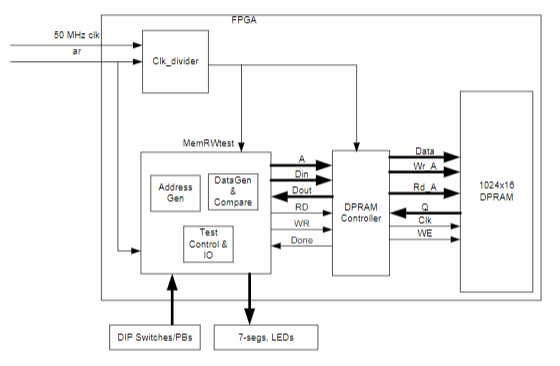
**Project 3 – DE2-115 Internal Memory Controller**

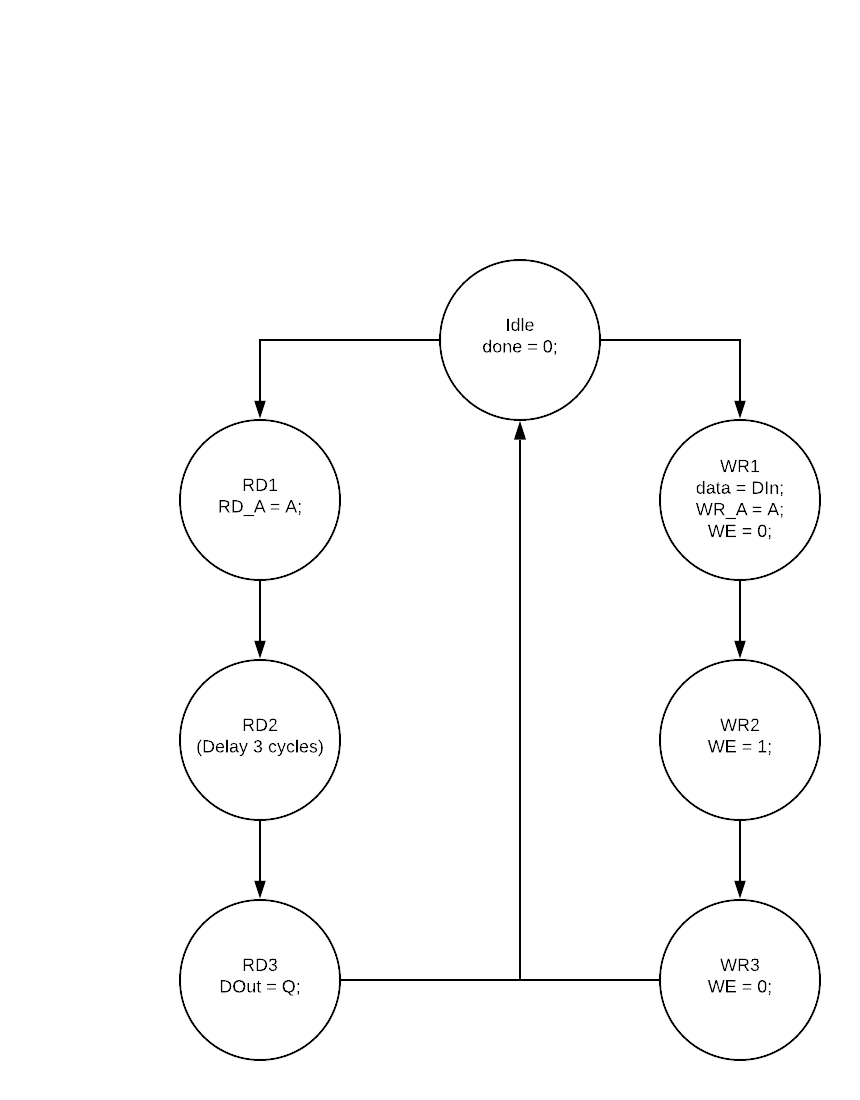
**Scope:**

The goal of this project was to design, implement, and test a memory controller for an internal DPRAM. The project contained 3 modules, the DPRAM, controller, and a test module. The goal of the test module was to perform an internal test and also offer the user I/O to load and read data from any of the memory locations.

**Design Strategy:**

The Project was built to the specifications in the diagram below excluding the clock divider which was not needed.

The DPRAM module was created in Quartus through the IP setup wizard. The controller was implemented following the state machine design we went over in class. State diagram shown below. The MemRWtest module implemented two similar state machines, one for the internal test, and one for the I/O read and write. The active state machine is controlled by a switch on the board. When in the internal test mode, the button on the far left is pushed to start the test, at the end of the test the green LED will light up if the test passes or the red LED will light up if the test fails. In use I/O mode the buttons from left to right are address, read, write, and reset. The address button will take in the 10 rightmost switches as the address and the write uses the 16 rightmost to take in the data to be written.

