ECE241 Final Project

Final Report

Boxuan Wang, Qianhan Cui

# 1.0 Introduction

Different from micro-processors, the inherent parallelism of FPGA provides it with advantage when multiple dissected workloads need simultaneous computation[1], or when inputs and outputs have varying clock rate or bandwidth. [2] However, the advantage of FPGA has yet to be well demonstrated in the game industry. [3] The goal of the project is to create a simple two-player 2D combat game to demonstrate the feasibility of employing FPGA board in games with its inherent advantage.

# 2.0 The Design

## 2.1 Overall Structure

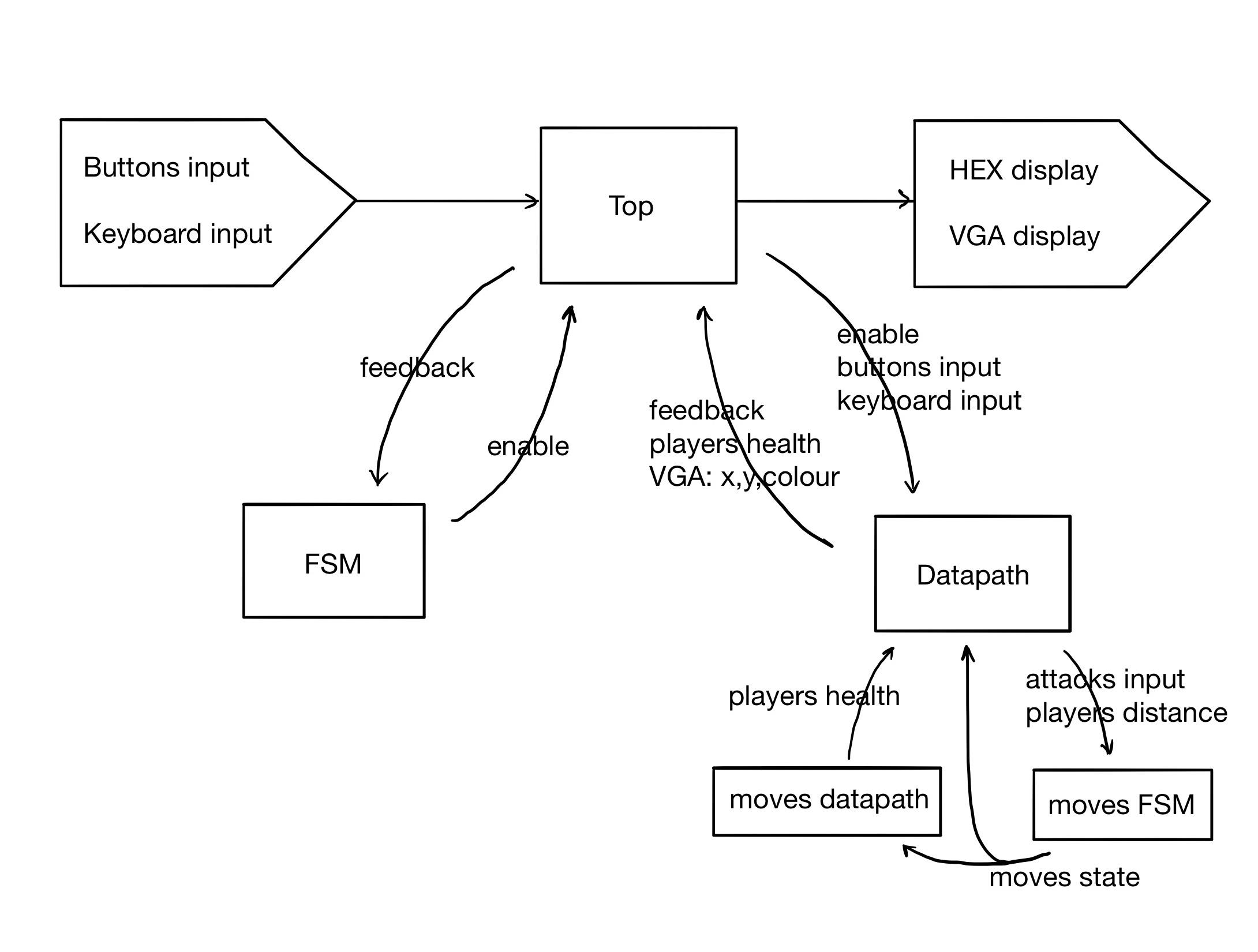


Fig 2.1.0 overall structure block diagram

The game takes four types of inputs. It takes PS2 keyboard as player1 input, game controller buttons as player2 input (Fig 2.1.1), KEY[3] as the “start game” feedback signal and KEY[0] as “reset game” signal.

FSM module stores and updates the current “game state”. That is to say stores ad updates whether the game is at starting screen, or drawing the sprite images, or erasing the sprite images, or waiting for the frame update, or that one character has won the game.

Datapath and related modules implements the gameplay logics, computes and stores the game data. For gameplay logics, by taking the input from the players, it updates the moves states(which represents which moves the character is performing) and deducts health according to the moves states. For game data, it stores and updates the position of both characters and retrieve colour of the sprite image from On-Chip Memory to be drawn on the VGA display.

The game outputs health (in base 10) of both characters to HEX display, and outputs the updated corresponding starting page, sprite image, or winning page to the VGA.

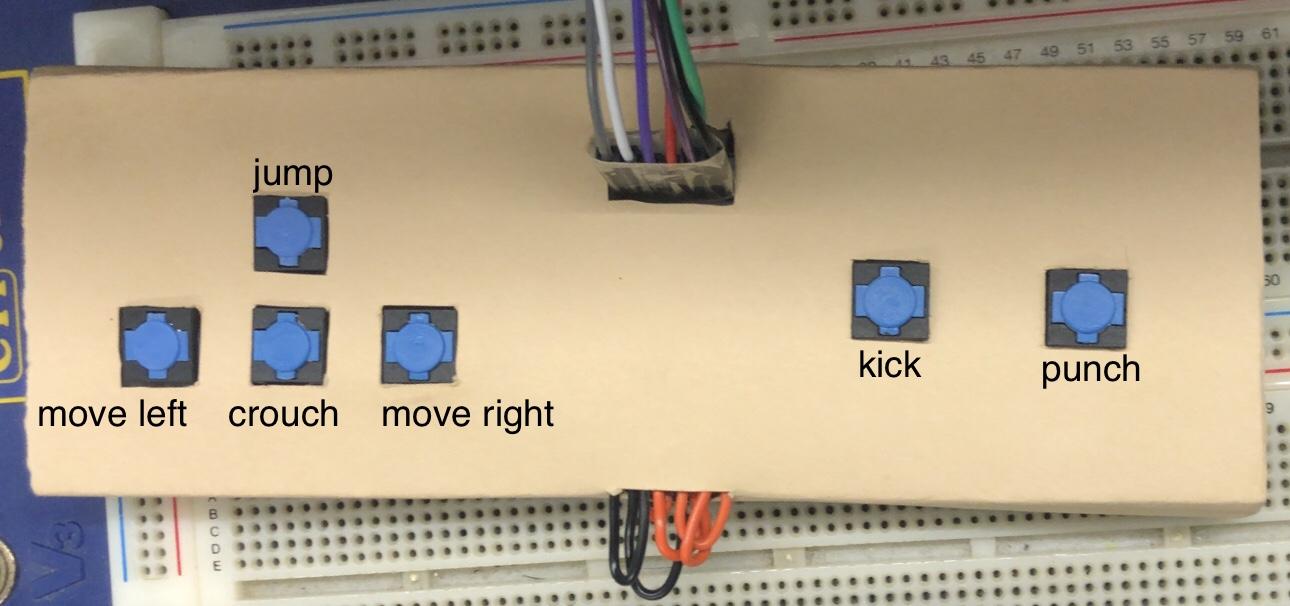


Fig 2.1.1 game controller



Fig 2.1.2 HEX display

## 2.2 Inputs

The keyboard operates in a different clock rate to CLOCK\_50 and keeps sending signals during the time the key is pressed down. Therefore, in the top module file, an keyboard\_FSM\_alt is instantiated to take the original keyboard signal and to send out a one CLOCK\_50 clock cycle of impulse signal per press-and-release. We use six keys for six types of input: A(move left), D(move right), W(fly), S(crouch), J(kick) and L(punch).

Since the keyboard can only sense one key pressing and send one key signal at a time, in order to prevent interference between the inputs of two players, an additional game controller is designed as an input device independent of the keyboard. It has six buttons for six types of input: move left, move right, jump, crouch, kick and punch.

KEY[3] button acts as a feedback signal that changes from initial state to actual gameplay. KEY[0] resets the entire game and also acts as a feedback signal that changes the game back to the initial state.

## 2.3 FSM

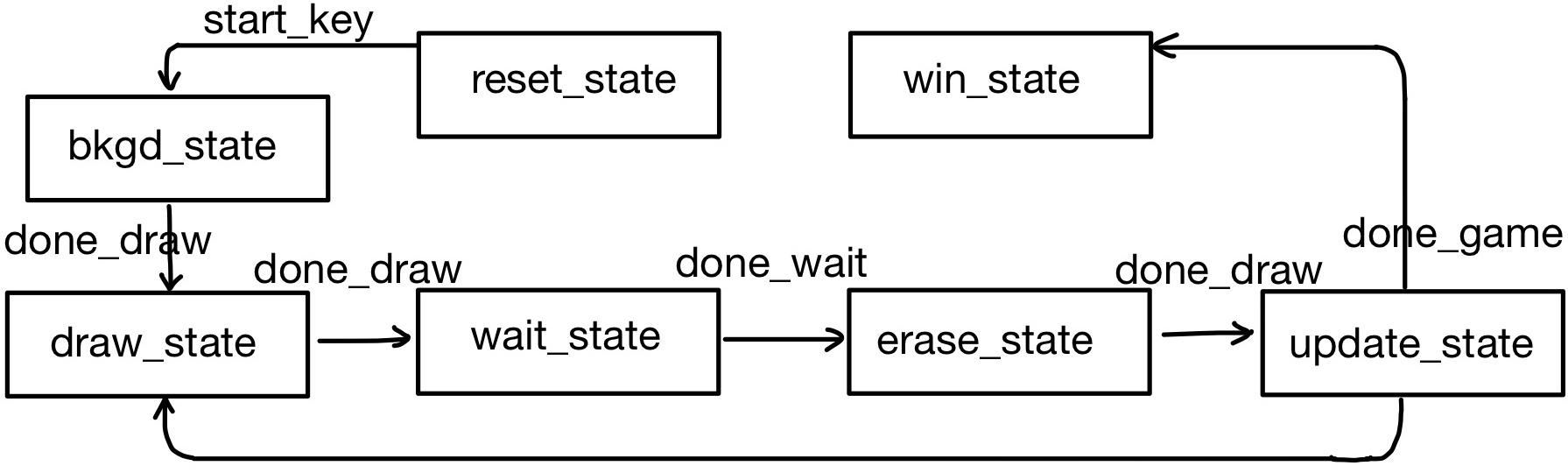


Fig 2.3.0 FSM

FSM module changes the game states between following states. reset\_state: This is the initial state of the game which a starting screen is shown. bkgd\_state: This is an intermediate state between starting screen and the actual gameplay. The background of the game is drawn. draw\_state: The updated sprite image is drawn. wait\_state: This is the state that ensures the data update in a reasonable frame rate that VGA supports. erase\_state: Erases the sprite images. update\_state: It updates the data according to inputs. The game will continue to draw\_state if no winner occurs. win\_state: The game draws corresponding win screen for the winner.

## 2.4 Datapath and Related Modules

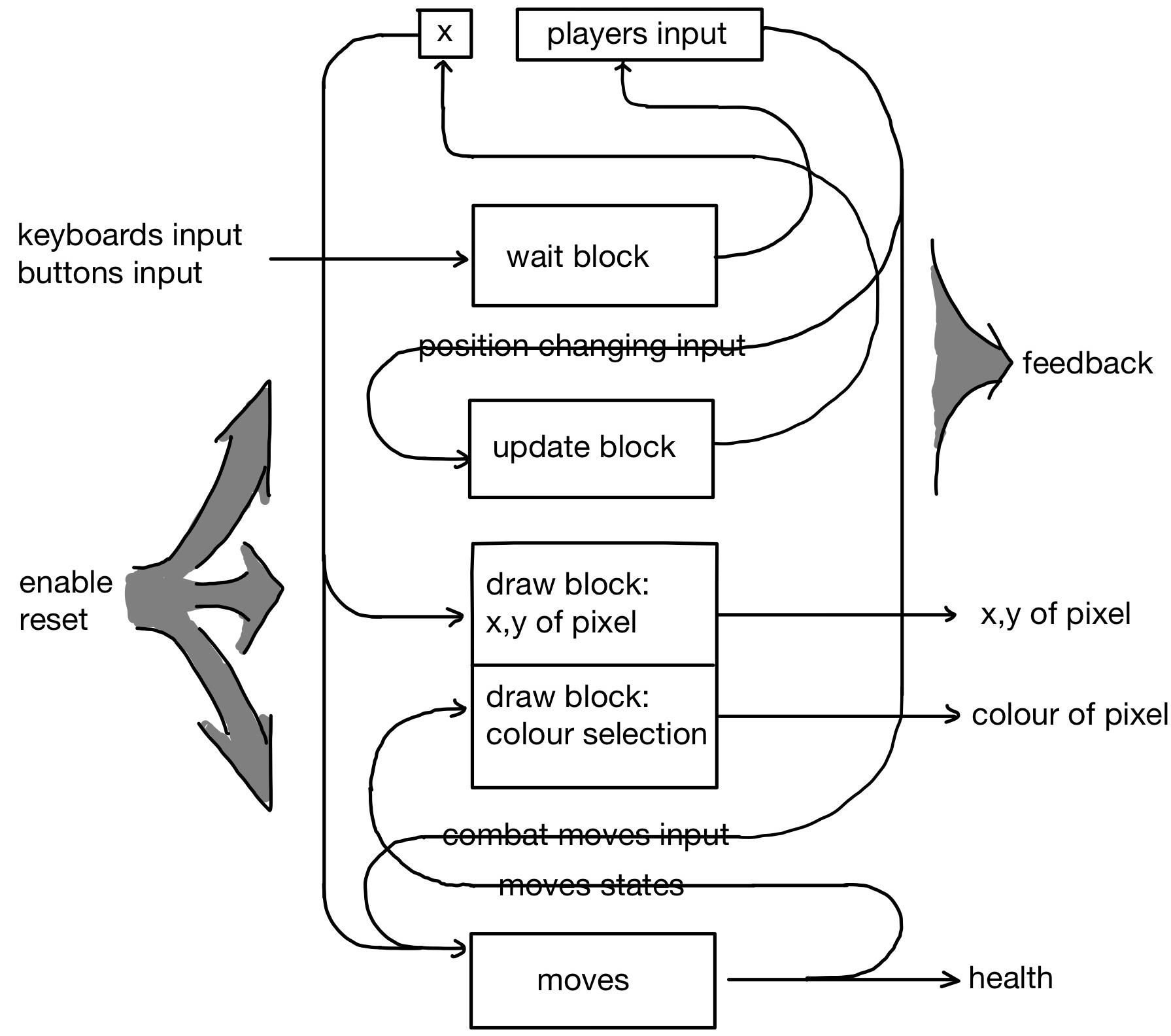


Fig 2.4.0

datapath:

Datapath module takes enable signals and perform tasks accordingly: drawing images pixel by pixel, waiting for a certain time by counting down, updating the horizontal position of sprite.

PS2 keyboard and game controller inputs are one clock cycle impulse signals. They are transformed into a one frame length signal in the wait block and stored as players inputs.

Players inputs are processed separately depending on whether one is a position changing input or combat move input. Position changing input(move left/right) is used to update the horizontal position of both characters in the update block. Combat move input (jump, crouch, kick, punch) are used in moves modules (moves\_FSM, moves\_datapath) to update the moves states and to calculate character health.

Eventually, the datapath module outputs health of both players for HEX display and x,y, colour signals for VGA and feedback signals for FSM.

moves\_FSM:

Moves FSM module changes states of the character corresponding to the input from the player. Two moves FSM is used in order to keep track of the state change of the two players at the same time.

The inputs are clock, resetn, and the current state of the player. The clock signal used for this module is the ‘done\_wait’ signal from ‘datapath’ which is divided into the “frame rate”. The enable signals of kick\_in, punch\_in, jump\_in, crouch\_in are also sent from ‘datapath’ in the rate of “frame rate” which is slower than CLOCK\_50 in order for the VGA to display in a visible way. As a result, the state are changed in “frame rate” as designed. The module includes a state table and a register to update the state. Seven states are designed with five basic moves states and two combo moves states. The game always starts with idle\_state and returned to idle\_state after each move if there is no input immediately after. The two combo state - jumpkicking\_state and sweeping\_state - are entered by having two input signals one after another. Jumpkicking\_state is entered by be in jumping\_state and receives enable ‘kick\_in’. Similarly, sweeping\_state is entered by be in crouching\_state and receives enable ‘kick\_in’. The completed state diagram is shown below (Fig 2.4.1).

