ECE241
Road to Heaven

Boyuan Shao 1004871882
Sharon Li 1004947800

1.0 Introduction

The game *"Road to Heaven"* is realized by successful integration of Verilog coding, FPGA board, VGA display, sound sensor and pixel graphic design. Below is a brief description of the game.

A character is standing at the right side of the screen, and cars are coming from the left side of the screen towards the character. In order to continue the game, the player must ensure that as soon as the car is about to hit the character, the player makes a loud enough sound to be received by the sound sensor, and thus instruct the character to make a jump. Else the character is going to be hit by the car and the game will be over.

The motivation of the game is to provide an easy, non-addicting platform for stressed engineering students to relieve stress. The graphic design of the game gives out a very cute and funny vibe.

2.0 Design

The design is discussed in detail in this section.

2.1 Top-Module (Verilog code shown in Appendix A)

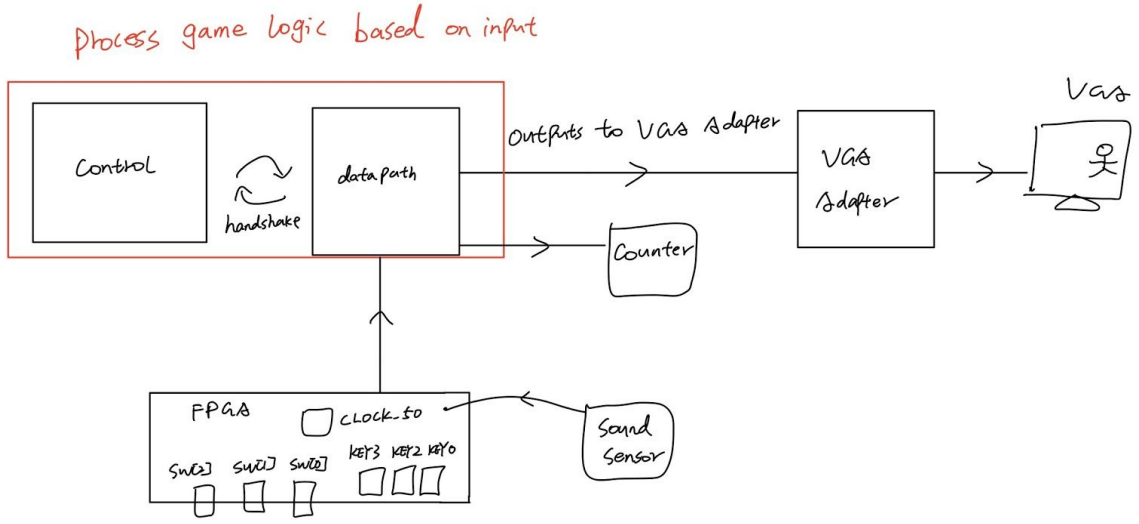

Figure 1: Top module block diagram

The player could:

1. Select difficulty using SW[2:0] (it can be changed any time during the game)
2. Press KEY[0] to start the game
3. Press KEY[2] to restart the game when game is over (the car hits the character)
4. Shout "jump" to the sound sensor to control the character

The above inputs along with CLOCK_50 are sent to the Game Logic Processor, as indicated by the red box in figure 1, to control the game. The Processor then outputs the corresponding image to the VGA adapter every 1/60 second, which then plots it to the screen(colour & x,y position are produced by datapath). There is also a counter that will

increment on each successful jump. If game is over, the counter will be reset to 0. The purpose of the counter is to record the score of the player.