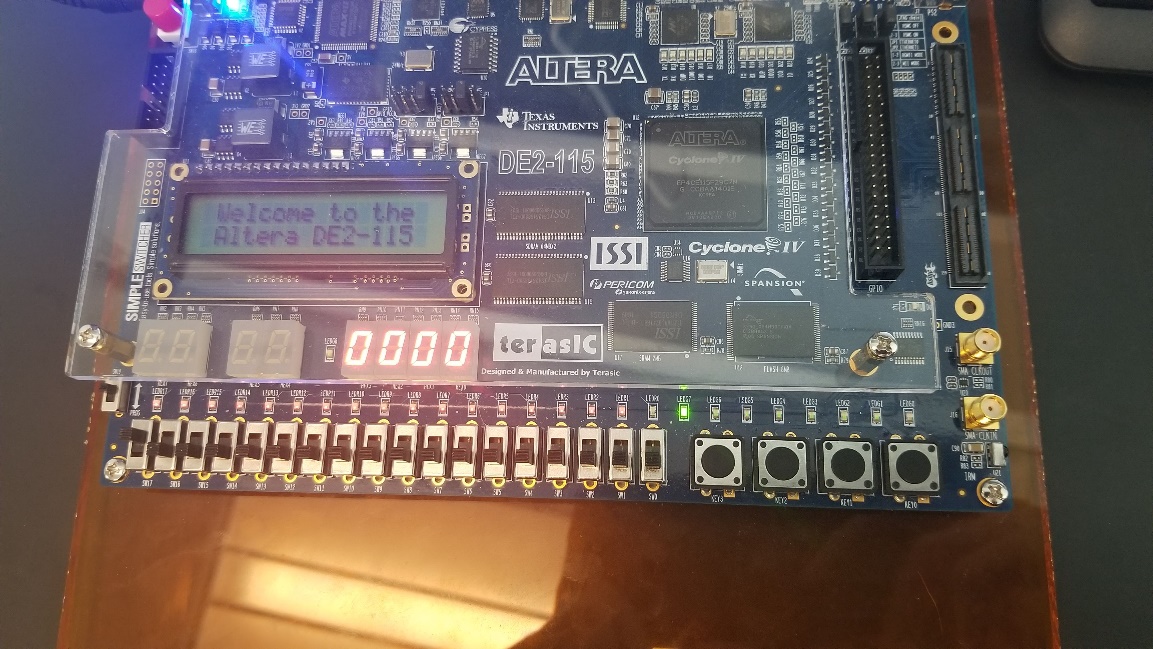


**Results:**

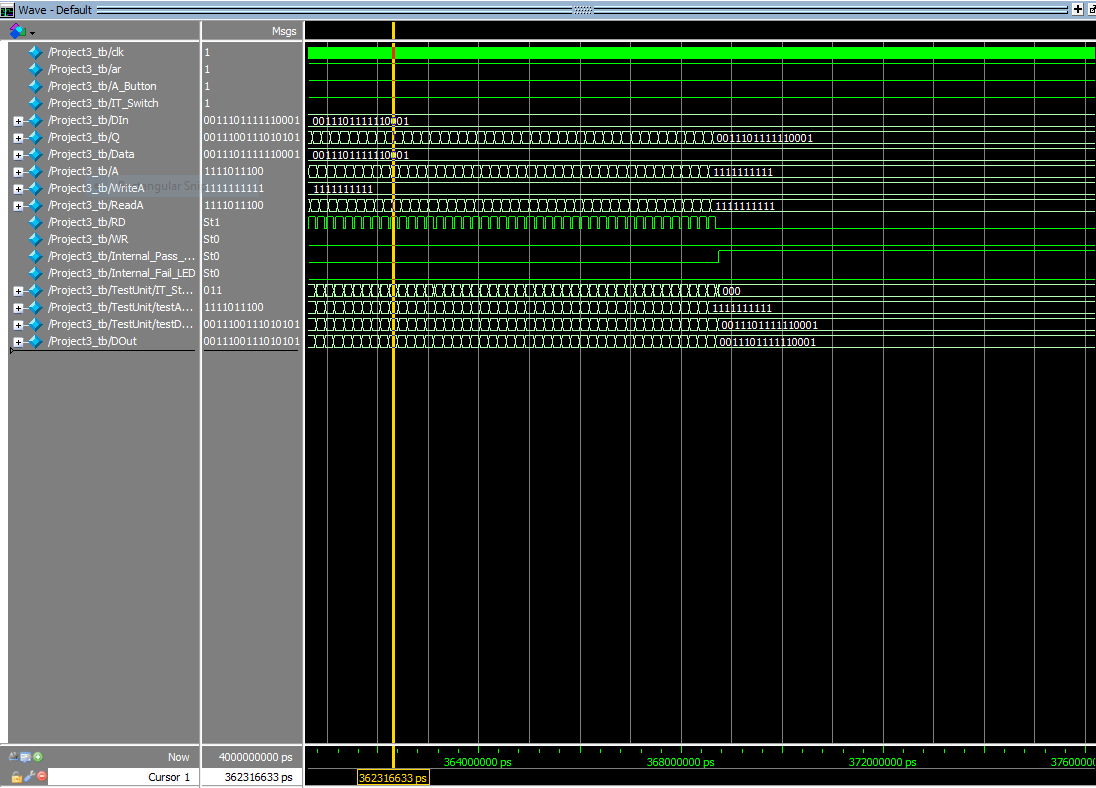
The following link contains video showing the I/O working as expected. The hex value F is stored in memory location 1 and the hex value E is store in location 2.