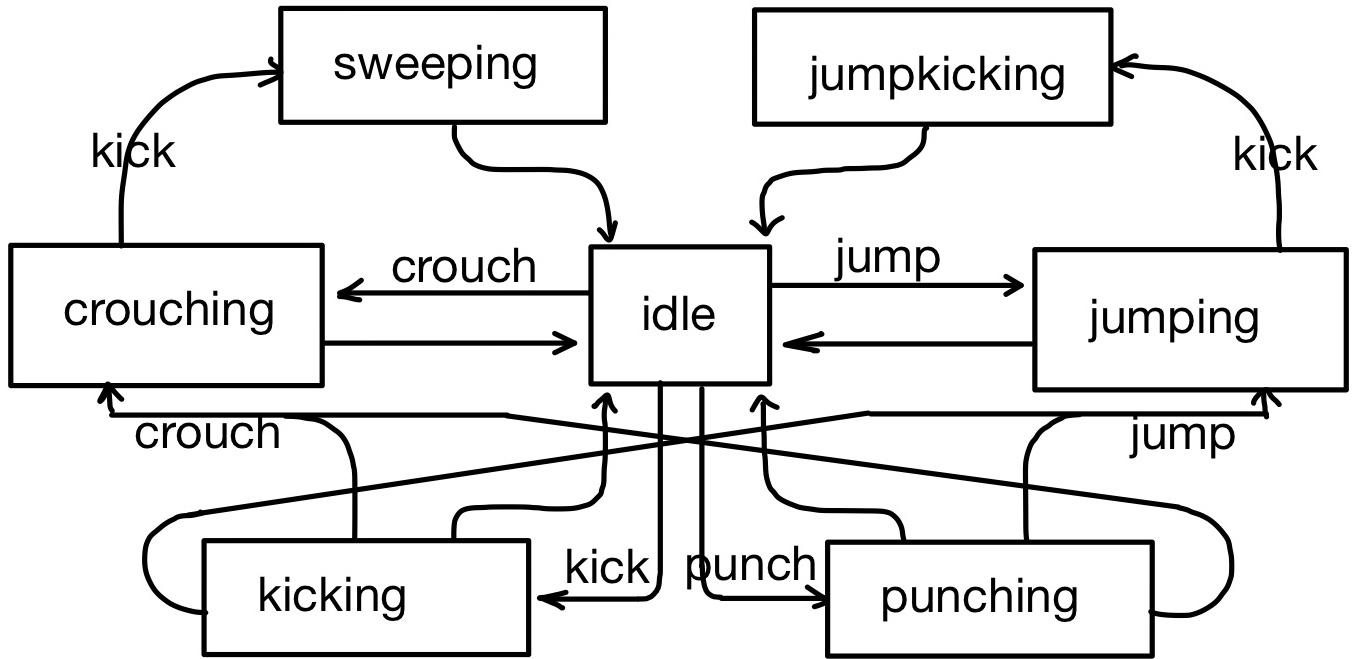


Fig 2.4.1 state diagram for combat moves states

The output of the module is the current\_state of the player which is outputted into moves\_datapath for health calculation and into datapath for selecting the ROM to retrieve the colour of the image to draw.

moves\_datapath:

Moves Datapath module calculates the health points of the two players accordinging to the players’ states. The health points are calculated with a total of 40 points and each player has 20 points to start with. The health points calculation is designed in a way that (player1 health = 40 - player2 health), where the total health does not change.

Players’ states and the distance between the two players are inputted from datapath. Taking into account the distance in between, any distance larger than 0 will not be considered a valid attack. If the distance in between reaches 0, the health point of player1 will be added or subtracted according to player states. The module lists all possibilities of player1 and corresponding player2 states.The game is designed with the following rules:

* punching as -1 points
* kicking as -2 points
* sweeping as -3 points
* jumpkicking as -4 points
* jumping can avoid kicking and sweeping
* crouching can avoid punching and jumpkicking

Since the inputted states are in the rate of “frame rate” so the corresponding health points outputted to HEX display are also in the rate of “frame rate”. Moreover, the output HEX display are converted into decimal numbers for convenience.

# 3.0 Report on Success

Our project has mostly worked as we expected with most of the functionalities implemented. At first, a starting screen is displayed. By pressing KEY[3], players proceeds to actual gameplay. A background with the two characters located at both ends of the screen are displayed. Two players are able to change horizontal position and perform combat moves at their positions. Certain combinations of positions and combat moves result in change of health of the characters. Corresponding sprite images are displayed and health points are updated and displayed on HEX accordingly. At the end of the game, the monitor displays the winning screen with one player reaching the maximum of 40 health points. The game can be reseted to starting screen by pressing KEY[0].



Fig 3.0.0 starting screen, gameplay, player2 winning screen

However, the two characters are not able to have “physical contact” during the fight which is a result of not having able to allow transparency with the mif converter provided. In order to come over with the problem, the sprite images are designed in a way that shows a visible energy waves which matches with the sci-fi theme and compensate the aforementioned restriction.

Furthermore, we omitted sound output as we progressed on the project. Since the gameplay logic proved to be more time-consuming than originally expected.

# 4.0 Potential for Improvement

The planning for the project in the beginning has been very effective and helpful. However, due to the lack of experience on fully self designed project, too much modules are implemented at once which extends the debugging process to be very time consuming. Since it is much more difficult to debug with multiple possible locations of mistakes and one could affect another. As a way to improve, each module and the sub parts of the module should be implemented step by step with effective checking by simulation correspondingly. The process of the design and debugging will be much more smooth and time efficient.

# 5.0 References

[1] D. Black, “Intel Extends FPGA Ecosystem with 10nm Agilex,” *HPCwire*, 12-Apr-2019. [Online]. Available: https://www.hpcwire.com/2019/04/11/intel-extends-fpga-ecosystem-with-10nm-agilex/. [Accessed: 02-Dec-2019].

[2] A. van der Ploeg, “Why use an FPGA instead of a CPU or GPU?,” *Medium*, 14-Aug-2018. [Online]. Available: https://blog.esciencecenter.nl/why-use-an-fpga-instead-of-a-cpu-or-gpu-b234cd4f309c. [Accessed: 02-Dec-2019].

[3] “Advanced game programming,” *Department of Computing*. [Online]. Available: https://www.doc.gold.ac.uk/~mas01at/lecture\_notes/agp/agp\_fpgas\_in\_games.html. [Accessed: 02-Dec-2019].

# Appendix

module top

(

CLOCK\_50, // On Board 50 MHz

// Your inputs and outputs here

SW,

KEY, // On Board Keys

//VGA output

VGA\_CLK, // VGA Clock

VGA\_HS, // VGA H\_SYNC

VGA\_VS, // VGA V\_SYNC

VGA\_BLANK\_N, // VGA BLANK

VGA\_SYNC\_N, // VGA SYNC

VGA\_R, // VGA Red[9:0]

VGA\_G, // VGA Green[9:0]

VGA\_B, // VGA Blue[9:0]

//p1 input

PS2\_CLK,

PS2\_DAT,

//p2 input

GPIO\_0,

//output

HEX0,

HEX1,

HEX2,

HEX3,

HEX4,

HEX5

);

input CLOCK\_50; // 50 MHz

input [3:0] KEY;

// Declare your inputs and outputs here

input [9:0] SW;

// Do not change the following outputs

output VGA\_CLK; // VGA Clock

output VGA\_HS; // VGA H\_SYNC

output VGA\_VS; // VGA V\_SYNC

output VGA\_BLANK\_N; // VGA BLANK

output VGA\_SYNC\_N; // VGA SYNC

output [7:0] VGA\_R; // VGA Red[7:0] Changed from 10 to 8-bit DAC

output [7:0] VGA\_G; // VGA Green[7:0]

output [7:0] VGA\_B; // VGA Blue[7:0]

inout PS2\_CLK,

PS2\_DAT;

input [35:0] GPIO\_0;

output [6:0] HEX0,

HEX1,

HEX2,

HEX3,

HEX4,

HEX5;

wire resetn;

assign resetn = KEY[0];

// Create the colour, x, y and writeEn wires that are inputs to the controller.

wire [8:0] colour;

wire [7:0] x;

wire [6:0] y;

wire writeEn;

assignments a1(CLOCK\_50, KEY[0], colour, x, y, writeEn,

~KEY[3],

//refer to this when make connection

GPIO\_0[10], GPIO\_0[12], GPIO\_0[14], GPIO\_0[16], GPIO\_0[18], GPIO\_0[20],

PS2\_CLK, PS2\_DAT,

HEX0,

HEX1,

HEX2,

HEX3,

HEX4,

HEX5);

// Create an Instance of a VGA controller - there can be only one!

// Define the number of colours as well as the initial background

// image file (.MIF) for the controller.

vga\_adapter VGA(

.resetn(resetn),

.clock(CLOCK\_50),

.colour(colour),

.x(x),

.y(y),

.plot(writeEn),

/\* Signals for the DAC to drive the monitor. \*/

.VGA\_R(VGA\_R),

.VGA\_G(VGA\_G),

.VGA\_B(VGA\_B),

.VGA\_HS(VGA\_HS),

.VGA\_VS(VGA\_VS),

.VGA\_BLANK(VGA\_BLANK\_N),

.VGA\_SYNC(VGA\_SYNC\_N),

.VGA\_CLK(VGA\_CLK));

defparam VGA.RESOLUTION = "160x120";

defparam VGA.MONOCHROME = "FALSE";

defparam VGA.BITS\_PER\_COLOUR\_CHANNEL = 3;

defparam VGA.BACKGROUND\_IMAGE = "start\_screen.mif";

// Put your code here. Your code should produce signals x,y,colour and writeEn

// for the VGA controller, in addition to any other functionality your design may require.

Endmodule

module assignments (clk, resetn, colour\_out, x, y, writeEn,

start\_key,

KICK\_BUTTON, PUNCH\_BUTTON, JUMP\_BUTTON, CROUCH\_BUTTON, LEFT\_BUTTON, RIGHT\_BUTTON,

PS2\_CLK,PS2\_DAT,

HEX0,

HEX1,

HEX2,

HEX3,

HEX4,

HEX5);

input clk;

input resetn;

output [8:0]colour\_out;

output [7:0]x;

output [6:0]y;

output writeEn;

input start\_key;

input KICK\_BUTTON, PUNCH\_BUTTON, JUMP\_BUTTON, CROUCH\_BUTTON, LEFT\_BUTTON, RIGHT\_BUTTON;

inout PS2\_CLK,

PS2\_DAT;

output [6:0]HEX0,

HEX1,

HEX2,

HEX3,

HEX4,

HEX5;

//assign HEX0 = 7'h7f;

//assign HEX1 = 7'h7f;

//assign HEX2 = 7'h7f;

//assign HEX3 = 7'h7f;

assign HEX4 = 7'h7f;

assign HEX5 = 7'h7f;

wire done\_draw, done\_wait, done\_game, go\_reset, go\_bkgd, go\_draw, go\_wait, go\_update, go\_erase, go\_win;

wire [7:0]key\_pressed\_data;

PS2\_Demo p0(

// Inputs

.clk(clk),

.reset(resetn),

// Bidirectionals

.PS2\_CLK(PS2\_CLK),

.PS2\_DAT(PS2\_DAT),

// Outputs

.HEX0(),

.HEX1(),

.HEX2(),

.HEX3(),

.HEX4(),

.HEX5(),

.HEX6(),

.HEX7(),

.last\_data\_received(key\_pressed\_data));

localparam kick\_data = 3'b000, punch\_data = 3'b001, jump\_data = 3'b010, crouch\_data = 3'b011, left\_data = 3'b100, right\_data = 3'b101;

reg [2:0] button\_pressed\_data=3'b111;

always@(\*)

begin

if(!KICK\_BUTTON) button\_pressed\_data = kick\_data;

else if(!PUNCH\_BUTTON) button\_pressed\_data = punch\_data;

else if(!JUMP\_BUTTON) button\_pressed\_data = jump\_data;

else if(!CROUCH\_BUTTON) button\_pressed\_data = crouch\_data;

else if(!LEFT\_BUTTON) button\_pressed\_data = left\_data;

else if(!RIGHT\_BUTTON) button\_pressed\_data = right\_data;

else button\_pressed\_data = 3'b111;

end

//implement start\_key!!!

FSM s1 (clk, resetn, start\_key, done\_draw, done\_wait, done\_game, go\_reset, go\_bkgd, go\_draw, go\_wait, go\_update, go\_erase, go\_win);

datapath d1 (HEX3,HEX2,HEX1,HEX0, clk, resetn, key\_pressed\_data, button\_pressed\_data, colour\_out, go\_reset, go\_bkgd, go\_draw, go\_wait, go\_update, go\_erase, go\_win, done\_draw, done\_wait, done\_game, x, y);

assign writeEn = go\_draw;

endmodule

module PS2\_Demo (

// Inputs

clk,

reset,

// Bidirectionals

PS2\_CLK,

PS2\_DAT,

// Outputs

HEX0,

HEX1,

HEX2,

HEX3,

HEX4,

HEX5,

HEX6,

HEX7,

last\_data\_received

);

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

\* Parameter Declarations \*

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

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

\* Port Declarations \*

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

// Inputs

input clk;

input reset;

// Bidirectionals

inout PS2\_CLK;

inout PS2\_DAT;

// Outputs

output [6:0] HEX0;

output [6:0] HEX1;

output [6:0] HEX2;

output [6:0] HEX3;

output [6:0] HEX4;

output [6:0] HEX5;

output [6:0] HEX6;

output [6:0] HEX7;

output reg [7:0] last\_data\_received;

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

\* Internal Wires and Registers Declarations \*

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

// Internal Wires

wire [7:0] ps2\_key\_data;

wire ps2\_key\_pressed;

// State Machine Registers

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

\* Finite State Machine(s) \*

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

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

\* Sequential Logic \*

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

/\*

wire [7:0] last\_data\_received\_temp;

keyboard\_FSM s1(HEX0, HEX1, clk,reset,ps2\_key\_pressed,ps2\_key\_data,last\_data\_received\_temp);

always @(posedge clk)

begin

if (reset == 1'b0)

last\_data\_received <= 8'h00;

else

last\_data\_received <= last\_data\_received\_temp;

end

\*/

wire enable;

keyboard\_FSM\_alt s1(HEX0, clk,reset,ps2\_key\_pressed,ps2\_key\_data, enable);

always @(posedge clk)

begin

if (reset == 1'b0)

last\_data\_received <= 8'h00;

else

last\_data\_received <= (enable?ps2\_key\_data:8'h00);

end

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

\* Combinational Logic \*

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

//assign HEX2 = 7'h7F;

//assign HEX3 = 7'h7F;

//assign HEX4 = 7'h7F;

//assign HEX5 = 7'h7F;

//assign HEX6 = 7'h7F;

//assign HEX7 = 7'h7F;

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

\* Internal Modules \*

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

PS2\_Controller PS2 (

// Inputs

.CLOCK\_50 (clk),

.reset (~reset),

// Bidirectionals

.PS2\_CLK (PS2\_CLK),

.PS2\_DAT (PS2\_DAT),

// Outputs

.received\_data (ps2\_key\_data),

.received\_data\_en (ps2\_key\_pressed)

);

endmodule

module keyboard\_FSM\_alt(HEX0, clk,reset,ps2\_key\_pressed,ps2\_key\_data,enable);

output [6:0] HEX0;//newly

input clk;

input reset;

input ps2\_key\_pressed;

input [7:0] ps2\_key\_data;

output reg enable;

reg done\_received;

reg [4:0] current\_state, next\_state;

localparam wait\_state=2'b00, pre\_state=2'b01, start\_state=2'b10;

//state change according to input

reg [27:0] count;

always@(\*)

begin

case(current\_state)//how bout

wait\_state: next\_state=(ps2\_key\_data==8'hf0)?pre\_state:wait\_state;

pre\_state: next\_state=(ps2\_key\_data==8'hf0)?pre\_state:start\_state;

start\_state: next\_state=wait\_state;

default: next\_state=wait\_state;

endcase

end

//enable signals

always@(\*)

begin

case(current\_state)

wait\_state: enable=0;

pre\_state: enable=0;

start\_state: enable=1;

default: enable=0;

endcase

end

//state change

always @(posedge clk)

begin

if (reset == 0)

current\_state <= wait\_state;

else

current\_state <= next\_state;

end

Hexadecimal\_To\_Seven\_Segment Segment0 (

// Inputs

.hex\_number (current\_state[3:0]),

// Bidirectional

// Outputs

.seven\_seg\_display (HEX0)

);

endmodule

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Module: Altera\_UP\_PS2 \*

\* Description: \*

\* This module communicates with the PS2 core. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

module PS2\_Controller #(parameter INITIALIZE\_MOUSE = 0) (

// Inputs

CLOCK\_50,

reset,

the\_command,

send\_command,

// Bidirectionals

PS2\_CLK, // PS2 Clock

PS2\_DAT, // PS2 Data

// Outputs

command\_was\_sent,

error\_communication\_timed\_out,

received\_data,

received\_data\_en // If 1 - new data has been received

);

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Parameter Declarations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Port Declarations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

// Inputs

input CLOCK\_50;

input reset;

input [7:0] the\_command;

input send\_command;

// Bidirectionals

inout PS2\_CLK;

inout PS2\_DAT;

// Outputs

output command\_was\_sent;

output error\_communication\_timed\_out;

output [7:0] received\_data;

output received\_data\_en;

wire [7:0] the\_command\_w;

wire send\_command\_w, command\_was\_sent\_w, error\_communication\_timed\_out\_w;

generate

if(INITIALIZE\_MOUSE) begin

assign the\_command\_w = init\_done ? the\_command : 8'hf4;

assign send\_command\_w = init\_done ? send\_command : (!command\_was\_sent\_w && !error\_communication\_timed\_out\_w);

assign command\_was\_sent = init\_done ? command\_was\_sent\_w : 0;

assign error\_communication\_timed\_out = init\_done ? error\_communication\_timed\_out\_w : 1;

reg init\_done;

always @(posedge CLOCK\_50)

if(reset) init\_done <= 0;

else if(command\_was\_sent\_w) init\_done <= 1;

end else begin

assign the\_command\_w = the\_command;

assign send\_command\_w = send\_command;

assign command\_was\_sent = command\_was\_sent\_w;

assign error\_communication\_timed\_out = error\_communication\_timed\_out\_w;

end

endgenerate

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Constant Declarations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

