EE3921 Digital System Design

Section 031, Fall 2020

Professor: Kerry R. Widder, Ph.D.

Electrical Engineering and Computer Science Department

Milwaukee School of Engineering

Laboratory 2: “Finite State Machine (FSM) logic design"

Jorge J. Jurado-Garcia

Date: 09/17/2020

Level Attempted: Advanced

Objective: The objective of this lab was review VHDL Programming by creating a finite state machine that creates bread, using the DE10-Lite board.

Description: In this lab I created the bread machine controller, with two bread inputs regular or dough only, start switch, and reset switch. The outputs for this machine will be Heaters 1 and 2, beater, beeper, and Finished signal. The process of the bread being taken depends on the type of bread. If the user selected dough the process should be as following. Mix=>Kneed=>Done. When regular is selected the process is Mix => Knead => Rise1 => Beat down => Rise2 => Bake => Done. Each process has certain process this can be described below.

1. Mix – heat is off and the beater will be turned on for 2 seconds, off for three seconds repeat two more times.
2. Knead – beater is on for 10 seconds
3. Rise1 – beater off, heat1 on for 10 seconds,
4. Beater done – heat is off, beater on for 2 seconds.
5. Rise2 – beater off, heat 1 on for 12 seconds,
6. Bake – heat1 and heat 2 on for 7 seconds,
7. Done – all outputs off, beeper on for 5 seconds, then light the Done LED. Repeat cycle until start is high again.

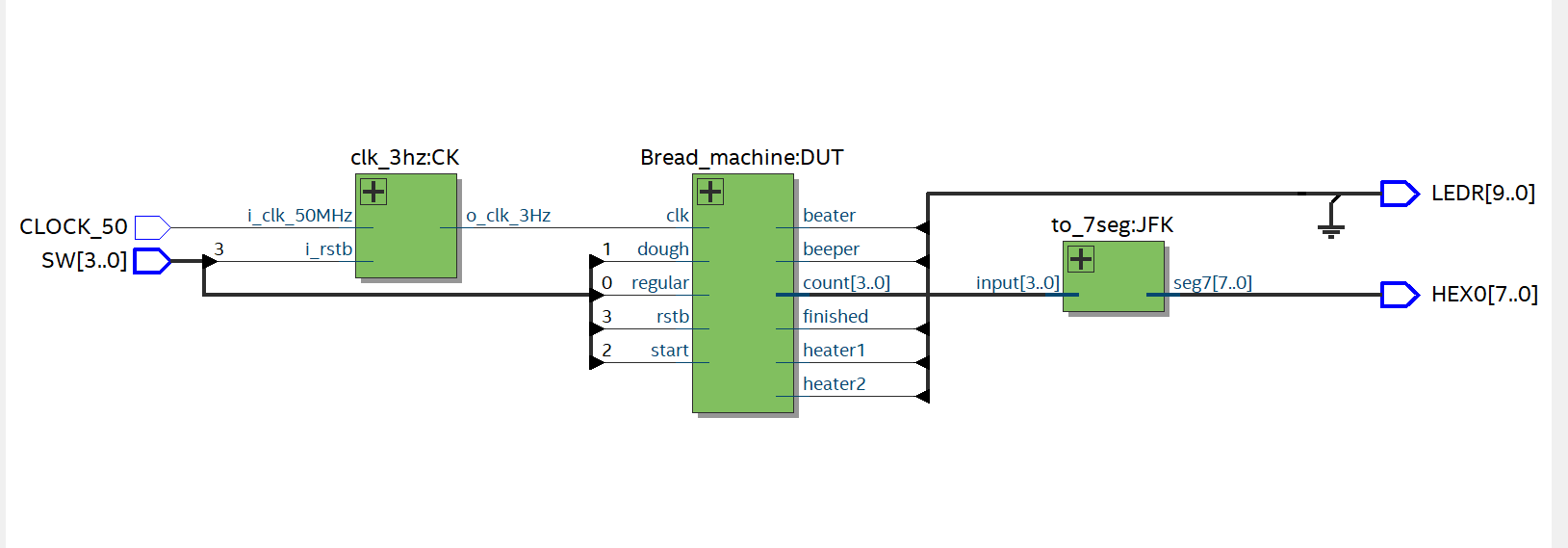
--INPUTS: BREAD TYPE: REGULAR(SW0) OR DOUGH(SW1)