<https://photos.app.goo.gl/BiKGWyWHXTih5EFc8>

The picture below shows the board in internal test mode with the green LED on showing it has passed the test.



From Quartus the Fmax is 99.39 MHz and the board uses 232 logic elements which is <1%. The DPRAM module is clearly seen as implemented with Quartus showing 16,384 bits which is equivalent to 1024 X 16.

The screenshot below shows the internal test running in model sim where it compares the data written which was the address \* 15 to show all 16 bits working, to the address being read at the time \* 15 as well. When all 1024 addresses are read the internal pass LED turns on which is also shown.

**What I Learned:**

In doing this project I learned more about the importance of planning out the code before you write it. Having created the state diagrams before I even started writing any code, I found that I was able to easily translate the diagrams into code quickly and was able to have success after only fixing a couple of syntax errors.

**Conclusions:**

This project went very smoothly after initial planning. The hardest thing I actually had to deal with was modelsim not working on my personal computer. I spent more than a couple hours trying to debug it before attempting on a lab computer where it worked fine on my first attempt.

I feel like this project helped me to increase my understanding of how memory works at a level I had not seen before. This will definitely be a project I pull from in the future as I work on other digital design projects.

**Appendix:**

Project\_3\_top:

/\*

/ Jack Mravunac

/ Project3\_top.v

/ Top file for DPRAM controller project

/ 27 February 2020

\*/

`timescale 100 ns **/** 1 ns

**module** Project3\_top**(**clk**,** ar**,** universalIn**,** A\_Button**,** Rd\_Button**,** Wr\_Button**,** IT\_Switch**,**

Done\_LED**,** Internal\_Pass\_LED**,** Internal\_Fail\_LED**,** SevenSeg\_Zero\_Out**,** SevenSeg\_One\_Out**,** SevenSeg\_Two\_Out**,** SevenSeg\_Three\_Out**);**

**input** clk**,** ar**,** A\_Button**,** Rd\_Button**,** Wr\_Button**,** IT\_Switch**;**

**input** **[**15**:**0**]** universalIn**;**

**output** Done\_LED**,** Internal\_Pass\_LED**,** Internal\_Fail\_LED**;**

**output** **[**6**:**0**]** SevenSeg\_Zero\_Out**,** SevenSeg\_One\_Out**,** SevenSeg\_Two\_Out**,** SevenSeg\_Three\_Out**;**

**wire** **[**3**:**0**]** SevenSeg\_Zero**,** SevenSeg\_One**,** SevenSeg\_Two**,** SevenSeg\_Three**;**

**wire** **[**15**:**0**]** DOut**,** DIn**,** Q**,** Data**;**

**wire** **[**9**:**0**]** A**,** WriteA**,** ReadA**;**

**wire** Done**,** RD**,** WR**,** WE**;**

MemRWTest TestUnit**(.**clk**(**clk**),** **.**ar**(**ar**),** **.**UniversalIn**(**universalIn**),** **.**DOut**(**DOut**),** **.**Done**(**Done**),** **.**A\_Button**(**A\_Button**),** **.**Rd\_Button**(**Rd\_Button**),** **.**Wr\_Button**(**Wr\_Button**),** **.**IT\_Switch**(**IT\_Switch**),**

**.**A**(**A**),** **.**DIn**(**DIn**),** **.**RD**(**RD**),** **.**WR**(**WR**),** **.**Done\_LED**(**Done\_LED**),** **.**Internal\_Pass\_LED**(**Internal\_Pass\_LED**),** **.**Internal\_Fail\_LED**(**Internal\_Fail\_LED**),**

**.**SevenSeg\_Zero**(**SevenSeg\_Zero**),** **.**SevenSeg\_One**(**SevenSeg\_One**),** **.**SevenSeg\_Two**(**SevenSeg\_Two**),** **.**SevenSeg\_Three**(**SevenSeg\_Three**));**

sevseg\_dec SegZero**(.**x\_in**(**SevenSeg\_Zero**),** **.**segs**(**SevenSeg\_Zero\_Out**));**

sevseg\_dec SegOne**(.**x\_in**(**SevenSeg\_One**),** **.**segs**(**SevenSeg\_One\_Out**));**

sevseg\_dec SegTwo**(.**x\_in**(**SevenSeg\_Two**),** **.**segs**(**SevenSeg\_Two\_Out**));**

sevseg\_dec SegThree**(.**x\_in**(**SevenSeg\_Three**),** **.**segs**(**SevenSeg\_Three\_Out**));**

DPRAM\_Controller Controller**(.**clk**(**clk**),** **.**ar**(**ar**),** **.**RD**(**RD**),** **.**WR**(**WR**),** **.**A**(**A**),** **.**DIn**(**DIn**),** **.**Q**(**Q**),**