// states

localparam PS2\_STATE\_0\_IDLE = 3'h0,

PS2\_STATE\_1\_DATA\_IN = 3'h1,

PS2\_STATE\_2\_COMMAND\_OUT = 3'h2,

PS2\_STATE\_3\_END\_TRANSFER = 3'h3,

PS2\_STATE\_4\_END\_DELAYED = 3'h4;

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Internal wires and registers Declarations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

// Internal Wires

wire ps2\_clk\_posedge;

wire ps2\_clk\_negedge;

wire start\_receiving\_data;

wire wait\_for\_incoming\_data;

// Internal Registers

reg [7:0] idle\_counter;

reg ps2\_clk\_reg;

reg ps2\_data\_reg;

reg last\_ps2\_clk;

// State Machine Registers

reg [2:0] ns\_ps2\_transceiver;

reg [2:0] s\_ps2\_transceiver;

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Finite State Machine(s) \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

always @(posedge CLOCK\_50)

begin

if (reset == 1'b1)

s\_ps2\_transceiver <= PS2\_STATE\_0\_IDLE;

else

s\_ps2\_transceiver <= ns\_ps2\_transceiver;

end

always @(\*)

begin

// Defaults

ns\_ps2\_transceiver = PS2\_STATE\_0\_IDLE;

case (s\_ps2\_transceiver)

PS2\_STATE\_0\_IDLE:

begin

if ((idle\_counter == 8'hFF) &&

(send\_command == 1'b1))

ns\_ps2\_transceiver = PS2\_STATE\_2\_COMMAND\_OUT;

else if ((ps2\_data\_reg == 1'b0) && (ps2\_clk\_posedge == 1'b1))

ns\_ps2\_transceiver = PS2\_STATE\_1\_DATA\_IN;

else

ns\_ps2\_transceiver = PS2\_STATE\_0\_IDLE;

end

PS2\_STATE\_1\_DATA\_IN:

begin

if ((received\_data\_en == 1'b1)/-\* && (ps2\_clk\_posedge == 1'b1)\*-/)

ns\_ps2\_transceiver = PS2\_STATE\_0\_IDLE;

else

ns\_ps2\_transceiver = PS2\_STATE\_1\_DATA\_IN;

end

PS2\_STATE\_2\_COMMAND\_OUT:

begin

if ((command\_was\_sent == 1'b1) ||

(error\_communication\_timed\_out == 1'b1))

ns\_ps2\_transceiver = PS2\_STATE\_3\_END\_TRANSFER;

else

ns\_ps2\_transceiver = PS2\_STATE\_2\_COMMAND\_OUT;

end

PS2\_STATE\_3\_END\_TRANSFER:

begin

if (send\_command == 1'b0)

ns\_ps2\_transceiver = PS2\_STATE\_0\_IDLE;

else if ((ps2\_data\_reg == 1'b0) && (ps2\_clk\_posedge == 1'b1))

ns\_ps2\_transceiver = PS2\_STATE\_4\_END\_DELAYED;

else

ns\_ps2\_transceiver = PS2\_STATE\_3\_END\_TRANSFER;

end

PS2\_STATE\_4\_END\_DELAYED:

begin

if (received\_data\_en == 1'b1)

begin

if (send\_command == 1'b0)

ns\_ps2\_transceiver = PS2\_STATE\_0\_IDLE;

else

ns\_ps2\_transceiver = PS2\_STATE\_3\_END\_TRANSFER;

end

else

ns\_ps2\_transceiver = PS2\_STATE\_4\_END\_DELAYED;

end

default:

ns\_ps2\_transceiver = PS2\_STATE\_0\_IDLE;

endcase

end

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Sequential logic \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

always @(posedge CLOCK\_50)

begin

if (reset == 1'b1)

begin

last\_ps2\_clk <= 1'b1;

ps2\_clk\_reg <= 1'b1;

ps2\_data\_reg <= 1'b1;

end

else

begin

last\_ps2\_clk <= ps2\_clk\_reg;

ps2\_clk\_reg <= PS2\_CLK;

ps2\_data\_reg <= PS2\_DAT;

end

end

always @(posedge CLOCK\_50)

begin

if (reset == 1'b1)

idle\_counter <= 6'h00;

else if ((s\_ps2\_transceiver == PS2\_STATE\_0\_IDLE) &&

(idle\_counter != 8'hFF))

idle\_counter <= idle\_counter + 6'h01;

else if (s\_ps2\_transceiver != PS2\_STATE\_0\_IDLE)

idle\_counter <= 6'h00;

end

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Combinational logic \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

assign ps2\_clk\_posedge =

((ps2\_clk\_reg == 1'b1) && (last\_ps2\_clk == 1'b0)) ? 1'b1 : 1'b0;

assign ps2\_clk\_negedge =

((ps2\_clk\_reg == 1'b0) && (last\_ps2\_clk == 1'b1)) ? 1'b1 : 1'b0;

assign start\_receiving\_data = (s\_ps2\_transceiver == PS2\_STATE\_1\_DATA\_IN);

assign wait\_for\_incoming\_data =

(s\_ps2\_transceiver == PS2\_STATE\_3\_END\_TRANSFER);

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Internal Modules \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

Altera\_UP\_PS2\_Data\_In PS2\_Data\_In (

// Inputs

.clk (CLOCK\_50),

.reset (reset),

.wait\_for\_incoming\_data (wait\_for\_incoming\_data),

.start\_receiving\_data (start\_receiving\_data),

.ps2\_clk\_posedge (ps2\_clk\_posedge),

.ps2\_clk\_negedge (ps2\_clk\_negedge),

.ps2\_data (ps2\_data\_reg),

// Bidirectionals

// Outputs

.received\_data (received\_data),

.received\_data\_en (received\_data\_en)

);

Altera\_UP\_PS2\_Command\_Out PS2\_Command\_Out (

// Inputs

.clk (CLOCK\_50),

.reset (reset),

.the\_command (the\_command\_w),

.send\_command (send\_command\_w),

.ps2\_clk\_posedge (ps2\_clk\_posedge),

.ps2\_clk\_negedge (ps2\_clk\_negedge),

// Bidirectionals

.PS2\_CLK (PS2\_CLK),

.PS2\_DAT (PS2\_DAT),

// Outputs

.command\_was\_sent (command\_was\_sent\_w),

.error\_communication\_timed\_out (error\_communication\_timed\_out\_w)

);

endmodule

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

\* \*

\* Module: Altera\_UP\_PS2\_Data\_In \*

\* Description: \*

\* This module accepts incoming data from a PS2 core. \*

\* \*

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

module Altera\_UP\_PS2\_Data\_In (

// Inputs

clk,

reset,

wait\_for\_incoming\_data,

start\_receiving\_data,

ps2\_clk\_posedge,

ps2\_clk\_negedge,

ps2\_data,

// Bidirectionals

// Outputs

received\_data,

received\_data\_en // If 1 - new data has been received

);

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

\* Parameter Declarations \*

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

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

\* Port Declarations \*

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

// Inputs

input clk;

input reset;

input wait\_for\_incoming\_data;

input start\_receiving\_data;

input ps2\_clk\_posedge;

input ps2\_clk\_negedge;

input ps2\_data;

// Bidirectionals

// Outputs

output reg [7:0] received\_data;

output reg received\_data\_en;

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

\* Constant Declarations \*

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

// states

localparam PS2\_STATE\_0\_IDLE = 3'h0,

PS2\_STATE\_1\_WAIT\_FOR\_DATA = 3'h1,

PS2\_STATE\_2\_DATA\_IN = 3'h2,

PS2\_STATE\_3\_PARITY\_IN = 3'h3,

PS2\_STATE\_4\_STOP\_IN = 3'h4;

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

\* Internal wires and registers Declarations \*

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

// Internal Wires

reg [3:0] data\_count;

reg [7:0] data\_shift\_reg;

// State Machine Registers

reg [2:0] ns\_ps2\_receiver;

reg [2:0] s\_ps2\_receiver;

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

\* Finite State Machine(s) \*

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

always @(posedge clk)

begin

if (reset == 1'b1)

s\_ps2\_receiver <= PS2\_STATE\_0\_IDLE;

else

s\_ps2\_receiver <= ns\_ps2\_receiver;

end

always @(\*)

begin

// Defaults

ns\_ps2\_receiver = PS2\_STATE\_0\_IDLE;

case (s\_ps2\_receiver)

PS2\_STATE\_0\_IDLE:

begin

if ((wait\_for\_incoming\_data == 1'b1) &&

(received\_data\_en == 1'b0))

ns\_ps2\_receiver = PS2\_STATE\_1\_WAIT\_FOR\_DATA;

else if ((start\_receiving\_data == 1'b1) &&

(received\_data\_en == 1'b0))

ns\_ps2\_receiver = PS2\_STATE\_2\_DATA\_IN;

else

ns\_ps2\_receiver = PS2\_STATE\_0\_IDLE;

end

PS2\_STATE\_1\_WAIT\_FOR\_DATA:

begin

if ((ps2\_data == 1'b0) && (ps2\_clk\_posedge == 1'b1))

ns\_ps2\_receiver = PS2\_STATE\_2\_DATA\_IN;

else if (wait\_for\_incoming\_data == 1'b0)

ns\_ps2\_receiver = PS2\_STATE\_0\_IDLE;

else

ns\_ps2\_receiver = PS2\_STATE\_1\_WAIT\_FOR\_DATA;

end

PS2\_STATE\_2\_DATA\_IN:

begin

if ((data\_count == 3'h7) && (ps2\_clk\_posedge == 1'b1))

ns\_ps2\_receiver = PS2\_STATE\_3\_PARITY\_IN;

else

ns\_ps2\_receiver = PS2\_STATE\_2\_DATA\_IN;

end

PS2\_STATE\_3\_PARITY\_IN:

begin

if (ps2\_clk\_posedge == 1'b1)

ns\_ps2\_receiver = PS2\_STATE\_4\_STOP\_IN;

else

ns\_ps2\_receiver = PS2\_STATE\_3\_PARITY\_IN;

end

PS2\_STATE\_4\_STOP\_IN:

begin

if (ps2\_clk\_posedge == 1'b1)

ns\_ps2\_receiver = PS2\_STATE\_0\_IDLE;

else

ns\_ps2\_receiver = PS2\_STATE\_4\_STOP\_IN;

end

default:

begin

ns\_ps2\_receiver = PS2\_STATE\_0\_IDLE;

end

endcase

end

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

\* Sequential logic \*

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

always @(posedge clk)

begin

if (reset == 1'b1)

data\_count <= 3'h0;

else if ((s\_ps2\_receiver == PS2\_STATE\_2\_DATA\_IN) &&

(ps2\_clk\_posedge == 1'b1))

data\_count <= data\_count + 3'h1;

else if (s\_ps2\_receiver != PS2\_STATE\_2\_DATA\_IN)

data\_count <= 3'h0;

end

always @(posedge clk)

begin

if (reset == 1'b1)

data\_shift\_reg <= 8'h00;

else if ((s\_ps2\_receiver == PS2\_STATE\_2\_DATA\_IN) &&

(ps2\_clk\_posedge == 1'b1))

data\_shift\_reg <= {ps2\_data, data\_shift\_reg[7:1]};

end

always @(posedge clk)

begin

if (reset == 1'b1)

received\_data <= 8'h00;

else if (s\_ps2\_receiver == PS2\_STATE\_4\_STOP\_IN)

received\_data <= data\_shift\_reg;

end

always @(posedge clk)

begin

if (reset == 1'b1)

received\_data\_en <= 1'b0;

else if ((s\_ps2\_receiver == PS2\_STATE\_4\_STOP\_IN) &&

(ps2\_clk\_posedge == 1'b1))

received\_data\_en <= 1'b1;

else

received\_data\_en <= 1'b0;

end

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

\* Combinational logic \*

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

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

\* Internal Modules \*

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

endmodule

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

\* \*

\* Module: Altera\_UP\_PS2\_Command\_Out \*

\* Description: \*

\* This module sends commands out to the PS2 core. \*

\* \*

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

module Altera\_UP\_PS2\_Command\_Out (

// Inputs

clk,

reset,

the\_command,

send\_command,

ps2\_clk\_posedge,

ps2\_clk\_negedge,

// Bidirectionals

PS2\_CLK,

PS2\_DAT,

// Outputs

command\_was\_sent,

error\_communication\_timed\_out

);

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

\* Parameter Declarations \*

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

// Timing info for initiating Host-to-Device communication

// when using a 50MHz system clock

parameter CLOCK\_CYCLES\_FOR\_101US = 5050;

parameter NUMBER\_OF\_BITS\_FOR\_101US = 13;

parameter COUNTER\_INCREMENT\_FOR\_101US = 13'h0001;

//parameter CLOCK\_CYCLES\_FOR\_101US = 50;

//parameter NUMBER\_OF\_BITS\_FOR\_101US = 6;

//parameter COUNTER\_INCREMENT\_FOR\_101US = 6'h01;

// Timing info for start of transmission error

// when using a 50MHz system clock

parameter CLOCK\_CYCLES\_FOR\_15MS = 750000;

parameter NUMBER\_OF\_BITS\_FOR\_15MS = 20;

parameter COUNTER\_INCREMENT\_FOR\_15MS = 20'h00001;

// Timing info for sending data error

// when using a 50MHz system clock

parameter CLOCK\_CYCLES\_FOR\_2MS = 100000;

parameter NUMBER\_OF\_BITS\_FOR\_2MS = 17;

parameter COUNTER\_INCREMENT\_FOR\_2MS = 17'h00001;

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

\* Port Declarations \*

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

// Inputs

input clk;

input reset;

input [7:0] the\_command;

input send\_command;

input ps2\_clk\_posedge;

input ps2\_clk\_negedge;

// Bidirectionals

inout PS2\_CLK;

inout PS2\_DAT;

// Outputs

output reg command\_was\_sent;

output reg error\_communication\_timed\_out;

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

\* Constant Declarations \*

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

// states

parameter PS2\_STATE\_0\_IDLE = 3'h0,

PS2\_STATE\_1\_INITIATE\_COMMUNICATION = 3'h1,

PS2\_STATE\_2\_WAIT\_FOR\_CLOCK = 3'h2,

PS2\_STATE\_3\_TRANSMIT\_DATA = 3'h3,

PS2\_STATE\_4\_TRANSMIT\_STOP\_BIT = 3'h4,

PS2\_STATE\_5\_RECEIVE\_ACK\_BIT = 3'h5,

PS2\_STATE\_6\_COMMAND\_WAS\_SENT = 3'h6,

PS2\_STATE\_7\_TRANSMISSION\_ERROR = 3'h7;

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

\* Internal wires and registers Declarations \*

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

// Internal Wires

// Internal Registers

reg [3:0] cur\_bit;

reg [8:0] ps2\_command;

reg [NUMBER\_OF\_BITS\_FOR\_101US:1] command\_initiate\_counter;

reg [NUMBER\_OF\_BITS\_FOR\_15MS:1] waiting\_counter;

reg [NUMBER\_OF\_BITS\_FOR\_2MS:1] transfer\_counter;

// State Machine Registers

reg [2:0] ns\_ps2\_transmitter;

reg [2:0] s\_ps2\_transmitter;

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

\* Finite State Machine(s) \*

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

always @(posedge clk)

begin

if (reset == 1'b1)

s\_ps2\_transmitter <= PS2\_STATE\_0\_IDLE;

else

s\_ps2\_transmitter <= ns\_ps2\_transmitter;

end

always @(\*)

begin

// Defaults

ns\_ps2\_transmitter = PS2\_STATE\_0\_IDLE;

case (s\_ps2\_transmitter)