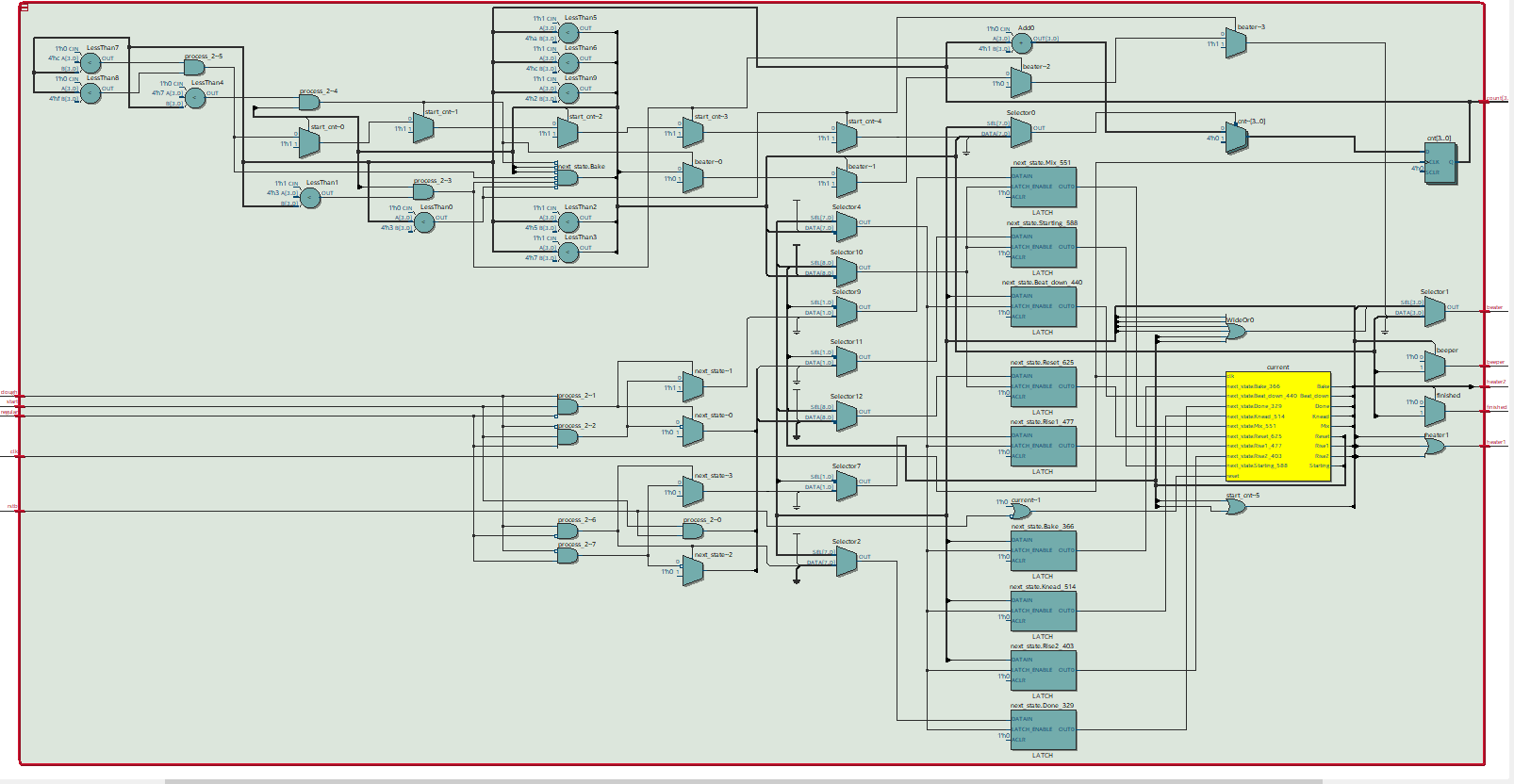
START(SW2), RESET(RSTB) (SW3)