**.**DOut**(**DOut**),** **.**Data**(**Data**),** **.**Wr\_A**(**WriteA**),** **.**Rd\_A**(**ReadA**),** **.**Done**(**Done**),** **.**WE**(**WE**));**

DPRAM Ram**(.**clock**(**clk**),** **.**data**(**Data**),** **.**rdaddress**(**ReadA**),** **.**wraddress**(**WriteA**),** **.**wren**(**WE**),**

.q(Q));

endmodule

MemRWTest:

/\*

/ Jack Mravunac

/ MemRWTest.v

/ Contains I/O and self tests for the DPRAM

/ 27 February 2020

\*/

`timescale 100 ns **/** 1 ns

**module** MemRWTest**(**clk**,** ar**,** UniversalIn**,** DOut**,** Done**,** A\_Button**,** Rd\_Button**,** Wr\_Button**,** IT\_Switch**,**

A**,** DIn**,** RD**,** WR**,** Done\_LED**,** Internal\_Pass\_LED**,** Internal\_Fail\_LED**,** SevenSeg\_Zero**,** SevenSeg\_One**,** SevenSeg\_Two**,** SevenSeg\_Three**);**

**input** clk**,** ar**,** Done**,** A\_Button**,** Rd\_Button**,** Wr\_Button**,** IT\_Switch**;**

**input** **[**15**:**0**]** UniversalIn**,** DOut**;**

**output** **reg** **[**3**:**0**]** SevenSeg\_Zero**,** SevenSeg\_One**,** SevenSeg\_Two**,** SevenSeg\_Three**;**

**output** **reg** **[**9**:**0**]** A**;**

**output** **reg** **[**15**:**0**]** DIn**;**

**output** **reg** RD**,** WR**,** Done\_LED**,** Internal\_Pass\_LED**,** Internal\_Fail\_LED**;**

**parameter** **[**2**:**0**]** Idle **=** 3'b0**,** Address **=** 3'b001**,** Wr1 **=** 3'b010**,** Wr2 **=** 3'b111**,** Wr3 **=** 3'b011**,** Rd1 **=** 3'b100**,** Rd2 **=** 3'b101**,** Rd3 **=** 3'b110**;**

**parameter** **[**2**:**0**]** Idle2 **=** 3'b0**,** AddrImp **=** 3'b001**,** Write **=** 3'b010**,** Read **=** 3'b011**,** Check **=** 3'b100**,** Success **=** 3'b101**,** Fail **=** 3'b110**;**

