_LOGIC\_VECTOR( unsigned ( cnt ) + 1);

end if ;

FastClock <= cnt (22);

MediumClock <= cnt (24);

SlowClock <= cnt (26);

slowClock\_sig <= cnt (26);

if ( slowClock\_sig = '1') then

led0 <= '1';

else

led0 <= '0';

end if ;

end process;

end beh;

**Seven decoder**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity sevenseg is

Port ( aa : in STD\_LOGIC\_VECTOR(3 downto 0);

seven : out STD\_LOGIC\_VECTOR(6 downto 0));

end sevenseg;

architecture Behavioral of sevenseg is

begin

process(aa)

begin

Case aa is

when "0000"=> seven <="1000000"; -- '0'

when "0001"=> seven <="1111001"; -- '1'

when "0010"=> seven <="0100100"; -- '2'

when "0011"=> seven <="0110000"; -- '3'

when "0100"=> seven <="0011001"; -- '4'

when "0101"=> seven <="0010010"; -- '5'

when "0110"=> seven <="0000010"; -- '6'

when "0111"=> seven <="1111000"; -- '7'

when "1000"=> seven <="0000000"; -- '8'

when "1001"=> seven <="0011000"; -- '9'

when "1111"=>seven<="1111001"; -- 'E'

when others=> seven <= "1111001";-- everything else would be 0

end case;

end process;

end Behavioral;

**UCF**

# -------------------------------------------------------------------------- #

#

# Copyright (C) 2020 Intel Corporation. All rights reserved.

# Your use of Intel Corporation's design tools, logic functions

# and other software and tools, and any partner logic

# functions, and any output files from any of the foregoing

# (including device programming or simulation files), and any

# associated documentation or information are expressly subject

# to the terms and conditions of the Intel Program License

# Subscription Agreement, the Intel Quartus Prime License Agreement,

# the Intel FPGA IP License Agreement, or other applicable license

# agreement, including, without limitation, that your use is for

# the sole purpose of programming logic devices manufactured by

# Intel and sold by Intel or its authorized distributors. Please

# refer to the applicable agreement for further details, at

# https://fpgasoftware.intel.com/eula.

#

# -------------------------------------------------------------------------- #

#

# Quartus Prime

# Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition

# Date created = 15:30:35 November 23, 2020

#

# -------------------------------------------------------------------------- #

#

# Notes:

#

# 1) The default values for assignments are stored in the file:

# washing\_assignment\_defaults.qdf

# If this file doesn't exist, see file:

# assignment\_defaults.qdf

#

# 2) Altera recommends that you do not modify this file. This

# file is updated automatically by the Quartus Prime software

# and any changes you make may be lost or overwritten.

#

# -------------------------------------------------------------------------- #

set\_global\_assignment -name FAMILY "MAX 10"

set\_global\_assignment -name DEVICE 10M50DAF484C7G

set\_global\_assignment -name TOP\_LEVEL\_ENTITY toplevel

set\_global\_assignment -name ORIGINAL\_QUARTUS\_VERSION 20.1.1

set\_global\_assignment -name PROJECT\_CREATION\_TIME\_DATE "15:30:35 NOVEMBER 23, 2020"

set\_global\_assignment -name LAST\_QUARTUS\_VERSION "20.1.1 Lite Edition"

set\_global\_assignment -name VHDL\_FILE ../../Downloads/timingcontroller.vhd

set\_global\_assignment -name VHDL\_FILE "../../Downloads/washing-machine.vhd"

set\_global\_assignment -name VHDL\_FILE ../../Downloads/clockdiv.vhd

set\_global\_assignment -name PROJECT\_OUTPUT\_DIRECTORY output\_files

set\_global\_assignment -name MIN\_CORE\_JUNCTION\_TEMP 0

set\_global\_assignment -name MAX\_CORE\_JUNCTION\_TEMP 85

set\_global\_assignment -name DEVICE\_FILTER\_PIN\_COUNT 484

set\_global\_assignment -name DEVICE\_FILTER\_SPEED\_GRADE 7

set\_global\_assignment -name ERROR\_CHECK\_FREQUENCY\_DIVISOR 256

set\_global\_assignment -name VHDL\_FILE washingcontroller.vhd

set\_global\_assignment -name VHDL\_FILE sevensegdecoder.vhd

set\_global\_assignment -name VHDL\_FILE toplevel.vhd

set\_global\_assignment -name PARTITION\_NETLIST\_TYPE SOURCE -section\_id Top

set\_global\_assignment -name PARTITION\_FITTER\_PRESERVATION\_LEVEL PLACEMENT\_AND\_ROUTING -section\_id Top

set\_global\_assignment -name PARTITION\_COLOR 16764057 -section\_id Top

set\_location\_assignment PIN\_P11 -to clk

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to clk

set\_location\_assignment PIN\_C10 -to coin

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to coin

set\_location\_assignment PIN\_A8 -to coininside

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to coininside

set\_location\_assignment PIN\_A9 -to donewash

set\_location\_assignment PIN\_A10 -to lidajar

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to donewash

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to lidajar

set\_location\_assignment PIN\_C11 -to lidcloser

set\_location\_assignment PIN\_B14 -to r

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to lidcloser

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to r

set\_location\_assignment PIN\_F15 -to starttimer

set\_location\_assignment PIN\_A14 -to startwashing

set\_location\_assignment PIN\_A13 -to ultrawashing

set\_location\_assignment PIN\_B11 -to tempo

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to ultrawashing

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to tempo

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to startwashing

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to starttimer

set\_location\_assignment PIN\_C14 -to seven1[0]

set\_location\_assignment PIN\_E15 -to seven1[1]

set\_location\_assignment PIN\_C15 -to seven1[2]

set\_location\_assignment PIN\_C16 -to seven1[3]

set\_location\_assignment PIN\_E16 -to seven1[4]

set\_location\_assignment PIN\_D17 -to seven1[5]

set\_location\_assignment PIN\_C17 -to seven1[6]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1[6]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1[5]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1[4]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1[3]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1[2]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1[1]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1[0]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven1

set\_location\_assignment PIN\_C18 -to seven2[0]

set\_location\_assignment PIN\_D18 -to seven2[1]

set\_location\_assignment PIN\_E18 -to seven2[2]

set\_location\_assignment PIN\_B16 -to seven2[3]

set\_location\_assignment PIN\_A17 -to seven2[4]

set\_location\_assignment PIN\_A18 -to seven2[5]

set\_location\_assignment PIN\_B17 -to seven2[6]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2[6]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2[5]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2[4]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2[3]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2[2]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2[1]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2[0]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to seven2

set\_location\_assignment PIN\_A11 -to stages[0]

set\_location\_assignment PIN\_D14 -to stages[1]

set\_location\_assignment PIN\_E14 -to stages[2]

set\_location\_assignment PIN\_C13 -to stages[3]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to stages[0]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to stages[1]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to stages[2]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to stages[3]

set\_instance\_assignment -name IO\_STANDARD "3.3-V LVTTL" -to stages

set\_location\_assignment PIN\_B12 -to washeron

set\_instance\_assignment -name PARTITION\_HIERARCHY root\_partition -to | -section\_id Top

**References**

1. The Deo Lab manual within blackboard
2. Class notes from ECEN 424 on fsm, timers, and top down design

**Conclusion**

The project involving the washing machine was a failure this time around. A lot of it came from trying to do more than I needed to and not just getting down the basics. The pitfalls and achieved helped implement growth as a student and a FPGA programmer. The elevator helps to teach the important of top down design, finite state machines, and timers in VHDL.