--OUTPUTS: HEATER(1)[LEDR0], HEATER(2)[LEDR1], BEATER[LEDR2], BEEPER[LEDR3]. FINISHED[LEDR9]

Conclusion: Overall, this lab was a success, the toughest part I had with this lab would be the Mix state and implementing my FSM with my DE10 Lite. The solution to this problem was to hardcode the states for Mix and to follow the proper project setup when implementing my DE10 lite board. Overall, the lab was a success, and I learned a lot about keeping my process statements in separate processes for debugging.

RTL Viewer





VHDL CODE

BREAD\_MACHINE

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

--LAB 2 FINITE STATE MACHINE LOGIC DESIGN

--FOR BREAD MAKER

--DATE: 9/14/2021

--AUTHOR: JORGE JURADO-GARCIA

--REV 3

--included output logic of fsm 9/15/21

--created RESET state and fixed output logic in MIX state 9/16/21

--fixed project setup and redid logic for mix state 9/17/21

--DE10lite qsf implemented.

--

--INPUTS: BREAD TYPE: REGULAR OR DOUGH

-- START, RESET(RSTB)

--

--OUTPUTS: HEATER(1), HEATER(2), BEATER, BEEPER.

--PROCESS:

--1.Mix – HEATER[1] & HEATER [2] off, BEATER ON for (2) seconds, off (3) seconds (3x)

--2.Knead – BEATER ON for (10) seconds

--3.Rise1 – BEATER OFF, HEATER[1] for (10) seconds

--4.Beat down – HEATER[1] OFF, BEATER ON For (2) seconds

--5.Rise2 - BEATER OFF, HEATER[1] for (12) seconds

--6.Bake – HEATER[1] and HEATER[2] ON for (7) seconds

--7.Done – HEATER[1] AND HEATER[2] OFF, BEATER OFF, BEEPER ON for (5) seconds,

-- then light the Done LED until either a reset or the start of another cycle.

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**Entity** Bread\_machine **is**

**port(**

clk**,**start**,**rstb**,** dough**,**regular**:** **IN** std\_logic**;**

count**:** **OUT** std\_logic\_vector**(**3 **downto** 0**);**

heater1**,** heater2**,** beater**,** beeper**,** finished **:** **OUT** std\_logic

**);**

**end** **entity** Bread\_machine**;**

**Architecture** behavioral **of** Bread\_machine **is**

-----name of the states that fsm will be in

**TYPE** state\_type **is** **(**Reset**,** Starting**,** Mix**,** Knead**,** Rise1**,** Beat\_down**,** Rise2**,** Bake**,** Done**);**