**reg** **[**2**:**0**]** User\_State**;**

**reg** **[**2**:**0**]** IT\_State**;**

**reg** **[**2**:**0**]** delay**,** delay2**,** delay3**;**

**reg** **[**9**:**0**]** testAddr**;**

**reg** **[**15**:**0**]** testData**;**

**always** **@(negedge** ar **or** **posedge** clk**)**

**if(~**ar**)**

**begin**

A **=** 10'b0**;**

DIn **=** 16'b0**;**

RD **=** 1'b0**;**

WR **=** 1'b0**;**

testAddr **=** 10'b0**;**

testData **=** 16'b0**;**

Done\_LED **=** 1'b0**;**

Internal\_Pass\_LED **=** 1'b0**;**

Internal\_Fail\_LED **=** 1'b0**;**

SevenSeg\_Zero **=** 4'b0**;**

SevenSeg\_One **=** 4'b0**;**

SevenSeg\_Two **=** 4'b0**;**

SevenSeg\_Three **=** 4'b0**;**

**end**

**else**

**begin**

**if(**IT\_Switch**)**

**begin**

**case(**IT\_State**)**

Idle2**:**

**begin**

**if(~**A\_Button**)**

IT\_State **=** AddrImp**;**

**end**

AddrImp**:**

**begin**

testAddr **=** testAddr **+** 1**;**

testData **=** testAddr **\*** 15**;**

delay2 **=** 3'b0**;**

delay3 **=** 3'b0**;**

A **=** testAddr**;**

DIn **=** testData**;**

IT\_State **=** Write**;**

**end**

Write**:**

**begin**

WR **=** 1'b1**;**

**if(**delay2 **>** 3'b110**)**

**begin**

**if(**testAddr **>** 1022**)**

**begin**

testAddr **=** 0**;**

testData **=** 0**;**

IT\_State **=** Read**;**

**end**

**else**

IT\_State **=** AddrImp**;**

**end**

**else**

delay2 **=** delay2 **+** 1**;**

**end**

Read**:**

**begin**

WR **=** 1'b0**;**

delay2 **=** 3'b0**;**

A **=** testAddr**;**

RD **=** 1'b1**;**

**if(**delay3 **>** 3'b110**)**

**begin**

RD **=** 1'b0**;**

delay3 **=** 3'b0**;**

IT\_State **=** Check**;**

**end**

**else**

delay3 **=** delay3 **+** 1**;**

**end**

Check**:**

**begin**

testData **=** testAddr **\*** 15**;**

**if(**testData **!=** DOut**)**

IT\_State **=** Fail**;**

**else**

**begin**

**if(**testAddr **>** 1022**)**

IT\_State **=** Success**;**

**else**

**begin**

testAddr **=** testAddr **+** 1**;**

IT\_State **=** Read**;**

**end**

**end**

**end**

Success**:**

**begin**

Internal\_Pass\_LED **=** 1'b1**;**

Internal\_Fail\_LED **=** 1'b0**;**

IT\_State **=** Idle2**;**

**end**

Fail**:**

**begin**

Internal\_Pass\_LED **=** 1'b0**;**

Internal\_Fail\_LED **=** 1'b1**;**

IT\_State **=** Idle2**;**

**end**

**default:**

**begin**

IT\_State **=** Idle2**;**

**end**

**endcase**

**end**

**else**

**begin**

**case(**User\_State**)**

Idle**:**

**begin**

**if(~**A\_Button**)**

User\_State **=** Address**;**

**else** **if(~**Rd\_Button**)**

User\_State **=** Rd1**;**

**else** **if(~**Wr\_Button**)**

User\_State = Wr1;

end

Address:

begin

A = UniversalIn[9:0];

User\_State = Idle;

end

Rd1:

begin

Done\_LED = 1'b0;

delay = 3'b0;

RD = 1'b1;

User\_State = Rd2;

end

Rd2:

begin

if(delay > 3'b110)

User\_State = Rd3;

else

delay = delay + 1;

end

Rd3:

begin

RD = 1'b0;

SevenSeg\_Zero = DOut[3:0];

SevenSeg\_One = DOut[7:4];

SevenSeg\_Two = DOut[11:8];

SevenSeg\_Three = DOut[15:12];

Done\_LED = 1'b1;

User\_State = Idle;

end

Wr1:

begin

Done\_LED = 1'b0;

DIn = UniversalIn[15:0];

User\_State = Wr2;

end

Wr2:

begin

WR = 1'b1;

User\_State = Wr3;

end

Wr3:

begin

WR = 1'b0;

Done\_LED = 1'b1;

User\_State = Idle;

end

default:

begin

User\_State = Idle;

end

endcase

end

end

endmodule

DPRAM\_Controller:

/\*

/ Jack Mravunac

/ DPRAM\_Controller.v

/ Controller for the DPRAM

/ 27 February 2020

\*/

`timescale 100 ns **/** 1 ns

**module** DPRAM\_Controller**(**clk**,** ar**,** RD**,** WR**,** A**,** DIn**,** Q**,**

DOut**,** Data**,** Wr\_A**,** Rd\_A**,** Done**,** WE**);**

**input** clk**,** RD**,** WR**,** ar**;**

**input** **[**9**:**0**]** A**;**

**input** **[**15**:**0**]** DIn**,** Q**;**

**output** **reg** **[**15**:**0**]** DOut**,** Data**;**

**output** **reg** **[**9**:**0**]** Wr\_A**,** Rd\_A**;**

**output** **reg** Done**,** WE**;**

**parameter** **[**2**:**0**]** Idle **=** 3'b0**,** Wr1 **=** 3'b001**,** Wr2 **=** 3'b010**,** Wr3 **=** 3'b011**,** Rd1 **=** 3'b100**,** Rd2 **=** 3'b101**,** Rd3 **=** 3'b110**;**

**reg** **[**2**:**0**]** state**;**

**reg** **[**1**:**0**]** delay**;**

**always** **@(negedge** ar **or** **posedge** clk**)**

**if(~**ar**)**

**begin**

state **=** Idle**;**

DOut **=** 16'b0**;**

Data **=** 16'b0**;**

Wr\_A **=** 10'b0**;**

Rd\_A **=** 10'b0**;**

Done **=** 1'b0**;**

WE **=** 1'b0**;**

**end**

**else**

**begin**

**case(**state**)**

Idle**:**

begin

Done = 1'b0;

if(RD)

state = Rd1;

else if(WR)

state = Wr1;

end

Wr1:

begin

Data = DIn;

Wr\_A = A;

WE = 1'b0;

state = Wr2;

end

Wr2:

begin

WE = 1'b1;

state = Wr3;

end

Wr3:

begin

WE = 1'b0;

state = Idle;

end

Rd1:

begin

Rd\_A = A;

delay = 2'b0;

state = Rd2;

end

Rd2:

begin

if(delay > 2'b10)

state = Rd3;

else

delay = delay + 1;

end

Rd3:

begin

DOut = Q;

delay = 2'b0;

state = Idle;

end

default:

begin

state = Idle;

end

endcase

end

endmodule

Project3\_tb:

/\*

/ Jack Mravunac

/ Project3\_tb.v

/ Contains test bence for the internal test

/ 27 February 2020

\*/

`timescale 1 ns **/** 1 ns

**module** Project3\_tb**;**

**reg** clk**,** ar**,** A\_Button**,** Rd\_Button**,** Wr\_Button**,** IT\_Switch**;**

**reg** **[**15**:**0**]** universalIn**;**

**wire** **[**3**:**0**]** SevenSeg\_Zero**,** SevenSeg\_One**,** SevenSeg\_Two**,** SevenSeg\_Three**;**

**wire** **[**15**:**0**]** DOut**,** DIn**,** Q**,** Data**;**

**wire** **[**9**:**0**]** A**,** WriteA**,** ReadA**;**

**wire** Done**,** RD**,** WR**,** WE**;**

MemRWTest TestUnit**(.**clk**(**clk**),** **.**ar**(**ar**),** **.**UniversalIn**(**universalIn**),** **.**DOut**(**DOut**),** **.**Done**(**Done**),** **.**A\_Button**(**A\_Button**),** **.**Rd\_Button**(**Rd\_Button**),** **.**Wr\_Button**(**Wr\_Button**),** **.**IT\_Switch**(**IT\_Switch**),**

**.**A**(**A**),** **.**DIn**(**DIn**),** **.**RD**(**RD**),** **.**WR**(**WR**),** **.**Done\_LED**(**Done\_LED**),** **.**Internal\_Pass\_LED**(**Internal\_Pass\_LED**),** **.**Internal\_Fail\_LED**(**Internal\_Fail\_LED**),**

**.**SevenSeg\_Zero**(**SevenSeg\_Zero**),** **.**SevenSeg\_One**(**SevenSeg\_One**),** **.**SevenSeg\_Two**(**SevenSeg\_Two**),** **.**SevenSeg\_Three**(**SevenSeg\_Three**));**

sevseg\_dec SegZero**(.**x\_in**(**SevenSeg\_Zero**),** **.**segs**(**SevenSeg\_Zero\_Out**));**

sevseg\_dec SegOne**(.**x\_in**(**SevenSeg\_One**),** **.**segs**(**SevenSeg\_One\_Out**));**

sevseg\_dec SegTwo**(.**x\_in**(**SevenSeg\_Two**),** **.**segs**(**SevenSeg\_Two\_Out**));**

sevseg\_dec SegThree**(.**x\_in**(**SevenSeg\_Three**),** **.**segs**(**SevenSeg\_Three\_Out**));**

DPRAM\_Controller Controller**(.**clk**(**clk**),** **.**ar**(**ar**),** **.**RD**(**RD**),** **.**WR**(**WR**),** **.**A**(**A**),** **.**DIn**(**DIn**),** **.**Q**(**Q**),**

**.**DOut**(**DOut**),** **.**Data**(**Data**),** **.**Wr\_A**(**WriteA**),** **.**Rd\_A**(**ReadA**),** **.**Done**(**Done**),** **.**WE**(**WE**));**

DPRAM Ram**(.**clock**(**clk**),** **.**data**(**Data**),** **.**rdaddress**(**ReadA**),** **.**wraddress**(**WriteA**),** **.**wren**(**WE**),**

**.**q**(**Q**));**

**initial**

**begin**

clk **=** 1'b0**;** // set to 0 so toggling can occur

ar **=** 1'b1**;** // Start reset at 1

**#**1 ar **=** 1'b0**;** // Set reset to 0 after 5 ns

**#**20 ar **=** 1'b1**;** // Set reset to 1

**#**20 IT\_Switch **=** 1'b1**;**

**#**100 A\_Button **=** 1'b1**;**

**#**100 A\_Button **=** 1'b0**;**

**#**20 A\_Button **=** 1'b1**;**

**end**

// Controls the test clock

**always**

**#**10 clk **=** **~**clk**;** // For 20 ns period (50 MHz)

**endmodule**

sevseg\_dec:

// Seven-segment decoder for hexadecimal values

// seven\_seg\_decoder.v

// Don M. Gruenbacher

// Jan. 23, 2006

**module** sevseg\_dec**(**x\_in**,** segs**);**

**input** **[**3**:**0**]** x\_in**;**

**output** **[**6**:**0**]** segs**;**

**assign** segs **=**

**(**x\_in **==** 4'h0**)** **?** 7'b1000000 **:**

**(**x\_in **==** 4'h1**)** **?** 7'b1111001 **:**

**(**x\_in **==** 4'h2**)** **?** 7'b0100100 **:**

**(**x\_in **==** 4'h3**)** **?** 7'b0110000 **:**

**(**x\_in **==** 4'h4**)** **?** 7'b0011001 **:**

**(**x\_in **==** 4'h5**)** **?** 7'b0010010 **:**

**(**x\_in **==** 4'h6**)** **?** 7'b0000010 **:**

**(**x\_in **==** 4'h7**)** **?** 7'b1111000 **:**

**(**x\_in **==** 4'h8**)** **?** 7'b0000000 **:**

**(**x\_in **==** 4'h9**)** **?** 7'b0010000 **:**

**(**x\_in **==** 4'ha**)** **?** 7'b0001000 **:**

**(**x\_in **==** 4'hb**)** **?** 7'b0000011 **:**

**(**x\_in **==** 4'hc**)** **?** 7'b1000110 **:**

**(**x\_in **==** 4'hd**)** **?** 7'b0100001 **:**

**(**x\_in **==** 4'he**)** **?** 7'b0000110 **:**

7'b0001110 **;**

**endmodule**

DPRAM:

// megafunction wizard: %RAM: 2-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

// ============================================================

// File Name: DPRAM.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

// ============================================================

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!