PS2\_STATE\_0\_IDLE:

begin

if (send\_command == 1'b1)

ns\_ps2\_transmitter = PS2\_STATE\_1\_INITIATE\_COMMUNICATION;

else

ns\_ps2\_transmitter = PS2\_STATE\_0\_IDLE;

end

PS2\_STATE\_1\_INITIATE\_COMMUNICATION:

begin

if (command\_initiate\_counter == CLOCK\_CYCLES\_FOR\_101US)

ns\_ps2\_transmitter = PS2\_STATE\_2\_WAIT\_FOR\_CLOCK;

else

ns\_ps2\_transmitter = PS2\_STATE\_1\_INITIATE\_COMMUNICATION;

end

PS2\_STATE\_2\_WAIT\_FOR\_CLOCK:

begin

if (ps2\_clk\_negedge == 1'b1)

ns\_ps2\_transmitter = PS2\_STATE\_3\_TRANSMIT\_DATA;

else if (waiting\_counter == CLOCK\_CYCLES\_FOR\_15MS)

ns\_ps2\_transmitter = PS2\_STATE\_7\_TRANSMISSION\_ERROR;

else

ns\_ps2\_transmitter = PS2\_STATE\_2\_WAIT\_FOR\_CLOCK;

end

PS2\_STATE\_3\_TRANSMIT\_DATA:

begin

if ((cur\_bit == 4'd8) && (ps2\_clk\_negedge == 1'b1))

ns\_ps2\_transmitter = PS2\_STATE\_4\_TRANSMIT\_STOP\_BIT;

else if (transfer\_counter == CLOCK\_CYCLES\_FOR\_2MS)

ns\_ps2\_transmitter = PS2\_STATE\_7\_TRANSMISSION\_ERROR;

else

ns\_ps2\_transmitter = PS2\_STATE\_3\_TRANSMIT\_DATA;

end

PS2\_STATE\_4\_TRANSMIT\_STOP\_BIT:

begin

if (ps2\_clk\_negedge == 1'b1)

ns\_ps2\_transmitter = PS2\_STATE\_5\_RECEIVE\_ACK\_BIT;

else if (transfer\_counter == CLOCK\_CYCLES\_FOR\_2MS)

ns\_ps2\_transmitter = PS2\_STATE\_7\_TRANSMISSION\_ERROR;

else

ns\_ps2\_transmitter = PS2\_STATE\_4\_TRANSMIT\_STOP\_BIT;

end

PS2\_STATE\_5\_RECEIVE\_ACK\_BIT:

begin

if (ps2\_clk\_posedge == 1'b1)

ns\_ps2\_transmitter = PS2\_STATE\_6\_COMMAND\_WAS\_SENT;

else if (transfer\_counter == CLOCK\_CYCLES\_FOR\_2MS)

ns\_ps2\_transmitter = PS2\_STATE\_7\_TRANSMISSION\_ERROR;

else

ns\_ps2\_transmitter = PS2\_STATE\_5\_RECEIVE\_ACK\_BIT;

end

PS2\_STATE\_6\_COMMAND\_WAS\_SENT:

begin

if (send\_command == 1'b0)

ns\_ps2\_transmitter = PS2\_STATE\_0\_IDLE;

else

ns\_ps2\_transmitter = PS2\_STATE\_6\_COMMAND\_WAS\_SENT;

end

PS2\_STATE\_7\_TRANSMISSION\_ERROR:

begin

if (send\_command == 1'b0)

ns\_ps2\_transmitter = PS2\_STATE\_0\_IDLE;

else

ns\_ps2\_transmitter = PS2\_STATE\_7\_TRANSMISSION\_ERROR;

end

default:

begin

ns\_ps2\_transmitter = PS2\_STATE\_0\_IDLE;

end

endcase

end

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

\* Sequential logic \*

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

always @(posedge clk)

begin

if (reset == 1'b1)

ps2\_command <= 9'h000;

else if (s\_ps2\_transmitter == PS2\_STATE\_0\_IDLE)

ps2\_command <= {(^the\_command) ^ 1'b1, the\_command};

end

always @(posedge clk)

begin

if (reset == 1'b1)

command\_initiate\_counter <= {NUMBER\_OF\_BITS\_FOR\_101US{1'b0}};

else if ((s\_ps2\_transmitter == PS2\_STATE\_1\_INITIATE\_COMMUNICATION) &&

(command\_initiate\_counter != CLOCK\_CYCLES\_FOR\_101US))

command\_initiate\_counter <=

command\_initiate\_counter + COUNTER\_INCREMENT\_FOR\_101US;

else if (s\_ps2\_transmitter != PS2\_STATE\_1\_INITIATE\_COMMUNICATION)

command\_initiate\_counter <= {NUMBER\_OF\_BITS\_FOR\_101US{1'b0}};

end

always @(posedge clk)

begin

if (reset == 1'b1)

waiting\_counter <= {NUMBER\_OF\_BITS\_FOR\_15MS{1'b0}};

else if ((s\_ps2\_transmitter == PS2\_STATE\_2\_WAIT\_FOR\_CLOCK) &&

(waiting\_counter != CLOCK\_CYCLES\_FOR\_15MS))

waiting\_counter <= waiting\_counter + COUNTER\_INCREMENT\_FOR\_15MS;

else if (s\_ps2\_transmitter != PS2\_STATE\_2\_WAIT\_FOR\_CLOCK)

waiting\_counter <= {NUMBER\_OF\_BITS\_FOR\_15MS{1'b0}};

end

always @(posedge clk)

begin

if (reset == 1'b1)

transfer\_counter <= {NUMBER\_OF\_BITS\_FOR\_2MS{1'b0}};

else

begin

if ((s\_ps2\_transmitter == PS2\_STATE\_3\_TRANSMIT\_DATA) ||

(s\_ps2\_transmitter == PS2\_STATE\_4\_TRANSMIT\_STOP\_BIT) ||

(s\_ps2\_transmitter == PS2\_STATE\_5\_RECEIVE\_ACK\_BIT))

begin

if (transfer\_counter != CLOCK\_CYCLES\_FOR\_2MS)

transfer\_counter <= transfer\_counter + COUNTER\_INCREMENT\_FOR\_2MS;

end

else

transfer\_counter <= {NUMBER\_OF\_BITS\_FOR\_2MS{1'b0}};

end

end

always @(posedge clk)

begin

if (reset == 1'b1)

cur\_bit <= 4'h0;

else if ((s\_ps2\_transmitter == PS2\_STATE\_3\_TRANSMIT\_DATA) &&

(ps2\_clk\_negedge == 1'b1))

cur\_bit <= cur\_bit + 4'h1;

else if (s\_ps2\_transmitter != PS2\_STATE\_3\_TRANSMIT\_DATA)

cur\_bit <= 4'h0;

end

always @(posedge clk)

begin

if (reset == 1'b1)

command\_was\_sent <= 1'b0;

else if (s\_ps2\_transmitter == PS2\_STATE\_6\_COMMAND\_WAS\_SENT)

command\_was\_sent <= 1'b1;

else if (send\_command == 1'b0)

command\_was\_sent <= 1'b0;

end

always @(posedge clk)

begin

if (reset == 1'b1)

error\_communication\_timed\_out <= 1'b0;

else if (s\_ps2\_transmitter == PS2\_STATE\_7\_TRANSMISSION\_ERROR)

error\_communication\_timed\_out <= 1'b1;

else if (send\_command == 1'b0)

error\_communication\_timed\_out <= 1'b0;

end

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

\* Combinational logic \*

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

assign PS2\_CLK =

(s\_ps2\_transmitter == PS2\_STATE\_1\_INITIATE\_COMMUNICATION) ?

1'b0 :

1'bz;

assign PS2\_DAT =

(s\_ps2\_transmitter == PS2\_STATE\_3\_TRANSMIT\_DATA) ? ps2\_command[cur\_bit] :

(s\_ps2\_transmitter == PS2\_STATE\_2\_WAIT\_FOR\_CLOCK) ? 1'b0 :

((s\_ps2\_transmitter == PS2\_STATE\_1\_INITIATE\_COMMUNICATION) &&

(command\_initiate\_counter[NUMBER\_OF\_BITS\_FOR\_101US] == 1'b1)) ? 1'b0 :

1'bz;

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

\* Internal Modules \*

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

endmodule

module FSM (clk, resetn, start\_key, done\_draw, done\_wait, done\_game, go\_reset, go\_bkgd, go\_draw, go\_wait, go\_update, go\_erase, go\_win);

input clk;

input resetn;

input start\_key;

input done\_draw;

input done\_wait;

input done\_game;

output reg go\_reset;

output reg go\_bkgd;

output reg go\_draw;

output reg go\_wait;

output reg go\_update;

output reg go\_erase;

output reg go\_win;

//declare four states

localparam reset\_state= 4'b0000, bkgd\_state=4'b0101, draw\_state = 4'b0001, wait\_state = 4'b0010, update\_state = 4'b0011, erase\_state = 4'b0100, win\_state = 4'b0110;

reg [3:0] current\_state = reset\_state, next\_state;

//state transition

always@(\*)

begin: state\_table

case (current\_state)

reset\_state: next\_state = start\_key ? bkgd\_state : reset\_state;

bkgd\_state: next\_state = done\_draw ? draw\_state : bkgd\_state;

draw\_state: next\_state = done\_draw ? wait\_state : draw\_state;

wait\_state: next\_state = done\_wait ? erase\_state : wait\_state;

erase\_state: next\_state = done\_draw ? update\_state : erase\_state;

update\_state: next\_state = done\_game ? win\_state : draw\_state;

win\_state: next\_state = win\_state;

endcase

end

//signal changes in each state

always @(\*)

begin: enable\_signals

go\_reset=1;

go\_bkgd=0;

go\_draw=0;

go\_wait=0;

go\_update=0;

go\_erase=0;

go\_win=0;

case (current\_state)

reset\_state:

begin

go\_reset=1;

go\_bkgd=0;

go\_draw=1;

go\_wait=0;

go\_erase=0;

go\_update=0;

go\_win=0;

end

bkgd\_state:

begin

go\_reset=0;

go\_bkgd=1;

go\_draw=1;

go\_wait=0;

go\_erase=0;

go\_update=0;

go\_win=0;

end

draw\_state:

begin

go\_reset=0;

go\_bkgd=0;

go\_draw=1;

go\_wait=0;

go\_erase=0;

go\_update=0;

go\_win=0;

end

wait\_state:

begin

go\_reset=0;

go\_bkgd=0;

go\_draw=0;

go\_wait=1;

go\_erase=0;

go\_update=0;

go\_win=0;

end

erase\_state:

begin

go\_reset=0;

go\_bkgd=0;

go\_draw=1;

go\_wait=0;

go\_erase=1;

go\_update=0;

go\_win=0;

end

update\_state:

begin

go\_reset=0;

go\_bkgd=0;

go\_draw=0;

go\_wait=0;

go\_erase=0;

go\_update=1;

go\_win=0;

end

win\_state:

begin

go\_reset=0;

go\_bkgd=0;

go\_draw=1;

go\_wait=0;

go\_erase=0;

go\_update=0;

go\_win=1;

end

endcase

end

//the actual state changes according to the clock edge

always @(posedge clk)

begin: state\_FFs

if(resetn == 0)

current\_state <= reset\_state;

else

current\_state <= next\_state;

end

endmodule

module datapath (HEX3,HEX2,HEX1,HEX0, clk\_50, resetn, key\_pressed\_data, button\_pressed\_data, colour\_out, go\_reset, go\_bkgd, go\_draw, go\_wait, go\_update, go\_erase, go\_win, done\_draw, done\_wait, done\_game, x\_out, y\_out);

output [6:0] HEX3; //newly

output [6:0] HEX2; //newly

output [6:0] HEX1; //newly

output [6:0] HEX0; //newly

input clk\_50;

input resetn;

input [7:0]key\_pressed\_data; //newly declared

input [2:0]button\_pressed\_data;

input go\_reset;

input go\_bkgd;

input go\_draw;

input go\_wait;

input go\_update;

input go\_erase;

input go\_win;

output reg [8:0]colour\_out;

output done\_draw;

output done\_wait;

output reg done\_game=0;

output reg [7:0]x\_out;

output reg [6:0]y\_out;

//wait-block

reg [7:0] key\_update\_data=8'h00;

reg [2:0] button\_update\_data=3'b111;

reg [27:0]out=28'd9999999;

always@(posedge clk\_50)

begin

if(!resetn)

begin

out<=28'd9999999;

key\_update\_data=8'h00;

button\_update\_data=3'b111;

end

else if (go\_wait)

begin

if (out[27:0]==28'd0)

out<=28'd9999999;

else

out<=out-1;

if (out[27:0]==28'd9999999)

begin

key\_update\_data=8'h00;

button\_update\_data=3'b111;

end

else

begin

if (key\_pressed\_data!=8'h00)//change

key\_update\_data<=key\_pressed\_data;

if (button\_pressed\_data!=3'b111)

button\_update\_data<=button\_pressed\_data;

end

end

end

assign done\_wait = (out==28'd0) ? 1:0;

//player1 horizontal update-block

localparam kick\_key=8'h3b, punch\_key=8'h4b, jump\_key=8'h1d, crouch\_key=8'h1b, left\_key=8'h1c, right\_key=8'h23;

reg [7:0] x\_start1=8'd1;

reg [6:0] y\_start1=7'd0;

always@(posedge clk\_50)

begin

if (!resetn)

begin

x\_start1=8'd1;

y\_start1=7'd0;

end

else if (go\_update && (key\_update\_data==left\_key) && ((x\_start1-1)>0))

x\_start1<=x\_start1-1;

else if (go\_update && (key\_update\_data==right\_key) && ((x\_start1+1)<=117) && ((x\_start1+1)<=(x\_start2-44)))

x\_start1<=x\_start1+1;

end

//player2 horizontal update-block

localparam kick\_button = 3'b000, punch\_button = 3'b001, jump\_button = 3'b010, crouch\_button = 3'b011, left\_button = 3'b100, right\_button = 3'b101;

reg [7:0] x\_start2=8'd115;

reg [6:0] y\_start2=7'd0;

always@(posedge clk\_50)

begin

if (!resetn)

begin

x\_start2=8'd115;

y\_start2=7'd0;

end

else if (go\_update && (button\_update\_data==left\_button) && ((x\_start2-1)>0) && x\_start1<=(x\_start2-1-44))

x\_start2<=x\_start2-1;

else if (go\_update && (button\_update\_data==right\_button)&& (x\_start2+1)<=117)

x\_start2<=x\_start2+1;

end

reg [1:0] GameState=0;

localparam no\_wins=2'b00, player1\_wins=2'b10, player2\_wins=2'b01;

always@(\*)

begin

if (go\_reset)

begin

GameState=no\_wins;

done\_game=0;

end

else if (player1\_health==40)

begin

GameState=player1\_wins;

done\_game=1;

end

else if (player2\_health==40)

begin

GameState=player2\_wins;

done\_game=1;

end

else

done\_game=0;

end

//draw-block and erase-block: x,y

reg [14:0]screen\_temp;

reg [14:0]bkgd0\_temp;

reg [13:0]temp;

reg [14:0]win\_temp;

always@(posedge clk\_50)

begin

if(!go\_draw)

begin

temp<=8800;

screen\_temp<=19200;

bkgd0\_temp<=19200;

win\_temp<=19200;

x\_out <= x\_start1;

y\_out <= y\_start1;

end

else if (go\_draw)

begin

//in reset state

if (go\_reset)

begin

if (screen\_temp!=0)

begin

screen\_temp <= screen\_temp-1;

x\_out <= (screen\_temp-1)%160;

y\_out <= (screen\_temp-1)/160;

end

else if(screen\_temp==0)

screen\_temp<=19200;

end

//in bkgd state

else if (go\_bkgd)

begin

if (bkgd0\_temp!=0)

begin

bkgd0\_temp <= bkgd0\_temp-1;

x\_out <= (bkgd0\_temp-1)%160;

y\_out <= (bkgd0\_temp-1)/160;

end

else if(bkgd0\_temp==0)

bkgd0\_temp<=19200;

end

//not in those state

else if (!go\_reset && !go\_bkgd && !go\_win)

begin

if (temp!=0) temp <= temp-1;

if(temp > 4400)

begin

x\_out <= x\_start1 + (temp-1-4400)%44;

y\_out <= y\_start1 + (temp-1-4400)/44;

end

else if (temp <= 4400)

begin

x\_out <= x\_start2 + (temp-1)%44;

y\_out <= y\_start2 + (temp-1)/44;

end

end

else if (go\_win)

begin

if (win\_temp!=0)

begin

win\_temp <= win\_temp-1;

x\_out <= (win\_temp-1)%160;

y\_out <= (win\_temp-1)/160;

end

else if(win\_temp==0)

win\_temp<=19200;

end

end

end

//Hexadecimal\_To\_Seven\_Segment debug3(x\_start1[6:4],HEX2);

//Hexadecimal\_To\_Seven\_Segment debug2(x\_start1[3:0],HEX2);

//Hexadecimal\_To\_Seven\_Segment debug1(y\_start1[6:4],HEX1);

//Hexadecimal\_To\_Seven\_Segment debug0(y\_start1[3:0],HEX0);

//draw-block and erase-block: color\_selection

wire [3:0] player1\_state, player2\_state;

wire [5:0] player1\_health,player2\_health;

wire [7:0] dist\_between;

assign dist\_between=(x\_start2-x\_start1-44);

moves\_top m\_t1 (

.clk(clk\_50), .resetn(resetn),

.kick\_in\_1(key\_update\_data==8'h3b), .punch\_in\_1(key\_update\_data==8'h4b), .crouch\_in\_1(key\_update\_data==8'h1b), .jump\_in\_1(key\_update\_data==8'h1d),

.kick\_in\_2(button\_update\_data==kick\_button), .punch\_in\_2(button\_update\_data==punch\_button), .crouch\_in\_2(button\_update\_data==crouch\_button), .jump\_in\_2(button\_update\_data==jump\_button),

.distance(),

.p1\_health1(), .p1\_health0(), .p2\_health1(), .p2\_health0(),

.player1\_state\_out(player1\_state), .player2\_state\_out(player2\_state),

.player1\_health\_out(), .player2\_health\_out());

moves\_top m\_t2 (

.clk(done\_wait), .resetn(resetn),

.kick\_in\_1(key\_update\_data==8'h3b), .punch\_in\_1(key\_update\_data==8'h4b), .crouch\_in\_1(key\_update\_data==8'h1b), .jump\_in\_1(key\_update\_data==8'h1d),

.kick\_in\_2(button\_update\_data==kick\_button), .punch\_in\_2(button\_update\_data==punch\_button), .crouch\_in\_2(button\_update\_data==crouch\_button), .jump\_in\_2(button\_update\_data==jump\_button),

.distance(dist\_between),

.p1\_health1(HEX3), .p1\_health0(HEX2), .p2\_health1(HEX1), .p2\_health0(HEX0),

.player1\_state\_out(), .player2\_state\_out(),

.player1\_health\_out(player1\_health), .player2\_health\_out(player2\_health));

//sprite colors

reg [8:0] sprite\_color;