---signals for names of transitions between states

**SIGNAL** current**,** next\_state **:** state\_type**;**

--singals for counter

**SIGNAL** cnt**:** unsigned**(**3 **downto** 0**);**

**SIGNAL** start\_cnt**:** std\_logic**;**

--constant values for timers and etc

**CONSTANT** time\_2**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**2**,**4**);**

**CONSTANT** time\_3**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**3**,**4**);**

**CONSTANT** time\_5**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**5**,**4**);**

**CONSTANT** time\_6**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**6**,**4**);**

**CONSTANT** time\_7**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**7**,**4**);**

**CONSTANT** time\_10**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**10**,**4**);**

**CONSTANT** time\_12**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**12**,**4**);**

**CONSTANT** time\_15**:** unsigned**(**3 **downto** 0**)** **:=** **to\_unsigned(**15**,**4**);**

**BEGIN**

count **<=** std\_logic\_vector**(**cnt**);**

--logic for rstb and rising clock edge

**PROCESS(**clk**,** rstb**)**

**BEGIN**

**IF(**rstb **=** '0'**)** **THEN**

current **<=** Reset**;**

**ELSIF(rising\_edge(**clk**))** **THEN**

current **<=** next\_state**;**

**END** **IF;**

**END** **PROCESS;**

--logic for timer and counter

counterr**:** **PROCESS(**clk**,** start\_cnt**)**

**BEGIN**

**IF(rising\_edge(**clk**))** **THEN**

**IF(**start\_cnt **=** '0'**)** **THEN**

cnt **<=** **(OTHERS** **=>** '0'**);**

**ELSE** cnt **<=** cnt **+** 1**;**

**END** **IF;**

**END** **IF;**

**END** **PROCESS** counterr**;**

--combinationals next state logic

**PROCESS(**current**,** cnt**,** rstb**,** start**)**

**BEGIN**

**CASE** current **is**

**WHEN** Reset **=>**

start\_cnt **<=** '1'**;** --keep timer going

beater **<=** '0'**;**

beeper **<=** '0'**;**

finished **<=** '0'**;**

**IF((**rstb **=** '1'**)** **and** **(**start **=** '1'**))** **THEN**

next\_state **<=** Starting**;**

start\_cnt **<=** '0'**;**

**ELSE**

next\_state **<=** Reset**;**

start\_cnt **<=** '0'**;**

**END** **IF;**

**WHEN** Starting **=>**

start\_cnt **<=** '1'**;**

beater **<=** '0'**;**

beeper **<=** '0'**;**

finished **<=** '0'**;**

**IF(** **(**dough **=** '1'**)** **and** **(**start **=** '1'**)** **and** **(**regular **=** '0'**)** **)** **THEN**

next\_state **<=** Mix**;**

start\_cnt **<=** '0'**;** --starting fresh

**ELSIF((**regular **=** '1'**)and(**start **=** '1'**)and(**dough **=** '0'**))** **THEN**

next\_state **<=** Mix**;**

start\_cnt **<=** '0'**;** --starting fresh

**ELSE**

next\_state **<=** Reset**;**

start\_cnt **<=** '0'**;** --starting fresh

**END** **IF;**

**WHEN** Mix **=>**

start\_cnt **<=** '1'**;** --keeps timer at one

finished **<=** '0'**;**

beater **<=** '0'**;**

beeper **<=** '0'**;**

**IF(**cnt **<** time\_3**)** **THEN**

beater **<=** '1'**;**

**ELSIF((**cnt **>=** time\_3**)** **and** **(**cnt **<=** time\_5**))** **THEN**

beater **<=** '0'**;**

**ELSIF(**cnt **<=** time\_7**)** **THEN**

beater **<=** '1'**;**