2.2 Control (Verilog code shown in Appendix B)

The Control module gives “orders” to the datapath. It consists of 9 stages. The content of “order” is different in each stage. In general, the transition between stages happens when Control receives a “done” signal from datapath with one exceptional case. Below is a brief explanation of the stages and the orders they give:

Stage	Output Signal	Transition Condition	Explanation
reset	None	When KEY[0] (game_start) or KEY[2](restart) is pressed	Initial stage: set all output signals to 0 and wait for user input
drawBackground	go_draw_background	When it is done	Draw the game background
drawCar	go_draw_car	When it is done	Draw the car image from the “mif” file
drawPerson	go_draw_person	When it is done	Draw the person image from the “mif” file
waitDraw	go_divide	When it is done	Make the rate divider count down from a specific number, as set by SW[2:0]
clearScreen	go_draw_background	When it is done	Draw the game background (to erase car and person images)
updateCar	go_update_car	If “enable” is 1, it will transit to “updatePerson”. Otherwise it will transit to “drawCar”	Update the car location (to make it seem as moving)
updatePerson	go_update_person	None	Update the person location (to make it seem as jumping)
gameOver	go_draw_gameOver	When KEY[2] is pressed (the user decides to restart the game)	**NOTE: this stage will only be visited if game is over** Draw the gameOver image from the “mif” file

2.3 Datapath (Verilog code shown in Appendix C)
The datapath consists of 7 different modules. Each of them will be discussed below.

(1) drawBackground
This module works only if it receives the “go_drawBackground” or “go_clear_screen” signal. At each positive clock edge, it will increment x (max = 160) and y (max = 120), which is the pixel location to plot on the screen. Then it inputs the x,y locations into an address translator (a sample of address translator can be found in Appendix D) to produce an address. Then it will read from the background.mif at the specific address to get the colour we want to output (a ROM that is initialized with background.mif is required). When the process is done, a “done_drawBackground” signal is outputted to the Control module to indicate that a state transition can now be made.

(2) drawCar
This module works only if it receives the “go_drawCar” signal.
This module is very similar to “drawBackground” module except that

(a) The output “x” is added with “xShift” from “updateCar” module to make the car seem as moving. It is then subtracted by 40 to make the car drawn from outside of the left end of screen.
(b) The limit of “xAddress” is 40 and the limit of “yAddress” is determined by “carType” from “updateCar” module, since different cars have different height.
(c) The output colour is also determined by “carType”.
(d) For each type of car, a unique “yShift” is assigned to shift the car to the road on the background image.

(3) drawPerson
This module works only if it receives the “go_draw_person” signal.
This module is very similar to “drawBackground” module except that

(a) The output “x” is added with 120 to make the person standing at the right end of the road. The output “y” is added with “yShift” from “updatePerson” module to make the person seem as jumping.
(b) There are two sprites regarding the person: jumping and standing. When yShift is close to 50 (close to the road), colour from “stand.mif” would be output. In all other cases colour from “jump.mif” would be output.

(4) waitDraw
This module works only if it receives the “go_divide” signal.
At every positive clock edge, Q will count down 1 unit from the defined limit (limit is defined by difficulty, which is SW[2:0]). When Q reaches 0, a “done_divide” signal will be output and Q will be reset to limit.

(5) updatePerson
This module works only if it receives the “go_update_person” signal.
“y” is initially set to yMax (on the road). At each positive edge of clock, y is subtracted by one. When it reaches yMin (top of screen), it counts back up to yMax.
Our goal is to make the person jump from the road, reaches the top and falls back down to the road when the player shouts “jump”. So the x,y locations of the person have to be updated multiple times during the process. In order to achieve this goal, I declared an output register named “enable”, which is set to 0 initially. If the user shouts “jump”, “enable” will be set to 1. When the counter counted a full cycle (count down to yMin and count back up to

yMax), “enable” will be set back to 0. This signal will determine whether the transition should happen from “updateCar” to “updatePerson” or from “updateCar” to “drawCar”.

(6) updateCar
This module works only if it receives the “go_update_car” signal.
“x” is initially set to xMin (left end of screen). At each positive edge of clock, x is incremented by 1. When it reaches xMax, it is set back to xMin. Also, “carType” got incremented, indicating to “drawCar” module that a different car should now be drawn.