// p1 colors

wire [12:0] sprite\_pixel\_address;

assign sprite\_pixel\_address=(temp>4400)?(temp-4400-1):(temp-1);

wire [8:0] sprite1\_idle\_color;

wire [8:0] sprite1\_kicking\_color;

wire [8:0] sprite1\_punching\_color;

wire [8:0] sprite1\_crouching\_color;

wire [8:0] sprite1\_jumping\_color;

wire [8:0] sprite1\_sweeping\_color;

wire [8:0] sprite1\_jumpkicking\_color;

sprite1\_idle\_rom m12( sprite\_pixel\_address, clk\_50, sprite1\_idle\_color);

sprite1\_kicking\_rom m13( sprite\_pixel\_address, clk\_50, sprite1\_kicking\_color);

sprite1\_punching\_rom m14( sprite\_pixel\_address, clk\_50, sprite1\_punching\_color);

sprite1\_crouching\_rom m15( sprite\_pixel\_address, clk\_50, sprite1\_crouching\_color);

sprite1\_jumping\_rom m16( sprite\_pixel\_address, clk\_50, sprite1\_jumping\_color);

sprite1\_sweeping\_rom m17( sprite\_pixel\_address, clk\_50, sprite1\_sweeping\_color);

sprite1\_jumpkicking\_rom m18( sprite\_pixel\_address, clk\_50, sprite1\_jumpkicking\_color);

// p2 colors

wire [8:0] sprite2\_idle\_color;

wire [8:0] sprite2\_kicking\_color;

wire [8:0] sprite2\_punching\_color;

wire [8:0] sprite2\_crouching\_color;

wire [8:0] sprite2\_jumping\_color;

wire [8:0] sprite2\_sweeping\_color;

wire [8:0] sprite2\_jumpkicking\_color;

sprite2\_idle\_rom m22( sprite\_pixel\_address, clk\_50, sprite2\_idle\_color);

sprite2\_kicking\_rom m23( sprite\_pixel\_address, clk\_50, sprite2\_kicking\_color);

sprite2\_punching\_rom m24( sprite\_pixel\_address, clk\_50, sprite2\_punching\_color);

sprite2\_crouching\_rom m25( sprite\_pixel\_address, clk\_50, sprite2\_crouching\_color);

sprite2\_jumping\_rom m26( sprite\_pixel\_address, clk\_50, sprite2\_jumping\_color);

sprite2\_sweeping\_rom m27( sprite\_pixel\_address, clk\_50, sprite2\_sweeping\_color);

sprite2\_jumpkicking\_rom m28( sprite\_pixel\_address, clk\_50, sprite2\_jumpkicking\_color);

always@(\*)

begin

if (temp>4400)

begin

case(player1\_state)

3'b000: sprite\_color=sprite1\_idle\_color;

3'b100: sprite\_color=sprite1\_jumping\_color;

3'b001: sprite\_color=sprite1\_kicking\_color;

3'b010: sprite\_color=sprite1\_punching\_color;

3'b011: sprite\_color=sprite1\_crouching\_color;

3'b101: sprite\_color=sprite1\_sweeping\_color;

3'b110: sprite\_color=sprite1\_jumpkicking\_color;

default: sprite\_color=sprite1\_idle\_color;

endcase

end

else

begin

case(player2\_state)// change to p2 ver.

3'b000: sprite\_color=sprite2\_idle\_color;

3'b100: sprite\_color=sprite2\_jumping\_color;

3'b001: sprite\_color=sprite2\_kicking\_color;

3'b010: sprite\_color=sprite2\_punching\_color;

3'b011: sprite\_color=sprite2\_crouching\_color;

3'b101: sprite\_color=sprite2\_sweeping\_color;

3'b110: sprite\_color=sprite2\_jumpkicking\_color;

default: sprite\_color=sprite2\_idle\_color;

endcase

end

end

// background color

wire [8:0]bkgd\_color;

wire [14:0]bkgd\_pixel\_address;

assign bkgd\_pixel\_address=(temp>4400)?

(y\_start1 + (temp-1-4400)/44)\*15'd160 + (x\_start1 + (temp-1-4400)%44):

(y\_start2 + (temp-1)/44)\*15'd160 + (x\_start2 + (temp-1)%44);

background\_rom m3(bkgd\_pixel\_address, clk\_50, bkgd\_color);

//start screen color

wire [8:0]start\_screen\_color;

wire [14:0]start\_pixel\_address;

assign start\_pixel\_address=screen\_temp-1;

startscreen\_rom m1(start\_pixel\_address, clk\_50, start\_screen\_color);

//background initializer color

wire [8:0]bkgd0\_color;

wire [14:0]bkgd0\_pixel\_address;

assign bkgd0\_pixel\_address=bkgd0\_temp-1;

background\_rom m2(bkgd0\_pixel\_address, clk\_50, bkgd0\_color);

//win color

wire [8:0]win\_color;

wire [8:0]p1\_win\_color;

wire [8:0]p2\_win\_color;

wire [14:0]win\_pixel\_address;

assign win\_pixel\_address=win\_temp-1;

p1\_win\_rom wm1(win\_pixel\_address, clk\_50, p1\_win\_color);

p2\_win\_rom wm2(win\_pixel\_address, clk\_50, p2\_win\_color);

assign win\_color=(GameState==player1\_wins)?p1\_win\_color:p2\_win\_color;

//final colour selection

always@(\*)

begin

if (go\_reset) colour\_out = start\_screen\_color;

else if (go\_bkgd) colour\_out = bkgd0\_color;

else if (go\_win) colour\_out = win\_color;

else if (go\_erase || sprite\_color==9'b0) colour\_out = bkgd\_color;

else colour\_out = sprite\_color;

end

assign done\_draw =(temp==0 || screen\_temp==0 || bkgd0\_temp==0);

endmodule

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

\* \*

\* Module: Hexadecimal\_To\_Seven\_Segment \*

\* Description: \*

\* This module converts hexadecimal numbers for seven segment displays. \*

\* \*

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

module Hexadecimal\_To\_Seven\_Segment (

// Inputs

hex\_number,

// Bidirectional

// Outputs

seven\_seg\_display

);

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

\* Parameter Declarations \*

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

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

\* Port Declarations \*

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

// Inputs

input [3:0] hex\_number;

// Bidirectional

// Outputs

output [6:0] seven\_seg\_display;

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

\* Internal Wires and Registers Declarations \*

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

// Internal Wires

// Internal Registers

// State Machine Registers

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

\* Finite State Machine(s) \*

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

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

\* Sequential Logic \*

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

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

\* Combinational Logic \*

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

assign seven\_seg\_display =

({7{(hex\_number == 4'h0)}} & 7'b1000000) |

({7{(hex\_number == 4'h1)}} & 7'b1111001) |

({7{(hex\_number == 4'h2)}} & 7'b0100100) |

({7{(hex\_number == 4'h3)}} & 7'b0110000) |

({7{(hex\_number == 4'h4)}} & 7'b0011001) |

({7{(hex\_number == 4'h5)}} & 7'b0010010) |

({7{(hex\_number == 4'h6)}} & 7'b0000010) |

({7{(hex\_number == 4'h7)}} & 7'b1111000) |

({7{(hex\_number == 4'h8)}} & 7'b0000000) |

({7{(hex\_number == 4'h9)}} & 7'b0010000) |

({7{(hex\_number == 4'hA)}} & 7'b0001000) |

({7{(hex\_number == 4'hB)}} & 7'b0000011) |

({7{(hex\_number == 4'hC)}} & 7'b1000110) |

({7{(hex\_number == 4'hD)}} & 7'b0100001) |

({7{(hex\_number == 4'hE)}} & 7'b0000110) |

({7{(hex\_number == 4'hF)}} & 7'b0001110);

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

\* Internal Modules \*

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

endmodule

module moves\_top(

clk, resetn,

kick\_in\_1, punch\_in\_1, crouch\_in\_1, jump\_in\_1,

kick\_in\_2, punch\_in\_2, crouch\_in\_2, jump\_in\_2,

distance,

p1\_health1, p1\_health0, p2\_health1, p2\_health0,

player1\_state\_out, player2\_state\_out,

player1\_health\_out, player2\_health\_out);

input clk, resetn;

input kick\_in\_1, punch\_in\_1, crouch\_in\_1, jump\_in\_1;

input kick\_in\_2, punch\_in\_2, crouch\_in\_2, jump\_in\_2;

input [7:0] distance;

output [6:0] p1\_health1;

output [6:0] p1\_health0;

output [6:0] p2\_health1;

output [6:0] p2\_health0;

output [3:0] player1\_state\_out;

output [3:0] player2\_state\_out;

assign player1\_state\_out = go\_player1\_state;

assign player2\_state\_out = go\_player2\_state;

output [5:0] player1\_health\_out;

output [5:0] player2\_health\_out;

assign player1\_health\_out = player1\_health;

assign player2\_health\_out = player2\_health;

wire [3:0] go\_player1\_state;

wire [3:0] go\_player2\_state;

wire [5:0] player1\_health;

wire [5:0] player2\_health;

moves\_FSM player1 (clk, resetn, kick\_in\_1, punch\_in\_1, crouch\_in\_1, jump\_in\_1, go\_player1\_state);

moves\_FSM player2 (clk, resetn, kick\_in\_2, punch\_in\_2, crouch\_in\_2, jump\_in\_2, go\_player2\_state);

moves\_datapath moves\_datapath1 (clk,resetn,distance,go\_player1\_state,go\_player2\_state,player1\_health,player2\_health);

Hexadecimal\_To\_Seven\_Segment h7((player1\_health/10), p1\_health1);

Hexadecimal\_To\_Seven\_Segment h6((player1\_health%10), p1\_health0);

Hexadecimal\_To\_Seven\_Segment h5((player2\_health/10), p2\_health1);

Hexadecimal\_To\_Seven\_Segment h4((player2\_health%10), p2\_health0);

endmodule

module moves\_FSM (clk, resetn, kick\_in, punch\_in, crouch\_in, jump\_in, go\_state);

input clk, resetn, kick\_in, punch\_in, crouch\_in, jump\_in;

output [3:0] go\_state;

assign go\_state = current\_state;

//declare four states

localparam idle\_state= 3'b000, kicking\_state = 3'b001, punching\_state = 3'b010, crouching\_state = 3'b011, jumping\_state = 3'b100, sweeping\_state = 3'b101, jumpkicking\_state=3'b110;

reg [2:0] current\_state=idle\_state;

reg [2:0] next\_state;

//TRY THIS NOW!!!

//state transition

always@(\*)

begin: state\_table

case (current\_state)

idle\_state:

begin

if (kick\_in) next\_state = kicking\_state;

else if (punch\_in) next\_state = punching\_state;

else if (crouch\_in) next\_state = crouching\_state;

else if (jump\_in) next\_state = jumping\_state;

else next\_state=idle\_state;

end

kicking\_state:

begin

if (crouch\_in) next\_state = crouching\_state;

else if (jump\_in) next\_state = jumping\_state;

else if (kick\_in) next\_state = kicking\_state;

else next\_state = idle\_state;

end

punching\_state:

begin

if (crouch\_in) next\_state = crouching\_state;

else if (jump\_in) next\_state = jumping\_state;

else if (punch\_in) next\_state = punching\_state;

else next\_state = idle\_state;

end

crouching\_state:

begin

if (kick\_in) next\_state = sweeping\_state;

else if (crouch\_in) next\_state = crouching\_state;

else next\_state = idle\_state;

end

jumping\_state:

begin

if (kick\_in) next\_state = jumpkicking\_state;

else if (jump\_in) next\_state = jumping\_state;

else next\_state = idle\_state;

end

sweeping\_state:

begin

if (kick\_in) next\_state = sweeping\_state;

else next\_state = idle\_state;

end

jumpkicking\_state:

begin

if (kick\_in) next\_state = jumpkicking\_state;

else next\_state = idle\_state;

end

default: next\_state = idle\_state;

endcase

end

//the actual state changes according to the clock edge

always @(posedge clk)

begin: state\_FFs

if(resetn == 0)

current\_state <= idle\_state;

else

current\_state <= next\_state;

end

endmodule

module moves\_datapath(clk,reset,distance,player1\_state,player2\_state,player1\_health,player2\_health);

input clk,reset;

input [7:0] distance;

input [2:0] player1\_state,player2\_state;

output reg [5:0] player1\_health=20;//multi-bits

output [5:0] player2\_health ;//80-player1\_health

localparam idle\_state= 3'b000, kicking\_state = 3'b001, punching\_state = 3'b010, crouching\_state = 3'b011, jumping\_state = 3'b100, sweeping\_state = 3'b101, jumpkicking\_state=3'b110;

always@(posedge clk, negedge reset)

begin

if (!reset)

begin

player1\_health<=20;

end

else

begin

if (distance>1)//>=1 ?

player1\_health<=player1\_health;

else if (player1\_state==idle\_state && player2\_state==idle\_state)

player1\_health<=player1\_health;

else if (player1\_state==idle\_state && player2\_state==punching\_state)

player1\_health<=(player1\_health<1)?0:player1\_health-1;

else if (player1\_state==idle\_state && player2\_state==kicking\_state)

player1\_health<=(player1\_health<2)?0:player1\_health-2;

else if (player1\_state==idle\_state && player2\_state==jumping\_state)

player1\_health<=player1\_health;

else if (player1\_state==idle\_state && player2\_state==crouching\_state)

player1\_health<=player1\_health;

else if (player1\_state==idle\_state && player2\_state==sweeping\_state)

player1\_health<=(player1\_health<3)?0:player1\_health-3;

else if (player1\_state==idle\_state && player2\_state==jumpkicking\_state)

player1\_health<=(player1\_health<4)?0:player1\_health-4;

else if (player1\_state==punching\_state && player2\_state==idle\_state)

player1\_health<=(player1\_health>39)?40:player1\_health+1;

else if (player1\_state==punching\_state && player2\_state==punching\_state)

player1\_health<=player1\_health;

else if (player1\_state==punching\_state && player2\_state==kicking\_state)

player1\_health<=(player1\_health<1)?0:player1\_health-1;

else if (player1\_state==punching\_state && player2\_state==jumping\_state)

player1\_health<=(player1\_health>39)?40:player1\_health+1;

else if (player1\_state==punching\_state && player2\_state==crouching\_state)

player1\_health<=player1\_health;

else if (player1\_state==punching\_state && player2\_state==sweeping\_state)

player1\_health<=(player1\_health<3)?0:player1\_health-3;

else if (player1\_state==punching\_state && player2\_state==jumpkicking\_state)

player1\_health<=(player1\_health<3)?0:player1\_health-3;

else if (player1\_state==kicking\_state && player2\_state==idle\_state)

player1\_health<=(player1\_health>38)?40:player1\_health+2;

else if (player1\_state==kicking\_state && player2\_state==punching\_state)

player1\_health<=(player1\_health>39)?40:player1\_health+1;

else if (player1\_state==kicking\_state && player2\_state==kicking\_state)

player1\_health<=player1\_health;

else if (player1\_state==kicking\_state && player2\_state==jumping\_state)

player1\_health<=player1\_health;

else if (player1\_state==kicking\_state && player2\_state==crouching\_state)

player1\_health<=(player1\_health>38)?40:player1\_health+2;

else if (player1\_state==kicking\_state && player2\_state==sweeping\_state)

player1\_health<=(player1\_health<1)?0:player1\_health-1;

else if (player1\_state==kicking\_state && player2\_state==jumpkicking\_state)

player1\_health<=(player1\_health<4)?0:player1\_health-4;

else if (player1\_state==jumping\_state && player2\_state==idle\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumping\_state && player2\_state==punching\_state)

player1\_health<=(player1\_health<1)?0:player1\_health-1;

else if (player1\_state==jumping\_state && player2\_state==kicking\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumping\_state && player2\_state==jumping\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumping\_state && player2\_state==crouching\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumping\_state && player2\_state==sweeping\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumping\_state && player2\_state==jumpkicking\_state)

player1\_health<=(player1\_health<4)?0:player1\_health-4;

else if (player1\_state==crouching\_state && player2\_state==idle\_state)

player1\_health<=player1\_health;

else if (player1\_state==crouching\_state && player2\_state==punching\_state)

player1\_health<=player1\_health;

else if (player1\_state==crouching\_state && player2\_state==kicking\_state)

player1\_health<=(player1\_health<2)?0:player1\_health-2;

else if (player1\_state==crouching\_state && player2\_state==jumping\_state)

player1\_health<=player1\_health;

else if (player1\_state==crouching\_state && player2\_state==crouching\_state)

player1\_health<=player1\_health;

else if (player1\_state==crouching\_state && player2\_state==sweeping\_state)

player1\_health<=(player1\_health<3)?0:player1\_health-3;

else if (player1\_state==crouching\_state && player2\_state==jumpkicking\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumpkicking\_state && player2\_state==idle\_state)

player1\_health<=(player1\_health>36)?40:player1\_health+4;

else if (player1\_state==jumpkicking\_state && player2\_state==punching\_state)

player1\_health<=(player1\_health>37)?40:player1\_health+3;

else if (player1\_state==jumpkicking\_state && player2\_state==kicking\_state)

player1\_health<=(player1\_health>36)?40:player1\_health+4;

else if (player1\_state==jumpkicking\_state && player2\_state==jumping\_state)

player1\_health<=(player1\_health>36)?40:player1\_health+4;

else if (player1\_state==jumpkicking\_state && player2\_state==crouching\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumpkicking\_state && player2\_state==sweeping\_state)

player1\_health<=player1\_health;

else if (player1\_state==jumpkicking\_state && player2\_state==jumpkicking\_state)

player1\_health<=player1\_health;

else if (player1\_state==sweeping\_state && player2\_state==idle\_state)

player1\_health<=(player1\_health>37)?40:player1\_health+3;

else if (player1\_state==sweeping\_state && player2\_state==punching\_state)

player1\_health<=(player1\_health>37)?40:player1\_health+3;

else if (player1\_state==sweeping\_state && player2\_state==kicking\_state)

player1\_health<=(player1\_health>39)?40:player1\_health+1;

else if (player1\_state==sweeping\_state && player2\_state==jumping\_state)

player1\_health<=player1\_health;

else if (player1\_state==sweeping\_state && player2\_state==crouching\_state)

player1\_health<=(player1\_health>37)?40:player1\_health+3;

else if (player1\_state==sweeping\_state && player2\_state==sweeping\_state)

player1\_health<=player1\_health;

else if (player1\_state==sweeping\_state && player2\_state==jumpkicking\_state)

player1\_health<=player1\_health;

else

player1\_health<=player1\_health;

end

end

assign player2\_health=7'd40-player1\_health;

endmodule

/-\* This module implements the VGA controller. It assumes a 25MHz clock is supplied as input.