**ELSIF((**cnt **>** time\_7**)** **and** **(**cnt **<=** time\_10**))** **THEN**

beater **<=** '0'**;**

**ELSIF(**cnt **<=** time\_12**)** **THEN**

beater **<=** '1'**;**

**ELSIF((**cnt **>** time\_12**)** **and** **(**cnt **<** time\_15**))** **THEN**

beater **<=** '0'**;**

**ELSE**

beater **<=** '0'**;**

next\_state **<=** Knead**;**

start\_cnt **<=** '0'**;**

**END** **IF;**

**WHEN** Knead **=>**

start\_cnt **<=** '1'**;** --keeps timer at one

finished **<=** '0'**;**

beeper **<=** '0'**;**

beater **<=** '0'**;**

**IF(**cnt **<=** time\_10**)** **THEN**

beater **<=** '1'**;**

**ELSIF(** **(**dough **=** '1'**)** **and** **(**regular **=** '0'**)** **)** **THEN**

beater **<=** '0'**;**

next\_state **<=** Done**;**

start\_cnt **<=** '0'**;**

**ELSIF(** **(**dough **=** '0'**)** **and** **(**regular **=** '1'**)** **)** **THEN**

beater **<=** '0'**;**

next\_state **<=** Rise1**;**

start\_cnt **<=** '0'**;** --starting fresh

**ELSE**

beater **<=** '0'**;**

next\_state **<=** Reset**;**

start\_cnt **<=** '0'**;** --starting fresh

**END** **IF;**

**WHEN** Rise1 **=>**

start\_cnt **<=** '1'**;** --keeps timer at one

finished **<=** '0'**;**

beater **<=** '0'**;**

beeper **<=** '0'**;**

**IF(**cnt **<=** time\_10**)** **THEN**

-- heater1 <= '1';

**ELSE**

next\_state **<=** Beat\_down**;**

start\_cnt **<=** '0'**;** --starting fresh

**END** **IF;**

**WHEN** Beat\_down **=>**

start\_cnt **<=** '1'**;** --keeps timer at one

finished **<=** '0'**;**

beater **<=** '0'**;**

beeper **<=** '0'**;**

-- heater1 <= '0';

**IF(**cnt **<=** time\_2**)** **THEN**

beater **<=** '1'**;**

**ELSE**

beater **<=** '0'**;**

next\_state **<=** Rise2**;**

start\_cnt **<=** '0'**;** --starting fresh

**END** **IF;**

**WHEN** Rise2 **=>**

start\_cnt **<=** '1'**;** --keeps timer at one

beeper **<=** '0'**;**

finished **<=** '0'**;**

beater **<=** '0'**;**

**IF(**cnt **<=** time\_12**)** **THEN**

-- heater1 <= '1';

**ELSE**

next\_state **<=** Bake**;**

start\_cnt **<=** '0'**;** --starting fresh

**END** **IF;**

**WHEN** Bake **=>**

start\_cnt **<=** '1'**;** --keeps timer at one

finished **<=** '0'**;**

beeper **<=** '0'**;**

beater **<=** '0'**;**

**IF(**cnt **<=** time\_7**)** **THEN**

-- heater1 <= '1';

-- heater2 <= '1';

**ELSE**

next\_state **<=** Done**;**

start\_cnt **<=** '0'**;** --starting fre

**END** **IF;**

**WHEN** Done **=>**

start\_cnt **<=** '1'**;**

--heater1 <= '0';

--heater2 <= '0';

beater **<=** '0'**;**

finished **<=** '1'**;**

beeper **<=** '1'**;**

**IF(**cnt **<=** time\_5**)** **THEN**

finished **<=** '1'**;**

beeper **<=** '1'**;**

**ELSE**

finished **<=** '0'**;**

beeper **<=** '0'**;**

next\_state **<=** Reset**;**

start\_cnt **<=** '0'**;** --starting fresh

**END** **IF;**

**END** **CASE;**

**END** **PROCESS;**

--output logic

heater1 **<=** '1' **WHEN** **((**current **=** Rise1**)** **OR** **(**current **=** Rise2**)** **OR** **(**current **=** Bake**))** **ELSE** '0'**;**

heater2 **<=** '1' **WHEN** **(**current **=** Bake**)** **ELSE** '0'**;**

**END** behavioral**;**

TIMER\_N\_SEC.