//

// 17.1.0 Build 590 10/25/2017 SJ Lite Edition

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Copyright (C) 2017 Intel Corporation. All rights reserved.

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

//and other software and tools, and its AMPP 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.

// synopsys translate\_off

`timescale 1 ps **/** 1 ps

// synopsys translate\_on

**module** DPRAM **(**

clock**,**

data,

rdaddress,

wraddress,

wren,

q);

input clock;

input [15:0] data;

input [9:0] rdaddress;

input [9:0] wraddress;

input wren;

output [15:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

tri0 wren;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [15:0] sub\_wire0;

wire [15:0] q = sub\_wire0[15:0];

altsyncram altsyncram\_component (

.address\_a (wraddress),

.address\_b (rdaddress),

.clock0 (clock),

.data\_a (data),

.wren\_a (wren),

.q\_b (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.addressstall\_a (1'b0),

.addressstall\_b (1'b0),

.byteena\_a (1'b1),

.byteena\_b (1'b1),

.clock1 (1'b1),

.clocken0 (1'b1),

.clocken1 (1'b1),

.clocken2 (1'b1),

.clocken3 (1'b1),

.data\_b ({16{1'b1}}),

.eccstatus (),

.q\_a (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_b (1'b0));

defparam

altsyncram\_component.address\_aclr\_b = "NONE",

altsyncram\_component.address\_reg\_b = "CLOCK0",

altsyncram\_component.clock\_enable\_input\_a = "BYPASS",

altsyncram\_component.clock\_enable\_input\_b = "BYPASS",

altsyncram\_component.clock\_enable\_output\_b = "BYPASS",

altsyncram\_component.intended\_device\_family = "Cyclone IV E",

altsyncram\_component.lpm\_type = "altsyncram",

altsyncram\_component.numwords\_a = 1024,

altsyncram\_component.numwords\_b = 1024,

altsyncram\_component.operation\_mode = "DUAL\_PORT",

altsyncram\_component.outdata\_aclr\_b = "NONE",

altsyncram\_component.outdata\_reg\_b = "CLOCK0",

altsyncram\_component.power\_up\_uninitialized = "FALSE",

altsyncram\_component.read\_during\_write\_mode\_mixed\_ports = "DONT\_CARE",

altsyncram\_component.widthad\_a = 10,

altsyncram\_component.widthad\_b = 10,

altsyncram\_component.width\_a = 16,

altsyncram\_component.width\_b = 16,

altsyncram\_component.width\_byteena\_a = 1;

endmodule

// ============================================================

// CNX file retrieval info

// ============================================================

// Retrieval info: PRIVATE: ADDRESSSTALL\_A NUMERIC "0"

// Retrieval info: PRIVATE: ADDRESSSTALL\_B NUMERIC "0"

// Retrieval info: PRIVATE: BYTEENA\_ACLR\_A NUMERIC "0"

// Retrieval info: PRIVATE: BYTEENA\_ACLR\_B NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_ENABLE\_A NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_ENABLE\_B NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "8"

// Retrieval info: PRIVATE: BlankMemory NUMERIC "1"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_INPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_INPUT\_B NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_OUTPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_OUTPUT\_B NUMERIC "0"

// Retrieval info: PRIVATE: CLRdata NUMERIC "0"

// Retrieval info: PRIVATE: CLRq NUMERIC "0"

// Retrieval info: PRIVATE: CLRrdaddress NUMERIC "0"

// Retrieval info: PRIVATE: CLRrren NUMERIC "0"

// Retrieval info: PRIVATE: CLRwraddress NUMERIC "0"

// Retrieval info: PRIVATE: CLRwren NUMERIC "0"

// Retrieval info: PRIVATE: Clock NUMERIC "0"

// Retrieval info: PRIVATE: Clock\_A NUMERIC "0"

// Retrieval info: PRIVATE: Clock\_B NUMERIC "0"

// Retrieval info: PRIVATE: IMPLEMENT\_IN\_LES NUMERIC "0"

// Retrieval info: PRIVATE: INDATA\_ACLR\_B NUMERIC "0"

// Retrieval info: PRIVATE: INDATA\_REG\_B NUMERIC "0"

// Retrieval info: PRIVATE: INIT\_FILE\_LAYOUT STRING "PORT\_B"

// Retrieval info: PRIVATE: INIT\_TO\_SIM\_X NUMERIC "0"

// Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone IV E"

// Retrieval info: PRIVATE: JTAG\_ENABLED NUMERIC "0"

// Retrieval info: PRIVATE: JTAG\_ID STRING "NONE"

// Retrieval info: PRIVATE: MAXIMUM\_DEPTH NUMERIC "0"

// Retrieval info: PRIVATE: MEMSIZE NUMERIC "16384"

// Retrieval info: PRIVATE: MEM\_IN\_BITS NUMERIC "0"

// Retrieval info: PRIVATE: MIFfilename STRING ""

// Retrieval info: PRIVATE: OPERATION\_MODE NUMERIC "2"