\*

\* General approach:

\* Go through each line of the screen and read the colour each pixel on that line should have from

\* the Video memory. To do that for each (x,y) pixel on the screen convert (x,y) coordinate to

\* a memory\_address at which the pixel colour is stored in Video memory. Once the pixel colour is

\* read from video memory its brightness is first increased before it is forwarded to the VGA DAC.

\*-/

module vga\_controller( vga\_clock, resetn, pixel\_colour, memory\_address,

VGA\_R, VGA\_G, VGA\_B,

VGA\_HS, VGA\_VS, VGA\_BLANK,

VGA\_SYNC, VGA\_CLK);

/-\* Screen resolution and colour depth parameters. \*-/

parameter BITS\_PER\_COLOUR\_CHANNEL = 1;

/-\* The number of bits per colour channel used to represent the colour of each pixel. A value

\* of 1 means that Red, Green and Blue colour channels will use 1 bit each to represent the intensity

\* of the respective colour channel. For BITS\_PER\_COLOUR\_CHANNEL=1, the adapter can display 8 colours.

\* In general, the adapter is able to use 2^(3\*BITS\_PER\_COLOUR\_CHANNEL) colours. The number of colours is

\* limited by the screen resolution and the amount of on-chip memory available on the target device.

\*-/

parameter MONOCHROME = "FALSE";

/-\* Set this parameter to "TRUE" if you only wish to use black and white colours. Doing so will reduce

\* the amount of memory you will use by a factor of 3. \*-/

parameter RESOLUTION = "320x240";

/-\* Set this parameter to "160x120" or "320x240". It will cause the VGA adapter to draw each dot on

\* the screen by using a block of 4x4 pixels ("160x120" resolution) or 2x2 pixels ("320x240" resolution).

\* It effectively reduces the screen resolution to an integer fraction of 640x480. It was necessary

\* to reduce the resolution for the Video Memory to fit within the on-chip memory limits.

\*-/

//--- Timing parameters.

/-\* Recall that the VGA specification requires a few more rows and columns are drawn

\* when refreshing the screen than are actually present on the screen. This is necessary to

\* generate the vertical and the horizontal syncronization signals. If you wish to use a

\* display mode other than 640x480 you will need to modify the parameters below as well

\* as change the frequency of the clock driving the monitor (VGA\_CLK).

\*-/

parameter C\_VERT\_NUM\_PIXELS = 10'd480;

parameter C\_VERT\_SYNC\_START = 10'd493;

parameter C\_VERT\_SYNC\_END = 10'd494; //(C\_VERT\_SYNC\_START + 2 - 1);

parameter C\_VERT\_TOTAL\_COUNT = 10'd525;

parameter C\_HORZ\_NUM\_PIXELS = 10'd640;

parameter C\_HORZ\_SYNC\_START = 10'd659;

parameter C\_HORZ\_SYNC\_END = 10'd754; //(C\_HORZ\_SYNC\_START + 96 - 1);

parameter C\_HORZ\_TOTAL\_COUNT = 10'd800;

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

/-\* Declare inputs and outputs. \*-/

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

input vga\_clock, resetn;

input [((MONOCHROME == "TRUE") ? (0) : (BITS\_PER\_COLOUR\_CHANNEL\*3-1)):0] pixel\_colour;

output [((RESOLUTION == "320x240") ? (16) : (14)):0] memory\_address;

output reg [9:0] VGA\_R;

output reg [9:0] VGA\_G;

output reg [9:0] VGA\_B;

output reg VGA\_HS;

output reg VGA\_VS;

output reg VGA\_BLANK;

output VGA\_SYNC, VGA\_CLK;

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

/-\* Local Signals. \*-/

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

reg VGA\_HS1;

reg VGA\_VS1;

reg VGA\_BLANK1;

reg [9:0] xCounter, yCounter;

wire xCounter\_clear;

wire yCounter\_clear;

wire vcc;

reg [((RESOLUTION == "320x240") ? (8) : (7)):0] x;

reg [((RESOLUTION == "320x240") ? (7) : (6)):0] y;

/-\* Inputs to the converter. \*-/

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

/-\* Controller implementation. \*-/

/-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-/

assign vcc =1'b1;

/-\* A counter to scan through a horizontal line. \*-/

always @(posedge vga\_clock or negedge resetn)

begin

if (!resetn)

xCounter <= 10'd0;

else if (xCounter\_clear)

xCounter <= 10'd0;

else

begin

xCounter <= xCounter + 1'b1;

end

end

assign xCounter\_clear = (xCounter == (C\_HORZ\_TOTAL\_COUNT-1));

/-\* A counter to scan vertically, indicating the row currently being drawn. \*-/

always @(posedge vga\_clock or negedge resetn)

begin

if (!resetn)

yCounter <= 10'd0;

else if (xCounter\_clear && yCounter\_clear)

yCounter <= 10'd0;

else if (xCounter\_clear) //Increment when x counter resets

yCounter <= yCounter + 1'b1;

end

assign yCounter\_clear = (yCounter == (C\_VERT\_TOTAL\_COUNT-1));

/-\* Convert the xCounter/yCounter location from screen pixels (640x480) to our

\* local dots (320x240 or 160x120). Here we effectively divide x/y coordinate by 2 or 4,

\* depending on the resolution. \*-/

always @(\*)

begin

if (RESOLUTION == "320x240")

begin

x = xCounter[9:1];

y = yCounter[8:1];

end

else

begin

x = xCounter[9:2];

y = yCounter[8:2];

end

end

/-\* Change the (x,y) coordinate into a memory address. \*-/

vga\_address\_translator controller\_translator(

.x(x), .y(y), .mem\_address(memory\_address) );

defparam controller\_translator.RESOLUTION = RESOLUTION;

/-\* Generate the vertical and horizontal synchronization pulses. \*-/

always @(posedge vga\_clock)

begin

//- Sync Generator (ACTIVE LOW)

VGA\_HS1 <= ~((xCounter >= C\_HORZ\_SYNC\_START) && (xCounter <= C\_HORZ\_SYNC\_END));

VGA\_VS1 <= ~((yCounter >= C\_VERT\_SYNC\_START) && (yCounter <= C\_VERT\_SYNC\_END));

//- Current X and Y is valid pixel range

VGA\_BLANK1 <= ((xCounter < C\_HORZ\_NUM\_PIXELS) && (yCounter < C\_VERT\_NUM\_PIXELS));

//- Add 1 cycle delay

VGA\_HS <= VGA\_HS1;

VGA\_VS <= VGA\_VS1;

VGA\_BLANK <= VGA\_BLANK1;

end

/-\* VGA sync should be 1 at all times. \*-/

assign VGA\_SYNC = vcc;

/-\* Generate the VGA clock signal. \*-/

assign VGA\_CLK = vga\_clock;

/-\* Brighten the colour output. \*-/

// The colour input is first processed to brighten the image a little. Setting the top

// bits to correspond to the R,G,B colour makes the image a bit dull. To brighten the image,

// each bit of the colour is replicated through the 10 DAC colour input bits. For example,

// when BITS\_PER\_COLOUR\_CHANNEL is 2 and the red component is set to 2'b10, then the

// VGA\_R input to the DAC will be set to 10'b1010101010.

integer index;

integer sub\_index;

always @(pixel\_colour)

begin

VGA\_R <= 'b0;

VGA\_G <= 'b0;

VGA\_B <= 'b0;

if (MONOCHROME == "FALSE")

begin

for (index = 10-BITS\_PER\_COLOUR\_CHANNEL; index >= 0; index = index - BITS\_PER\_COLOUR\_CHANNEL)

begin

for (sub\_index = BITS\_PER\_COLOUR\_CHANNEL - 1; sub\_index >= 0; sub\_index = sub\_index - 1)

begin

VGA\_R[sub\_index+index] <= pixel\_colour[sub\_index + BITS\_PER\_COLOUR\_CHANNEL\*2];

VGA\_G[sub\_index+index] <= pixel\_colour[sub\_index + BITS\_PER\_COLOUR\_CHANNEL];

VGA\_B[sub\_index+index] <= pixel\_colour[sub\_index];

end

end

end

else

begin

for (index = 0; index < 10; index = index + 1)

begin

VGA\_R[index] <= pixel\_colour[0:0];

VGA\_G[index] <= pixel\_colour[0:0];

VGA\_B[index] <= pixel\_colour[0:0];

end

end

end

endmodule

/\* VGA Adapter

\* ----------------

\*

\* This is an implementation of a VGA Adapter. The adapter uses VGA mode signalling to initiate

\* a 640x480 resolution mode on a computer monitor, with a refresh rate of approximately 60Hz.

\* It is designed for easy use in an early digital logic design course to facilitate student

\* projects on the Altera DE2 Educational board.

\*

\* This implementation of the VGA adapter can display images of varying colour depth at a resolution of

\* 320x240 or 160x120 superpixels. The concept of superpixels is introduced to reduce the amount of on-chip

\* memory used by the adapter. The following table shows the number of bits of on-chip memory used by

\* the adapter in various resolutions and colour depths.

\*

\* -------------------------------------------------------------------------------------------------------------------------------

\* Resolution | Mono | 8 colours | 64 colours | 512 colours | 4096 colours | 32768 colours | 262144 colours | 2097152 colours |

\* -------------------------------------------------------------------------------------------------------------------------------

\* 160x120 | 19200 | 57600 | 115200 | 172800 | 230400 | 288000 | 345600 | 403200 |

\* 320x240 | 78600 | 230400 | ############## Does not fit ############################################################## |

\* -------------------------------------------------------------------------------------------------------------------------------

\*

\* By default the adapter works at the resolution of 320x240 with 8 colours. To set the adapter in any of

\* the other modes, the adapter must be instantiated with specific parameters. These parameters are:

\* - RESOLUTION - a string that should be either "320x240" or "160x120".

\* - MONOCHROME - a string that should be "TRUE" if you only want black and white colours, and "FALSE"

\* otherwise.

\* - BITS\_PER\_COLOUR\_CHANNEL - an integer specifying how many bits are available to describe each colour

\* (R,G,B). A default value of 1 indicates that 1 bit will be used for red

\* channel, 1 for green channel and 1 for blue channel. This allows 8 colours

\* to be used.

\*

\* In addition to the above parameters, a BACKGROUND\_IMAGE parameter can be specified. The parameter

\* refers to a memory initilization file (MIF) which contains the initial contents of video memory.

\* By specifying the initial contents of the memory we can force the adapter to initially display an

\* image of our choice. Please note that the image described by the BACKGROUND\_IMAGE file will only

\* be valid right after your program the DE2 board. If your circuit draws a single pixel on the screen,

\* the video memory will be altered and screen contents will be changed. In order to restore the background

\* image your circuti will have to redraw the background image pixel by pixel, or you will have to

\* reprogram the DE2 board, thus allowing the video memory to be rewritten.

\*

\* To use the module connect the vga\_adapter to your circuit. Your circuit should produce a value for

\* inputs X, Y and plot. When plot is high, at the next positive edge of the input clock the vga\_adapter

\* will change the contents of the video memory for the pixel at location (X,Y). At the next redraw

\* cycle the VGA controller will update the contants of the screen by reading the video memory and copying

\* it over to the screen. Since the monitor screen has no memory, the VGA controller has to copy the

\* contents of the video memory to the screen once every 60th of a second to keep the image stable. Thus,

\* the video memory should not be used for other purposes as it may interfere with the operation of the

\* VGA Adapter.

\*

\* As a final note, ensure that the following conditions are met when using this module:

\* 1. You are implementing the the VGA Adapter on the Altera DE2 board. Using another board may change

\* the amount of memory you can use, the clock generation mechanism, as well as pin assignments required

\* to properly drive the VGA digital-to-analog converter.

\* 2. Outputs VGA\_\* should exist in your top level design. They should be assigned pin locations on the

\* Altera DE2 board as specified by the DE2\_pin\_assignments.csv file.

\* 3. The input clock must have a frequency of 50 MHz with a 50% duty cycle. On the Altera DE2 board

\* PIN\_N2 is the source for the 50MHz clock.

\*

\* During compilation with Quartus II you may receive the following warnings:

\* - Warning: Variable or input pin "clocken1" is defined but never used

\* - Warning: Pin "VGA\_SYNC" stuck at VCC

\* - Warning: Found xx output pins without output pin load capacitance assignment

\* These warnings can be ignored. The first warning is generated, because the software generated

\* memory module contains an input called "clocken1" and it does not drive logic. The second warning

\* indicates that the VGA\_SYNC signal is always high. This is intentional. The final warning is

\* generated for the purposes of power analysis. It will persist unless the output pins are assigned

\* output capacitance. Leaving the capacitance values at 0 pf did not affect the operation of the module.

\*

\* If you see any other warnings relating to the vga\_adapter, be sure to examine them carefully. They may

\* cause your circuit to malfunction.

\*

\* NOTES/REVISIONS:

\* July 10, 2007 - Modified the original version of the VGA Adapter written by Sam Vafaee in 2006. The module

\* now supports 2 different resolutions as well as uses half the memory compared to prior

\* implementation. Also, all settings for the module can be specified from the point

\* of instantiation, rather than by modifying the source code. (Tomasz S. Czajkowski)

\*/

module vga\_adapter(

resetn,

clock,

colour,

x, y, plot,

/\* Signals for the DAC to drive the monitor. \*/

VGA\_R,

VGA\_G,

VGA\_B,

VGA\_HS,

VGA\_VS,

VGA\_BLANK,

VGA\_SYNC,

VGA\_CLK);

parameter BITS\_PER\_COLOUR\_CHANNEL = 1;

/\* The number of bits per colour channel used to represent the colour of each pixel. A value

\* of 1 means that Red, Green and Blue colour channels will use 1 bit each to represent the intensity

\* of the respective colour channel. For BITS\_PER\_COLOUR\_CHANNEL=1, the adapter can display 8 colours.

\* In general, the adapter is able to use 2^(3\*BITS\_PER\_COLOUR\_CHANNEL ) colours. The number of colours is

\* limited by the screen resolution and the amount of on-chip memory available on the target device.

\*/

parameter MONOCHROME = "FALSE";

/\* Set this parameter to "TRUE" if you only wish to use black and white colours. Doing so will reduce

\* the amount of memory you will use by a factor of 3. \*/

parameter RESOLUTION = "320x240";

/\* Set this parameter to "160x120" or "320x240". It will cause the VGA adapter to draw each dot on

\* the screen by using a block of 4x4 pixels ("160x120" resolution) or 2x2 pixels ("320x240" resolution).

\* It effectively reduces the screen resolution to an integer fraction of 640x480. It was necessary

\* to reduce the resolution for the Video Memory to fit within the on-chip memory limits.

\*/

parameter BACKGROUND\_IMAGE = "background.mif";

/\* The initial screen displayed when the circuit is first programmed onto the DE2 board can be

\* defined useing an MIF file. The file contains the initial colour for each pixel on the screen

\* and is placed in the Video Memory (VideoMemory module) upon programming. Note that resetting the

\* VGA Adapter will not cause the Video Memory to revert to the specified image. \*/

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

/\* Declare inputs and outputs. \*/

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

input resetn;

input clock;

/\* The colour input can be either 1 bit or 3\*BITS\_PER\_COLOUR\_CHANNEL bits wide, depending on

\* the setting of the MONOCHROME parameter.

\*/

input [((MONOCHROME == "TRUE") ? (0) : (BITS\_PER\_COLOUR\_CHANNEL\*3-1)):0] colour;

/\* Specify the number of bits required to represent an (X,Y) coordinate on the screen for

\* a given resolution.

\*/

input [((RESOLUTION == "320x240") ? (8) : (7)):0] x;

input [((RESOLUTION == "320x240") ? (7) : (6)):0] y;

/\* When plot is high then at the next positive edge of the clock the pixel at (x,y) will change to

\* a new colour, defined by the value of the colour input.

\*/

input plot;

/\* These outputs drive the VGA display. The VGA\_CLK is also used to clock the FSM responsible for

\* controlling the data transferred to the DAC driving the monitor. \*/

output [7:0] VGA\_R;

output [7:0] VGA\_G;

output [7:0] VGA\_B;

output VGA\_HS;

output VGA\_VS;

output VGA\_BLANK;

output VGA\_SYNC;

output VGA\_CLK;

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

/\* Declare local signals here. \*/

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

wire valid\_160x120;

wire valid\_320x240;

/\* Set to 1 if the specified coordinates are in a valid range for a given resolution.\*/

wire writeEn;

/\* This is a local signal that allows the Video Memory contents to be changed.

\* It depends on the screen resolution, the values of X and Y inputs, as well as

\* the state of the plot signal.

\*/

wire [((MONOCHROME == "TRUE") ? (0) : (BITS\_PER\_COLOUR\_CHANNEL\*3-1)):0] to\_ctrl\_colour;

/\* Pixel colour read by the VGA controller \*/

wire [((RESOLUTION == "320x240") ? (16) : (14)):0] user\_to\_video\_memory\_addr;

/\* This bus specifies the address in memory the user must write

\* data to in order for the pixel intended to appear at location (X,Y) to be displayed

\* at the correct location on the screen.