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** clk\_3hz **is**

**port(**

i\_clk\_50MHz**:** **in** std\_logic**;**

i\_rstb**:** **in** std\_logic**;**

o\_clk\_3Hz**:** **out** std\_logic

**);**

**end** **entity;**

**architecture** behavioral **of** clk\_3hz **is**

--

--constants and parameters

--

**constant** CLOCKS\_PER\_HALF\_PERIOD**:** signed**(**24 **downto** 0**)** **:=** **to\_signed((((**50\_000\_000**/**2**)/**3**)-**2**),**25**);**

--internal signals

**signal** cnt**:** signed**(**24 **downto** 0**);**

**signal** clk\_sig**:** std\_logic**;**

**begin**

**process(**i\_clk\_50MHz**,** i\_rstb**)**

**begin**

**if(**i\_rstb **=** '0'**)** **then**

cnt **<=** CLOCKS\_PER\_HALF\_PERIOD**;**

clk\_sig **<=** '0'**;**

**elsif** **(rising\_edge(**i\_clk\_50MHz**))** **then**

cnt **<=** cnt **-** 1**;**

**if(**cnt**<** 0**)** **then**

cnt **<=** CLOCKS\_PER\_HALF\_PERIOD**;**

clk\_sig **<=** **not** clk\_sig**;**

**end** **if;**

**end** **if;**

**end** **process;**

o\_clk\_3Hz **<=** clk\_sig**;**

**end** behavioral**;**

TO\_7Seg

----------------------------------------------

--

-- to\_7seg.vhdl

--

--created 9/11/21

--jorge jurado-garcia

--

--rev 0

------------------------------

-- takes a for bit bit value and then translates into a seven segment display

--------------------------------------------

--Inputs: input

--Outputs: seg7

------------------------------------------------

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** to\_7seg **is**

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

seg7 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**)**

**);**

**end** to\_7seg**;**

**architecture** Behavioral **of** to\_7seg **is**

**begin**

--'input' corresponds to MSB of seg7 and 'g' corresponds to LSB of seg7.

-- when a bit is low the led on the seg7 will be on else '0'

**process** **(**input**)**

**BEGIN**

**case** input **is**

**when** "0000"**=>** seg7 **<=**"01000000"**;** -- '0'

**when** "0001"**=>** seg7 **<=**"01111001"**;** -- '1'

**when** "0010"**=>** seg7 **<=**"00100100"**;** -- '2'

**when** "0011"**=>** seg7 **<=**"00110000"**;** -- '3'

**when** "0100"**=>** seg7 **<=**"00011001"**;** -- '4'

**when** "0101"**=>** seg7 **<=**"00010010"**;** -- '5'

**when** "0110"**=>** seg7 **<=**"00000010"**;** -- '6'

**when** "0111"**=>** seg7 **<=**"01111000"**;** -- '7'

**when** "1000"**=>** seg7 **<=**"00000000"**;** -- '8'

**when** "1001"**=>** seg7 **<=**"00011000"**;** -- '9'

**when** "1010"**=>** seg7 **<=**"00001000"**;** -- 'A'

**when** "1011"**=>** seg7 **<=**"00000000"**;** -- 'b'

**when** "1100"**=>** seg7 **<=**"01000110"**;** -- 'C'

**when** "1101"**=>** seg7 **<=**"00000000"**;** -- 'D'

**when** "1110"**=>** seg7 **<=**"00000110"**;** -- 'E'

**when** "1111"**=>** seg7 **<=**"00001110"**;** -- 'F'

**when** **others** **=>** **NULL;**

**end** **case;**

**end** **process;**

**end** Behavioral**;**

BREAD\_MACHINE\_DE10

--------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------

--LAB 2 FINITE STATE MACHINE LOGIC DESIGN

--FOR BREAD MAKER

--DATE: 9/16/2021

--AUTHOR: JORGE JURADO-GARCIA

--REV 1

--

--INPUTS: BREAD TYPE: REGULAR(SW0) OR DOUGH(SW1)

-- START(SW2), RESET(RSTB) (SW3)

--OUTPUTS: HEATER(1)[LEDR0], HEATER(2)[LEDR1], BEATER[LEDR2], BEEPER[LEDR3]. FINISHED[LEDR9]

--Timer count value will be displayed on HEX(0)