// Retrieval info: PRIVATE: OUTDATA\_ACLR\_B NUMERIC "0"

// Retrieval info: PRIVATE: OUTDATA\_REG\_B NUMERIC "1"

// Retrieval info: PRIVATE: RAM\_BLOCK\_TYPE NUMERIC "0"

// Retrieval info: PRIVATE: READ\_DURING\_WRITE\_MODE\_MIXED\_PORTS NUMERIC "2"

// Retrieval info: PRIVATE: READ\_DURING\_WRITE\_MODE\_PORT\_A NUMERIC "3"

// Retrieval info: PRIVATE: READ\_DURING\_WRITE\_MODE\_PORT\_B NUMERIC "3"

// Retrieval info: PRIVATE: REGdata NUMERIC "1"

// Retrieval info: PRIVATE: REGq NUMERIC "1"

// Retrieval info: PRIVATE: REGrdaddress NUMERIC "1"

// Retrieval info: PRIVATE: REGrren NUMERIC "1"

// Retrieval info: PRIVATE: REGwraddress NUMERIC "1"

// Retrieval info: PRIVATE: REGwren NUMERIC "1"

// Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "0"

// Retrieval info: PRIVATE: USE\_DIFF\_CLKEN NUMERIC "0"

// Retrieval info: PRIVATE: UseDPRAM NUMERIC "1"

// Retrieval info: PRIVATE: VarWidth NUMERIC "0"

// Retrieval info: PRIVATE: WIDTH\_READ\_A NUMERIC "16"

// Retrieval info: PRIVATE: WIDTH\_READ\_B NUMERIC "16"

// Retrieval info: PRIVATE: WIDTH\_WRITE\_A NUMERIC "16"

// Retrieval info: PRIVATE: WIDTH\_WRITE\_B NUMERIC "16"

// Retrieval info: PRIVATE: WRADDR\_ACLR\_B NUMERIC "0"

// Retrieval info: PRIVATE: WRADDR\_REG\_B NUMERIC "0"

// Retrieval info: PRIVATE: WRCTRL\_ACLR\_B NUMERIC "0"

// Retrieval info: PRIVATE: enable NUMERIC "0"

// Retrieval info: PRIVATE: rden NUMERIC "0"

// Retrieval info: LIBRARY: altera\_mf altera\_mf.altera\_mf\_components.all

// Retrieval info: CONSTANT: ADDRESS\_ACLR\_B STRING "NONE"

// Retrieval info: CONSTANT: ADDRESS\_REG\_B STRING "CLOCK0"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_INPUT\_A STRING "BYPASS"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_INPUT\_B STRING "BYPASS"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_OUTPUT\_B STRING "BYPASS"

// Retrieval info: CONSTANT: INTENDED\_DEVICE\_FAMILY STRING "Cyclone IV E"

// Retrieval info: CONSTANT: LPM\_TYPE STRING "altsyncram"

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "1024"

// Retrieval info: CONSTANT: NUMWORDS\_B NUMERIC "1024"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "DUAL\_PORT"

// Retrieval info: CONSTANT: OUTDATA\_ACLR\_B STRING "NONE"

// Retrieval info: CONSTANT: OUTDATA\_REG\_B STRING "CLOCK0"

// Retrieval info: CONSTANT: POWER\_UP\_UNINITIALIZED STRING "FALSE"

// Retrieval info: CONSTANT: READ\_DURING\_WRITE\_MODE\_MIXED\_PORTS STRING "DONT\_CARE"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "10"

// Retrieval info: CONSTANT: WIDTHAD\_B NUMERIC "10"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "16"

// Retrieval info: CONSTANT: WIDTH\_B NUMERIC "16"

// Retrieval info: CONSTANT: WIDTH\_BYTEENA\_A NUMERIC "1"

// Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT VCC "clock"

// Retrieval info: USED\_PORT: data 0 0 16 0 INPUT NODEFVAL "data[15..0]"

// Retrieval info: USED\_PORT: q 0 0 16 0 OUTPUT NODEFVAL "q[15..0]"

// Retrieval info: USED\_PORT: rdaddress 0 0 10 0 INPUT NODEFVAL "rdaddress[9..0]"

// Retrieval info: USED\_PORT: wraddress 0 0 10 0 INPUT NODEFVAL "wraddress[9..0]"

// Retrieval info: USED\_PORT: wren 0 0 0 0 INPUT GND "wren"

// Retrieval info: CONNECT: @address\_a 0 0 10 0 wraddress 0 0 10 0

// Retrieval info: CONNECT: @address\_b 0 0 10 0 rdaddress 0 0 10 0

// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

// Retrieval info: CONNECT: @data\_a 0 0 16 0 data 0 0 16 0

// Retrieval info: CONNECT: @wren\_a 0 0 0 0 wren 0 0 0 0

// Retrieval info: CONNECT: q 0 0 16 0 @q\_b 0 0 16 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL DPRAM.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL DPRAM.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL DPRAM.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL DPRAM.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL DPRAM\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL DPRAM\_bb.v FALSE

// Retrieval info: LIB\_FILE: altera\_mf