\*/

wire [((RESOLUTION == "320x240") ? (16) : (14)):0] controller\_to\_video\_memory\_addr;

/\* This bus specifies the address in memory the vga controller must read data from

\* in order to determine the colour of a pixel located at coordinate (X,Y) of the screen.

\*/

wire clock\_25;

/\* 25MHz clock generated by dividing the input clock frequency by 2. \*/

wire vcc, gnd;

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

/\* Instances of modules for the VGA adapter. \*/

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

assign vcc = 1'b1;

assign gnd = 1'b0;

vga\_address\_translator user\_input\_translator(

.x(x), .y(y), .mem\_address(user\_to\_video\_memory\_addr) );

defparam user\_input\_translator.RESOLUTION = RESOLUTION;

/\* Convert user coordinates into a memory address. \*/

assign valid\_160x120 = (({1'b0, x} >= 0) & ({1'b0, x} < 160) & ({1'b0, y} >= 0) & ({1'b0, y} < 120)) & (RESOLUTION == "160x120");

assign valid\_320x240 = (({1'b0, x} >= 0) & ({1'b0, x} < 320) & ({1'b0, y} >= 0) & ({1'b0, y} < 240)) & (RESOLUTION == "320x240");

assign writeEn = (plot) & (valid\_160x120 | valid\_320x240);

/\* Allow the user to plot a pixel if and only if the (X,Y) coordinates supplied are in a valid range. \*/

/\* Create video memory. \*/

altsyncram VideoMemory (

.wren\_a (writeEn),

.wren\_b (gnd),

.clock0 (clock), // write clock

.clock1 (clock\_25), // read clock

.clocken0 (vcc), // write enable clock

.clocken1 (vcc), // read enable clock

.address\_a (user\_to\_video\_memory\_addr),

.address\_b (controller\_to\_video\_memory\_addr),

.data\_a (colour), // data in

.q\_b (to\_ctrl\_colour) // data out

);

defparam

VideoMemory.WIDTH\_A = ((MONOCHROME == "FALSE") ? (BITS\_PER\_COLOUR\_CHANNEL\*3) : 1),

VideoMemory.WIDTH\_B = ((MONOCHROME == "FALSE") ? (BITS\_PER\_COLOUR\_CHANNEL\*3) : 1),

VideoMemory.INTENDED\_DEVICE\_FAMILY = "Cyclone II",

VideoMemory.OPERATION\_MODE = "DUAL\_PORT",

VideoMemory.WIDTHAD\_A = ((RESOLUTION == "320x240") ? (17) : (15)),

VideoMemory.NUMWORDS\_A = ((RESOLUTION == "320x240") ? (76800) : (19200)),

VideoMemory.WIDTHAD\_B = ((RESOLUTION == "320x240") ? (17) : (15)),

VideoMemory.NUMWORDS\_B = ((RESOLUTION == "320x240") ? (76800) : (19200)),

VideoMemory.OUTDATA\_REG\_B = "CLOCK1",

VideoMemory.ADDRESS\_REG\_B = "CLOCK1",

VideoMemory.CLOCK\_ENABLE\_INPUT\_A = "BYPASS",

VideoMemory.CLOCK\_ENABLE\_INPUT\_B = "BYPASS",

VideoMemory.CLOCK\_ENABLE\_OUTPUT\_B = "BYPASS",

VideoMemory.POWER\_UP\_UNINITIALIZED = "FALSE",

VideoMemory.INIT\_FILE = BACKGROUND\_IMAGE;

vga\_pll mypll(clock, clock\_25);

/\* This module generates a clock with half the frequency of the input clock.

\* For the VGA adapter to operate correctly the clock signal 'clock' must be

\* a 50MHz clock. The derived clock, which will then operate at 25MHz, is

\* required to set the monitor into the 640x480@60Hz display mode (also known as

\* the VGA mode).

\*/

wire [9:0] r;

wire [9:0] g;

wire [9:0] b;

/\* Assign the MSBs from the controller to the VGA signals \*/

assign VGA\_R = r[9:2];

assign VGA\_G = g[9:2];

assign VGA\_B = b[9:2];

vga\_controller controller(

.vga\_clock(clock\_25),

.resetn(resetn),

.pixel\_colour(to\_ctrl\_colour),

.memory\_address(controller\_to\_video\_memory\_addr),

.VGA\_R(r),

.VGA\_G(g),

.VGA\_B(b),

.VGA\_HS(VGA\_HS),

.VGA\_VS(VGA\_VS),

.VGA\_BLANK(VGA\_BLANK),

.VGA\_SYNC(VGA\_SYNC),

.VGA\_CLK(VGA\_CLK)

);

defparam controller.BITS\_PER\_COLOUR\_CHANNEL = BITS\_PER\_COLOUR\_CHANNEL ;

defparam controller.MONOCHROME = MONOCHROME;

defparam controller.RESOLUTION = RESOLUTION;

endmodule

// megafunction wizard: %ALTPLL%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altpll

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

// File Name: VgaPll.v

// Megafunction Name(s):

// altpll

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

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

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

//

// 5.0 Build 168 06/22/2005 SP 1 SJ Full Version

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

//Copyright (C) 1991-2005 Altera Corporation

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

//and other software and tools, and its AMPP partner logic

//functions, and any output files 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 Altera Program License

//Subscription Agreement, Altera MegaCore Function License

//Agreement, or other applicable license agreement, including,

//without limitation, that your use is for the sole purpose of

//programming logic devices manufactured by Altera and sold by

//Altera 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 vga\_pll (

clock\_in,

clock\_out);

input clock\_in;

output clock\_out;

wire [5:0] clock\_output\_bus;

wire [1:0] clock\_input\_bus;

wire gnd;

assign gnd = 1'b0;

assign clock\_input\_bus = { gnd, clock\_in };

altpll altpll\_component (

.inclk (clock\_input\_bus),

.clk (clock\_output\_bus)

);

defparam

altpll\_component.operation\_mode = "NORMAL",

altpll\_component.intended\_device\_family = "Cyclone II",

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

altpll\_component.pll\_type = "FAST",

/\* Specify the input clock to be a 50MHz clock. A 50 MHz clock is present

\* on PIN\_N2 on the DE2 board. We need to specify the input clock frequency

\* in order to set up the PLL correctly. To do this we must put the input clock

\* period measured in picoseconds in the inclk0\_input\_frequency parameter.

\* 1/(20000 ps) = 0.5 \* 10^(5) Hz = 50 \* 10^(6) Hz = 50 MHz. \*/

altpll\_component.inclk0\_input\_frequency = 20000,

altpll\_component.primary\_clock = "INCLK0",

/\* Specify output clock parameters. The output clock should have a

\* frequency of 25 MHz, with 50% duty cycle. \*/

altpll\_component.compensate\_clock = "CLK0",

altpll\_component.clk0\_phase\_shift = "0",

altpll\_component.clk0\_divide\_by = 2,

altpll\_component.clk0\_multiply\_by = 1,

altpll\_component.clk0\_duty\_cycle = 50;

assign clock\_out = clock\_output\_bus[0];

endmodule

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

// CNX file retrieval info

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

// Retrieval info: PRIVATE: MIRROR\_CLK0 STRING "0"

// Retrieval info: PRIVATE: PHASE\_SHIFT\_UNIT0 STRING "deg"

// Retrieval info: PRIVATE: OUTPUT\_FREQ\_UNIT0 STRING "MHz"

// Retrieval info: PRIVATE: INCLK1\_FREQ\_UNIT\_COMBO STRING "MHz"

// Retrieval info: PRIVATE: SPREAD\_USE STRING "0"

// Retrieval info: PRIVATE: SPREAD\_FEATURE\_ENABLED STRING "0"

// Retrieval info: PRIVATE: GLOCKED\_COUNTER\_EDIT\_CHANGED STRING "1"

// Retrieval info: PRIVATE: GLOCK\_COUNTER\_EDIT NUMERIC "1048575"

// Retrieval info: PRIVATE: SRC\_SYNCH\_COMP\_RADIO STRING "0"

// Retrieval info: PRIVATE: DUTY\_CYCLE0 STRING "50.00000000"

// Retrieval info: PRIVATE: PHASE\_SHIFT0 STRING "0.00000000"

// Retrieval info: PRIVATE: MULT\_FACTOR0 NUMERIC "1"

// Retrieval info: PRIVATE: OUTPUT\_FREQ\_MODE0 STRING "1"

// Retrieval info: PRIVATE: SPREAD\_PERCENT STRING "0.500"

// Retrieval info: PRIVATE: LOCKED\_OUTPUT\_CHECK STRING "0"

// Retrieval info: PRIVATE: PLL\_ARESET\_CHECK STRING "0"

// Retrieval info: PRIVATE: STICKY\_CLK0 STRING "1"

// Retrieval info: PRIVATE: BANDWIDTH STRING "1.000"

// Retrieval info: PRIVATE: BANDWIDTH\_USE\_CUSTOM STRING "0"

// Retrieval info: PRIVATE: DEVICE\_SPEED\_GRADE STRING "Any"

// Retrieval info: PRIVATE: SPREAD\_FREQ STRING "50.000"

// Retrieval info: PRIVATE: BANDWIDTH\_FEATURE\_ENABLED STRING "0"

// Retrieval info: PRIVATE: LONG\_SCAN\_RADIO STRING "1"

// Retrieval info: PRIVATE: PLL\_ENHPLL\_CHECK NUMERIC "0"

// Retrieval info: PRIVATE: LVDS\_MODE\_DATA\_RATE\_DIRTY NUMERIC "0"

// Retrieval info: PRIVATE: USE\_CLK0 STRING "1"

// Retrieval info: PRIVATE: INCLK1\_FREQ\_EDIT\_CHANGED STRING "1"

// Retrieval info: PRIVATE: SCAN\_FEATURE\_ENABLED STRING "0"

// Retrieval info: PRIVATE: ZERO\_DELAY\_RADIO STRING "0"

// Retrieval info: PRIVATE: PLL\_PFDENA\_CHECK STRING "0"

// Retrieval info: PRIVATE: CREATE\_CLKBAD\_CHECK STRING "0"

// Retrieval info: PRIVATE: INCLK1\_FREQ\_EDIT STRING "50.000"

// Retrieval info: PRIVATE: CUR\_DEDICATED\_CLK STRING "c0"

// Retrieval info: PRIVATE: PLL\_FASTPLL\_CHECK NUMERIC "0"

// Retrieval info: PRIVATE: ACTIVECLK\_CHECK STRING "0"

// Retrieval info: PRIVATE: BANDWIDTH\_FREQ\_UNIT STRING "MHz"

// Retrieval info: PRIVATE: INCLK0\_FREQ\_UNIT\_COMBO STRING "MHz"

// Retrieval info: PRIVATE: GLOCKED\_MODE\_CHECK STRING "0"

// Retrieval info: PRIVATE: NORMAL\_MODE\_RADIO STRING "1"

// Retrieval info: PRIVATE: CUR\_FBIN\_CLK STRING "e0"

// Retrieval info: PRIVATE: DIV\_FACTOR0 NUMERIC "1"

// Retrieval info: PRIVATE: INCLK1\_FREQ\_UNIT\_CHANGED STRING "1"

// Retrieval info: PRIVATE: HAS\_MANUAL\_SWITCHOVER STRING "1"

// Retrieval info: PRIVATE: EXT\_FEEDBACK\_RADIO STRING "0"

// Retrieval info: PRIVATE: PLL\_AUTOPLL\_CHECK NUMERIC "1"

// Retrieval info: PRIVATE: CLKLOSS\_CHECK STRING "0"

// Retrieval info: PRIVATE: BANDWIDTH\_USE\_AUTO STRING "1"

// Retrieval info: PRIVATE: SHORT\_SCAN\_RADIO STRING "0"

// Retrieval info: PRIVATE: LVDS\_MODE\_DATA\_RATE STRING "Not Available"

// Retrieval info: PRIVATE: CLKSWITCH\_CHECK STRING "1"

// Retrieval info: PRIVATE: SPREAD\_FREQ\_UNIT STRING "KHz"

// Retrieval info: PRIVATE: PLL\_ENA\_CHECK STRING "0"

// Retrieval info: PRIVATE: INCLK0\_FREQ\_EDIT STRING "50.000"

// Retrieval info: PRIVATE: CNX\_NO\_COMPENSATE\_RADIO STRING "0"

// Retrieval info: PRIVATE: INT\_FEEDBACK\_\_MODE\_RADIO STRING "1"

// Retrieval info: PRIVATE: OUTPUT\_FREQ0 STRING "25.000"

// Retrieval info: PRIVATE: PRIMARY\_CLK\_COMBO STRING "inclk0"

// Retrieval info: PRIVATE: CREATE\_INCLK1\_CHECK STRING "0"

// Retrieval info: PRIVATE: SACN\_INPUTS\_CHECK STRING "0"

// Retrieval info: PRIVATE: DEV\_FAMILY STRING "Cyclone II"

// Retrieval info: PRIVATE: SWITCHOVER\_COUNT\_EDIT NUMERIC "1"

// Retrieval info: PRIVATE: SWITCHOVER\_FEATURE\_ENABLED STRING "1"

// Retrieval info: PRIVATE: BANDWIDTH\_PRESET STRING "Low"

// Retrieval info: PRIVATE: GLOCKED\_FEATURE\_ENABLED STRING "1"

// Retrieval info: PRIVATE: USE\_CLKENA0 STRING "0"

// Retrieval info: PRIVATE: LVDS\_PHASE\_SHIFT\_UNIT0 STRING "deg"

// Retrieval info: PRIVATE: CLKBAD\_SWITCHOVER\_CHECK STRING "0"

// Retrieval info: PRIVATE: BANDWIDTH\_USE\_PRESET STRING "0"

// Retrieval info: PRIVATE: PLL\_LVDS\_PLL\_CHECK NUMERIC "0"

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

// Retrieval info: CONSTANT: CLK0\_DUTY\_CYCLE NUMERIC "50"

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

// Retrieval info: CONSTANT: CLK0\_MULTIPLY\_BY NUMERIC "1"

// Retrieval info: CONSTANT: INCLK0\_INPUT\_FREQUENCY NUMERIC "20000"

// Retrieval info: CONSTANT: CLK0\_DIVIDE\_BY NUMERIC "2"

// Retrieval info: CONSTANT: PLL\_TYPE STRING "FAST"

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

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "NORMAL"

// Retrieval info: CONSTANT: COMPENSATE\_CLOCK STRING "CLK0"

// Retrieval info: CONSTANT: CLK0\_PHASE\_SHIFT STRING "0"

// Retrieval info: USED\_PORT: c0 0 0 0 0 OUTPUT VCC "c0"

// Retrieval info: USED\_PORT: @clk 0 0 6 0 OUTPUT VCC "@clk[5..0]"

// Retrieval info: USED\_PORT: inclk0 0 0 0 0 INPUT GND "inclk0"

// Retrieval info: USED\_PORT: @extclk 0 0 4 0 OUTPUT VCC "@extclk[3..0]"

// Retrieval info: CONNECT: @inclk 0 0 1 0 inclk0 0 0 0 0

// Retrieval info: CONNECT: c0 0 0 0 0 @clk 0 0 1 0

// Retrieval info: CONNECT: @inclk 0 0 1 1 GND 0 0 0 0

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

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

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

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

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

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

/\* This module converts a user specified coordinates into a memory address.

\* The output of the module depends on the resolution set by the user.

\*/

module vga\_address\_translator(x, y, mem\_address);

parameter RESOLUTION = "160x120";

/\* Set this parameter to "160x120" or "320x240". It will cause the VGA adapter to draw each dot on

\* the screen by using a block of 4x4 pixels ("160x120" resolution) or 2x2 pixels ("320x240" resolution).

\* It effectively reduces the screen resolution to an integer fraction of 640x480. It was necessary

\* to reduce the resolution for the Video Memory to fit within the on-chip memory limits.

\*/

input [((RESOLUTION == "320x240") ? (8) : (7)):0] x;

input [((RESOLUTION == "320x240") ? (7) : (6)):0] y;

output reg [((RESOLUTION == "320x240") ? (16) : (14)):0] mem\_address;

/\* The basic formula is address = y\*WIDTH + x;

\* For 320x240 resolution we can write 320 as (256 + 64). Memory address becomes

\* (y\*256) + (y\*64) + x;

\* This simplifies multiplication a simple shift and add operation.

\* A leading 0 bit is added to each operand to ensure that they are treated as unsigned

\* inputs. By default the use a '+' operator will generate a signed adder.

\* Similarly, for 160x120 resolution we write 160 as 128+32.