--PROCESS:

--1.Mix – HEATER[1] & HEATER [2] off, BEATER ON for (2) seconds, off (3) seconds (3x)

--2.Knead – BEATER ON for (10) seconds

--3.Rise1 – BEATER OFF, HEATER[1] for (10) seconds

--4.Beat down – HEATER[1] OFF, BEATER ON For (2) seconds

--5.Rise2 - BEATER OFF, HEATER[1] for (12) seconds

--6.Bake – HEATER[1] and HEATER[2] ON for (7) seconds

--7.Done – HEATER[1] AND HEATER[2] OFF, BEATER OFF, BEEPER ON for (5) seconds,

-- then light the Done LED until either a reset or the start of another cycle.

----------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** Bread\_machine\_DE10 **is**

**port(**

CLOCK\_50**:** **in** std\_logic**;**

SW**:** **in** std\_logic\_vector**(**3 **downto** 0**);**

HEX0**:** **out** std\_logic\_vector**(**7 **downto** 0**);**

LEDR**:** **out** std\_logic\_vector**(**9 **downto** 0**)** **);**

**end** **entity** Bread\_machine\_DE10**;**

**Architecture** behavioral **of** Bread\_machine\_DE10 **is**

---signals that will be used

**signal** CLK\_SIG**:** std\_logic**;** --intermediate value

**signal** CNT\_SIG**:** std\_logic\_vector**(**3 **downto** 0**);**

---------------------------------

--component declarations

---------------------------------

**component** clk\_3hz **is**

**port(**

i\_clk\_50MHz**:** **in** std\_logic**;**

i\_rstb**:** **in** std\_logic**;**

o\_clk\_3Hz**:** **out** std\_logic

**);**

**end** **component;**

**component** Bread\_machine **is**

**port(**

clk**,**start**,**rstb**,** dough**,**regular**:** **IN** std\_logic**;**

count**:** **OUT** std\_logic\_vector**(**3 **downto** 0**);**

heater1**,** heater2**,** beater**,** beeper**,** finished **:** **OUT** std\_logic

**);**

**end** **component;**

**component** to\_7seg **is**

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

seg7 **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**)**

**);**

**end** **component;**

--------------------------------------------

**begin**

-----------------------------------

--Device under test (DUT)

-----------------------------------

CK**:** clk\_3hz

**port** **map(**

i\_clk\_50MHz **=>** CLOCK\_50**,**

i\_rstb **=>** SW**(**3**),**

o\_clk\_3Hz **=>** CLK\_SIG

**);**

DUT**:** Bread\_machine

**port** **map(**

clk **=>** CLK\_SIG**,**

start **=>** SW**(**2**),**

rstb **=>** SW**(**3**),**

dough **=>** SW**(**1**),**

regular **=>** SW**(**0**),**

count**(**3 **downto** 0**)** **=>** CNT\_SIG**,**

-- ll(3 downto 0) => CNT\_SIG2,

heater1 **=>** LEDR**(**0**),**

heater2 **=>** LEDR**(**1**),**

beater **=>** LEDR**(**2**),**

beeper **=>** LEDR**(**3**),**

finished **=>** LEDR**(**9**)**

**);**

JFK**:** to\_7seg

**port** **map(**

input **=>** CNT\_SIG**,**

seg7 **=>** HEX0**(**7 **downto** 0**)**

**);**

**end** **architecture;**

BREAD\_MACHINE\_TB

--------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------

--LAB 2 FINITE STATE MACHINE LOGIC DESIGN

--FOR BREAD MAKER\_tb

--DATE: 9/14/2021

--AUTHOR: JORGE JURADO-GARCIA

--REV 1

--included output logic of fsm 9/15/21

--

--INPUTS: BREAD TYPE: REGULAR OR DOUGH

-- START, RESET(RSTB)

--

--OUTPUTS: HEATER(1), HEATER(2), BEATER, BEEPER.