(7) drawGameOver
This module works only if it receives the “go_draw_gameOver” signal.
This module works identically the same as “drawBackground” module. The only exception is that the colour is output from “gameover.mif” instead of “background.mif”.

2.4 Game Over Condition
The game is regarded as over when the x position of car equals x position of person, and y position of car equals y position of person.

3.0 Report on Success

The overall game is successful since it is highly playable and amusing. But there was one problem throughout the project - the flashing issue of the images on VGA. Originally I used a resolution of 640x480, but I found out that the image is flashing so badly and it ruined the playability. One possible reason is that my code requires the FPGA to do a lot of drawing, but it failed to do so due to hardware limitations. If I wanted to fix this problem I would have to completely change my FSMs and therefore restart the project from the beginning. Since I do not have the time, I changed the resolution to 160x120 and hoped that it will fix the problem because the FPGA now could draw less. The result was both comforting and frustrating - the image was still flashing, but to a much lesser content.

4.0 What would I do differently

If I were to start the project all over again, I would use a completely different logic when I am implementing my control and datapath modules. Instead of drawing the entire image at one stage, it would just draw 1 pixel (so the drawing of the entire image requires thousands of cycles). Instead of drawing the images first and then drawing the background (serves as an eraser), it will draw only once, using the correct colour (car, person or background). I believe this way I could make the FPGA draw to a minimum content, and therefore fix the flashing problem.