\*/

wire [16:0] res\_320x240 = ({1'b0, y, 8'd0} + {1'b0, y, 6'd0} + {1'b0, x});

wire [15:0] res\_160x120 = ({1'b0, y, 7'd0} + {1'b0, y, 5'd0} + {1'b0, x});

always @(\*)

begin

if (RESOLUTION == "320x240")

mem\_address = res\_320x240;

else

mem\_address = res\_160x120[14:0];

end

endmodule

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: startscreen\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 startscreen\_rom (

address,

clock,

q);

input [14:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../start\_screen.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 32768,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 15,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../start\_screen.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "32768"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "15"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../start\_screen.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "32768"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "15"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 15 0 INPUT NODEFVAL "address[14..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 15 0 address 0 0 15 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

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

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

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

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

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

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

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: background\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 background\_rom (

address,

clock,

q);

input [14:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../background.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 32768,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 15,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../background.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "32768"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "15"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../background.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "32768"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "15"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 15 0 INPUT NODEFVAL "address[14..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 15 0 address 0 0 15 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

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

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

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

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

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

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

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite1\_idle\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite1\_idle\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite1\_idle.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite1\_idle.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite1\_idle.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_idle\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_idle\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_idle\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_idle\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_idle\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_idle\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite1\_jumping\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite1\_jumping\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite1\_jumping.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite1\_jumping.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite1\_jumping.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumping\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumping\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumping\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumping\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumping\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumping\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite1\_jumpkicking\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite1\_jumpkicking\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite1\_jumpkicking.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite1\_jumpkicking.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite1\_jumpkicking.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumpkicking\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumpkicking\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumpkicking\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumpkicking\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumpkicking\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_jumpkicking\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite1\_kicking\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite1\_kicking\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite1\_kicking.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite1\_kicking.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite1\_kicking.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_kicking\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_kicking\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_kicking\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_kicking\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_kicking\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_kicking\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite1\_punching\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite1\_punching\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite1\_punching.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite1\_punching.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite1\_punching.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_punching\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_punching\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_punching\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_punching\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_punching\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_punching\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite1\_sweeping\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite1\_sweeping\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite1\_sweeping.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite1\_sweeping.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite1\_sweeping.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_sweeping\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_sweeping\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_sweeping\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_sweeping\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_sweeping\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_sweeping\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite1\_crouching\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite1\_crouching\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite1\_crouching.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite1\_crouching.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite1\_crouching.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_crouching\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_crouching\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_crouching\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_crouching\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_crouching\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite1\_crouching\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite2\_idle\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite2\_idle\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite2\_idle.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite2\_idle.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite2\_idle.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_idle\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_idle\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_idle\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_idle\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_idle\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_idle\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite2\_jumping\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite2\_jumping\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite2\_jumping.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite2\_jumping.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite2\_jumping.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumping\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumping\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumping\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumping\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumping\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumping\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite2\_jumpkicking\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite2\_jumpkicking\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite2\_jumpkicking.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite2\_jumpkicking.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite2\_jumpkicking.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumpkicking\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumpkicking\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumpkicking\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumpkicking\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumpkicking\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_jumpkicking\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite2\_kicking\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite2\_kicking\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite2\_kicking.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite2\_kicking.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite2\_kicking.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_kicking\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_kicking\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_kicking\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_kicking\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_kicking\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_kicking\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite2\_punching\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite2\_punching\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite2\_punching.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite2\_punching.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite2\_punching.mif"

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

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

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

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

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

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

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

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

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

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

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

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_punching\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_punching\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_punching\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_punching\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_punching\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_punching\_rom\_bb.v FALSE

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

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

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

// File Name: sprite2\_sweeping\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

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

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

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

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

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

//Copyright (C) 2018 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 sprite2\_sweeping\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

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

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

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

altsyncram\_component.init\_file = "../sprite2\_sweeping.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

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

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

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

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

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

endmodule

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

// CNX file retrieval info

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

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

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

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

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

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

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

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

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

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

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

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

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

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

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

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

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

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

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite2\_sweeping.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

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

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

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

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

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

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

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

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

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

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

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

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

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite2\_sweeping.mif"

// Retrieval info: CONSTANT: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

// Retrieval info: CONSTANT: LPM\_TYPE STRING "altsyncram"

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

// Retrieval info: CONSTANT: OUTDATA\_ACLR\_A STRING "NONE"

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

// Retrieval info: CONSTANT: WIDTH\_BYTEENA\_A NUMERIC "1"

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

// Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT VCC "clock"

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_sweeping\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_sweeping\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_sweeping\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_sweeping\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_sweeping\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_sweeping\_rom\_bb.v FALSE

// Retrieval info: LIB\_FILE: altera\_mf

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

// ============================================================

// File Name: sprite2\_crouching\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

// ============================================================

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Copyright (C) 2018 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 sprite2\_crouching\_rom (

address,

clock,

q);

input [12:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

altsyncram\_component.address\_aclr\_a = "NONE",

altsyncram\_component.clock\_enable\_input\_a = "BYPASS",

altsyncram\_component.clock\_enable\_output\_a = "BYPASS",

altsyncram\_component.init\_file = "../sprite2\_crouching.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

altsyncram\_component.lpm\_type = "altsyncram",

altsyncram\_component.numwords\_a = 8192,

altsyncram\_component.operation\_mode = "ROM",

altsyncram\_component.outdata\_aclr\_a = "NONE",

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 13,

altsyncram\_component.width\_a = 9,

altsyncram\_component.width\_byteena\_a = 1;

endmodule

// ============================================================

// CNX file retrieval info

// ============================================================

// Retrieval info: PRIVATE: ADDRESSSTALL\_A NUMERIC "0"

// Retrieval info: PRIVATE: AclrAddr NUMERIC "0"

// Retrieval info: PRIVATE: AclrByte NUMERIC "0"

// Retrieval info: PRIVATE: AclrOutput NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_ENABLE NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

// Retrieval info: PRIVATE: BlankMemory NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_INPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_OUTPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: Clken NUMERIC "0"

// Retrieval info: PRIVATE: IMPLEMENT\_IN\_LES NUMERIC "0"

// Retrieval info: PRIVATE: INIT\_FILE\_LAYOUT STRING "PORT\_A"

// Retrieval info: PRIVATE: INIT\_TO\_SIM\_X NUMERIC "0"

// Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"

// Retrieval info: PRIVATE: JTAG\_ENABLED NUMERIC "0"

// Retrieval info: PRIVATE: JTAG\_ID STRING "NONE"

// Retrieval info: PRIVATE: MAXIMUM\_DEPTH NUMERIC "0"

// Retrieval info: PRIVATE: MIFfilename STRING "../sprite2\_crouching.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: PRIVATE: RAM\_BLOCK\_TYPE NUMERIC "0"

// Retrieval info: PRIVATE: RegAddr NUMERIC "1"

// Retrieval info: PRIVATE: RegOutput NUMERIC "0"

// Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "0"

// Retrieval info: PRIVATE: SingleClock NUMERIC "1"

// Retrieval info: PRIVATE: UseDQRAM NUMERIC "0"

// Retrieval info: PRIVATE: WidthAddr NUMERIC "13"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

// Retrieval info: PRIVATE: rden NUMERIC "0"

// Retrieval info: LIBRARY: altera\_mf altera\_mf.altera\_mf\_components.all

// Retrieval info: CONSTANT: ADDRESS\_ACLR\_A STRING "NONE"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_INPUT\_A STRING "BYPASS"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_OUTPUT\_A STRING "BYPASS"

// Retrieval info: CONSTANT: INIT\_FILE STRING "../sprite2\_crouching.mif"

// Retrieval info: CONSTANT: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

// Retrieval info: CONSTANT: LPM\_TYPE STRING "altsyncram"

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "8192"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

// Retrieval info: CONSTANT: OUTDATA\_ACLR\_A STRING "NONE"

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "13"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

// Retrieval info: CONSTANT: WIDTH\_BYTEENA\_A NUMERIC "1"

// Retrieval info: USED\_PORT: address 0 0 13 0 INPUT NODEFVAL "address[12..0]"

// Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT VCC "clock"

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 13 0 address 0 0 13 0

// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_crouching\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_crouching\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_crouching\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_crouching\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_crouching\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL sprite2\_crouching\_rom\_bb.v FALSE

// Retrieval info: LIB\_FILE: altera\_mf

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

// ============================================================

// File Name: p1\_win\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

// ============================================================

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Copyright (C) 2018 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 p1\_win\_rom (

address,

clock,

q);

input [14:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

altsyncram\_component.address\_aclr\_a = "NONE",

altsyncram\_component.clock\_enable\_input\_a = "BYPASS",

altsyncram\_component.clock\_enable\_output\_a = "BYPASS",

altsyncram\_component.init\_file = "../p1\_win.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

altsyncram\_component.lpm\_type = "altsyncram",

altsyncram\_component.numwords\_a = 32768,

altsyncram\_component.operation\_mode = "ROM",

altsyncram\_component.outdata\_aclr\_a = "NONE",

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 15,

altsyncram\_component.width\_a = 9,

altsyncram\_component.width\_byteena\_a = 1;

endmodule

// ============================================================

// CNX file retrieval info

// ============================================================

// Retrieval info: PRIVATE: ADDRESSSTALL\_A NUMERIC "0"

// Retrieval info: PRIVATE: AclrAddr NUMERIC "0"

// Retrieval info: PRIVATE: AclrByte NUMERIC "0"

// Retrieval info: PRIVATE: AclrOutput NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_ENABLE NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

// Retrieval info: PRIVATE: BlankMemory NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_INPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_OUTPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: Clken NUMERIC "0"

// Retrieval info: PRIVATE: IMPLEMENT\_IN\_LES NUMERIC "0"

// Retrieval info: PRIVATE: INIT\_FILE\_LAYOUT STRING "PORT\_A"

// Retrieval info: PRIVATE: INIT\_TO\_SIM\_X NUMERIC "0"

// Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"

// Retrieval info: PRIVATE: JTAG\_ENABLED NUMERIC "0"

// Retrieval info: PRIVATE: JTAG\_ID STRING "NONE"

// Retrieval info: PRIVATE: MAXIMUM\_DEPTH NUMERIC "0"

// Retrieval info: PRIVATE: MIFfilename STRING "../p1\_win.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "32768"

// Retrieval info: PRIVATE: RAM\_BLOCK\_TYPE NUMERIC "0"

// Retrieval info: PRIVATE: RegAddr NUMERIC "1"

// Retrieval info: PRIVATE: RegOutput NUMERIC "0"

// Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "0"

// Retrieval info: PRIVATE: SingleClock NUMERIC "1"

// Retrieval info: PRIVATE: UseDQRAM NUMERIC "0"

// Retrieval info: PRIVATE: WidthAddr NUMERIC "15"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

// Retrieval info: PRIVATE: rden NUMERIC "0"

// Retrieval info: LIBRARY: altera\_mf altera\_mf.altera\_mf\_components.all

// Retrieval info: CONSTANT: ADDRESS\_ACLR\_A STRING "NONE"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_INPUT\_A STRING "BYPASS"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_OUTPUT\_A STRING "BYPASS"

// Retrieval info: CONSTANT: INIT\_FILE STRING "../p1\_win.mif"

// Retrieval info: CONSTANT: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

// Retrieval info: CONSTANT: LPM\_TYPE STRING "altsyncram"

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "32768"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

// Retrieval info: CONSTANT: OUTDATA\_ACLR\_A STRING "NONE"

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "15"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

// Retrieval info: CONSTANT: WIDTH\_BYTEENA\_A NUMERIC "1"

// Retrieval info: USED\_PORT: address 0 0 15 0 INPUT NODEFVAL "address[14..0]"

// Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT VCC "clock"

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 15 0 address 0 0 15 0

// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p1\_win\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p1\_win\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p1\_win\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p1\_win\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p1\_win\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p1\_win\_rom\_bb.v FALSE

// Retrieval info: LIB\_FILE: altera\_mf

// megafunction wizard: %ROM: 1-PORT%

// GENERATION: STANDARD

// VERSION: WM1.0

// MODULE: altsyncram

// ============================================================

// File Name: p2\_win\_rom.v

// Megafunction Name(s):

// altsyncram

//

// Simulation Library Files(s):

// altera\_mf

// ============================================================

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!

//

// 18.0.0 Build 614 04/24/2018 SJ Standard Edition

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Copyright (C) 2018 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 p2\_win\_rom (

address,

clock,

q);

input [14:0] address;

input clock;

output [8:0] q;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_off

`endif

tri1 clock;

`ifndef ALTERA\_RESERVED\_QIS

// synopsys translate\_on

`endif

wire [8:0] sub\_wire0;

wire [8:0] q = sub\_wire0[8:0];

altsyncram altsyncram\_component (

.address\_a (address),

.clock0 (clock),

.q\_a (sub\_wire0),

.aclr0 (1'b0),

.aclr1 (1'b0),

.address\_b (1'b1),

.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\_a ({9{1'b1}}),

.data\_b (1'b1),

.eccstatus (),

.q\_b (),

.rden\_a (1'b1),

.rden\_b (1'b1),

.wren\_a (1'b0),

.wren\_b (1'b0));

defparam

altsyncram\_component.address\_aclr\_a = "NONE",

altsyncram\_component.clock\_enable\_input\_a = "BYPASS",

altsyncram\_component.clock\_enable\_output\_a = "BYPASS",

altsyncram\_component.init\_file = "../p2\_win.mif",

altsyncram\_component.intended\_device\_family = "Cyclone V",

altsyncram\_component.lpm\_hint = "ENABLE\_RUNTIME\_MOD=NO",

altsyncram\_component.lpm\_type = "altsyncram",

altsyncram\_component.numwords\_a = 32768,

altsyncram\_component.operation\_mode = "ROM",

altsyncram\_component.outdata\_aclr\_a = "NONE",

altsyncram\_component.outdata\_reg\_a = "UNREGISTERED",

altsyncram\_component.widthad\_a = 15,

altsyncram\_component.width\_a = 9,

altsyncram\_component.width\_byteena\_a = 1;

endmodule

// ============================================================

// CNX file retrieval info

// ============================================================

// Retrieval info: PRIVATE: ADDRESSSTALL\_A NUMERIC "0"

// Retrieval info: PRIVATE: AclrAddr NUMERIC "0"

// Retrieval info: PRIVATE: AclrByte NUMERIC "0"

// Retrieval info: PRIVATE: AclrOutput NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_ENABLE NUMERIC "0"

// Retrieval info: PRIVATE: BYTE\_SIZE NUMERIC "9"

// Retrieval info: PRIVATE: BlankMemory NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_INPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: CLOCK\_ENABLE\_OUTPUT\_A NUMERIC "0"

// Retrieval info: PRIVATE: Clken NUMERIC "0"

// Retrieval info: PRIVATE: IMPLEMENT\_IN\_LES NUMERIC "0"

// Retrieval info: PRIVATE: INIT\_FILE\_LAYOUT STRING "PORT\_A"

// Retrieval info: PRIVATE: INIT\_TO\_SIM\_X NUMERIC "0"

// Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"

// Retrieval info: PRIVATE: JTAG\_ENABLED NUMERIC "0"

// Retrieval info: PRIVATE: JTAG\_ID STRING "NONE"

// Retrieval info: PRIVATE: MAXIMUM\_DEPTH NUMERIC "0"

// Retrieval info: PRIVATE: MIFfilename STRING "../p2\_win.mif"

// Retrieval info: PRIVATE: NUMWORDS\_A NUMERIC "32768"

// Retrieval info: PRIVATE: RAM\_BLOCK\_TYPE NUMERIC "0"

// Retrieval info: PRIVATE: RegAddr NUMERIC "1"

// Retrieval info: PRIVATE: RegOutput NUMERIC "0"

// Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "0"

// Retrieval info: PRIVATE: SingleClock NUMERIC "1"

// Retrieval info: PRIVATE: UseDQRAM NUMERIC "0"

// Retrieval info: PRIVATE: WidthAddr NUMERIC "15"

// Retrieval info: PRIVATE: WidthData NUMERIC "9"

// Retrieval info: PRIVATE: rden NUMERIC "0"

// Retrieval info: LIBRARY: altera\_mf altera\_mf.altera\_mf\_components.all

// Retrieval info: CONSTANT: ADDRESS\_ACLR\_A STRING "NONE"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_INPUT\_A STRING "BYPASS"

// Retrieval info: CONSTANT: CLOCK\_ENABLE\_OUTPUT\_A STRING "BYPASS"

// Retrieval info: CONSTANT: INIT\_FILE STRING "../p2\_win.mif"

// Retrieval info: CONSTANT: INTENDED\_DEVICE\_FAMILY STRING "Cyclone V"

// Retrieval info: CONSTANT: LPM\_HINT STRING "ENABLE\_RUNTIME\_MOD=NO"

// Retrieval info: CONSTANT: LPM\_TYPE STRING "altsyncram"

// Retrieval info: CONSTANT: NUMWORDS\_A NUMERIC "32768"

// Retrieval info: CONSTANT: OPERATION\_MODE STRING "ROM"

// Retrieval info: CONSTANT: OUTDATA\_ACLR\_A STRING "NONE"

// Retrieval info: CONSTANT: OUTDATA\_REG\_A STRING "UNREGISTERED"

// Retrieval info: CONSTANT: WIDTHAD\_A NUMERIC "15"

// Retrieval info: CONSTANT: WIDTH\_A NUMERIC "9"

// Retrieval info: CONSTANT: WIDTH\_BYTEENA\_A NUMERIC "1"

// Retrieval info: USED\_PORT: address 0 0 15 0 INPUT NODEFVAL "address[14..0]"

// Retrieval info: USED\_PORT: clock 0 0 0 0 INPUT VCC "clock"

// Retrieval info: USED\_PORT: q 0 0 9 0 OUTPUT NODEFVAL "q[8..0]"

// Retrieval info: CONNECT: @address\_a 0 0 15 0 address 0 0 15 0

// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0

// Retrieval info: CONNECT: q 0 0 9 0 @q\_a 0 0 9 0

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p2\_win\_rom.v TRUE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p2\_win\_rom.inc FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p2\_win\_rom.cmp FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p2\_win\_rom.bsf FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p2\_win\_rom\_inst.v FALSE

// Retrieval info: GEN\_FILE: TYPE\_NORMAL p2\_win\_rom\_bb.v FALSE

// Retrieval info: LIB\_FILE: altera\_mf

/\* simulation.do\*/

vlib work

vlog buttons\_FSM.v

vsim buttons\_FSM

log {/\*}

add wave {/\*}

force clk 1 0ns, 0 {5ns} -r 10ns

force resetn 0

force key\_pressed 1

run 10ns

force resetn 1

run 90ns

force key\_pressed 0

run 100ns

force key\_pressed 1

run 100ns