--PROCESS:

--1.Mix – HEATER[1] & HEATER [2] off, BEATER ON for (2) seconds, off (3) seconds (3x)

--2.Knead – BEATER ON for (10) seconds

--3.Rise1 – BEATER OFF, HEATER[1] for (10) seconds

--4.Beat down – HEATER[1] OFF, BEATER ON For (2) seconds

--5.Rise2 - BEATER OFF, HEATER[1] for (12) seconds

--6.Bake – HEATER[1] and HEATER[2] ON for (7) seconds

--7.Done – HEATER[1] AND HEATER[2] OFF, BEATER OFF, BEEPER ON for (5) seconds,

-- then light the Done LED until either a reset or the start of another cycle.

----------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** Bread\_machine\_tb **is**

--non entry

**end** **entity;**

**architecture** testbench **of** Bread\_machine\_tb **is**

**signal** CLK**:** std\_logic**;**

**signal** START**:** std\_logic**;**

**signal** RSTB**:** std\_logic**;**

**signal** DOUGH**:** std\_logic**;**

**signal** REGULAR**:** std\_logic**;**

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

**signal** HEATER1**:** std\_logic**;**

**signal** HEATER2**:** std\_logic**;**

**signal** BEATER**:** std\_logic**;**

**signal** BEEPER**:** std\_logic**;**

**signal** FINISHED**:** std\_logic**;**

**constant** PER**:** time **:=** 15 ns**;**

------------------------------------

--Component Protype

------------------------------------

**component** Bread\_machine **is**

**port(**

clk**,**start**,**rstb**,** dough**,**regular**:** **IN** std\_logic**;**

count**:** **OUT** std\_logic\_vector**(**3 **downto** 0**);**

heater1**,** heater2**,** beater**,** beeper**,** finished **:** **OUT** std\_logic

**);**

**end** **component** Bread\_machine**;**

--end component

----------------------------------------------

**begin**

------------------------------------

--DUT(Device under testing)

------------------------------------

DUT**:** Bread\_machine

**port** **map(**

--inputs

clk **=>** CLK**,**

start**=>** START**,**

rstb **=>** RSTB**,**

dough **=>** DOUGH**,**

regular **=>** REGULAR**,**

--outputs

count **=>** CNT**,**

heater1 **=>** HEATER1**,**

heater2 **=>** HEATER2**,**

beater **=>** BEATER**,**

beeper **=>** BEEPER**,**

finished **=>** FINISHED

**);**

----------------------------------------

--Test Processes

----------------------------------------

--clock process

clock**:** **process**

**begin**

CLK **<=** '0'**;**

**wait** **for** **(**PER**/**2**);**

infinite**:** **loop**

CLK **<=** **not** CLK**;** **wait** **for** PER**/**2**;**

**end** **loop;**

**end** **process;**

--Reset Process

reset**:** **process**

**begin**

RSTB **<=** '0'**;** **wait** **for** 2**\***PER**;**

RSTB **<=** '1'**;** **wait;**

**end** **process** reset**;**

--run process

run**:** **process**

**begin**

-- intitalize inputs

DOUGH **<=** '0'**;**

REGULAR **<=** '0'**;**

START **<=** '0'**;**

**wait** **for** 4**\***PER**;**

DOUGH **<=** '1'**;**

REGULAR **<=** '1'**;**

START **<=** '1'**;**

**wait** **for** 5**\***PER**;**

--wait for reset

**wait** **for** PER**;**

DOUGH **<=** '1'**;**

REGULAR **<=** '0'**;**

START **<=** '1'**;**

**wait** **for** 45**\***PER**;**

DOUGH **<=** '0'**;**

REGULAR **<=** '1'**;**

START **<=** '1'**;**

**wait;**

**end** **process** run**;**

-----------------------------

--End test processes

--------------------------------

**end** **architecture;**