5.0 Appendices

Appendix A: Top-module Verilog

```
module car
	(
		CLOCK_50,			//	On Board 50 MHz
		// Your inputs and outputs here
		KEY,
		SW,
		GPIO_0,
		HEX0,
		HEX1,
		LEDR,// On Board Keys
		// The ports below are for the VGA output.  Do not change.
		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]
	);
	input [2:0]SW;
	input [0:0]GPIO_0;
	input	CLOCK_50;		//	50 MHz
	input	[3:0]	KEY;
	output [6:0]HEX0,HEX1;
	output [8:0]LEDR;
	// Declare your inputs and outputs here
	// 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]

	wire resetn;
	assign resetn = ~KEY[3];
	wire game_start;
	assign game_start = ~KEY[0];
	wire jump;
```

```
        assign jump = GPIO_0[0];
        wire restart;
        assign restart = ~KEY[2];
        wire [2:0]difficulty;
        assign difficulty[2:0] = SW[2:0];

        // Create the colour, x, y and writeEn wires that are inputs to the controller.
        reg [2:0]colour;
        wire [2:0]colourBackground,colourCar,colourPerson,colourGameOver;
        reg [7:0]x;
        reg [6:0]y;
        wire [7:0]xCar,xPerson,xBackground,xShift,xGameOver;
        wire [6:0]yCar,yPerson,yBackground,yShift,yGameOver;
        wire [2:0]carType;
        wire go;
        wire
done_plot_background,done_divide,done_plot_car,done_plot_person,done_clearScreen,

go_drawBackground,go_drawCar,go_drawPerson,go_divide,go_clearScreen,go_updateCar,
go_updatePerson,go_draw_gameOver;
        reg gameOver;
        initial gameOver = 0;


car_control(resetn,restart,gameOver,CLOCK_50,game_start,go,done_plot_background,don
e_divide,done_plot_car,done_plot_person,done_clearScreen,

go_drawBackground,go_drawCar,go_drawPerson,go_divide,go_clearScreen,go_updateCar,
go_updatePerson,go_draw_gameOver);

        drawBackground
drawB(restart,gameOver,CLOCK_50,go_drawBackground,go_clearScreen,xBackground,yB
ackground,colourBackground,done_plot_background);

        drawCar
drawC(restart,gameOver,CLOCK_50,go_drawCar,carType,xShift,xCar,yCar,colourCar,done
_plot_car);

        drawPerson
drawP(restart,gameOver,CLOCK_50,go_drawPerson,yShift,xPerson,yPerson,colourPerson,
done_plot_person);

        waitDraw w(CLOCK_50,go_divide,go,difficulty,done_divide);

        updatePerson p(restart,gameOver,CLOCK_50,go_updatePerson,jump,yShift,go);
```

```
	updateCar c(restart,gameOver,CLOCK_50,go_updateCar,xShift,carType);

	always@(*) begin

		if(xCar == xPerson && yCar == yPerson) gameOver <= 1;
		else gameOver <= 0;

	end


drawGameOver(restart,gameOver,CLOCK_50,go_draw_gameOver,xGameOver,yGameOve
r,colourGameOver);

	reg [7:0]score;

	always@(negedge go or posedge restart) begin
		if(restart) score <= 0;
		else begin

			If (difficulty == 3'b000) score <= score + 1;
			else if  (difficulty == 3'b001) score <= score + 2;
			else if  (difficulty == 3'b010) score <= score + 3;
			else if  (difficulty == 3'b011) score <= score + 4;
			else if  (difficulty == 3'b100) score <= score + 5;
			else if  (difficulty == 3'b101) score <= score + 6;
			else if  (difficulty == 3'b110) score <= score + 7;
			else if  (difficulty == 3'b111) score <= score + 8;
		end
	end

	hex_decoder upper(score[7:4],HEX1);
	hex_decoder lower(score[3:0],HEX0);

	wire enable;

	assign enable = go_drawBackground | go_drawCar | go_drawPerson |
go_clearScreen | go_draw_gameOver;

	always@(*) begin

		if(go_drawBackground || go_clearScreen ) begin
				colour <= colourBackground;
				x <= xBackground;
				y <= yBackground;

		end
```

```
        else if(go_drawCar) begin
                if(colourCar == 3'b111) begin
                        colour <= colourBackground;
                        x <= xBackground;
                        y <= yBackground;
                 end
                else begin
                        colour <= colourCar;
                        x <= xCar;
                        y <= yCar;
                end
        end

        else if(go_drawPerson) begin
                if(colourPerson == 3'b111) begin
                        colour <= colourBackground;
                        x <= xBackground;
                        y <= yBackground;
                end
                else begin
                        colour <= colourPerson;
                        x <= xPerson;
                        y <= yPerson;
                end
        end

        else if(go_draw_gameOver) begin
                colour <= colourGameOver;
                x <= xGameOver;
                y <= yGameOver;
        end
end

// 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(enable),
                /* 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 = 1;
        defparam VGA.BACKGROUND_IMAGE = "cover.mif";

endmodule
```

Appendix B: Control Module Verilog

```
module car_control(
        input resetn,restart,game_over,clock,game_start,go,
        input
done_plot_background,done_divide,done_plot_car,done_plot_person,done_clearScreen,
        output reg
go_drawBackground,go_drawCar,go_drawPerson,go_divide,go_clearScreen,go_updateCar,
go_updatePerson,go_draw_gameOver);

parameter               reset = 0,
                        drawBackground = 1,
                        drawCar = 2,
                        drawPerson = 3,
                        waitDraw = 4,
                        clearScreen = 5,
                        updateCar = 6,
                        updatePerson = 7,
                        gameOver = 8;



reg [3:0]current_state,next_state;

always@(*)
begin : state_table
        case (current_state)
                reset : next_state = (game_start||restart) ? drawBackground : reset;
                drawBackground : next_state = done_plot_background ? drawCar :
drawBackground;
                drawCar : next_state = done_plot_car ? drawPerson : drawCar;
```

```
                drawPerson : next_state = done_plot_person ? waitDraw :
drawPerson;
                waitDraw : next_state = done_divide ? clearScreen : waitDraw;
                clearScreen : next_state = done_plot_background ? updateCar :
clearScreen;
                updateCar : begin
                        if(go) next_state = updatePerson;
                        else next_state = drawCar;
                end
                updatePerson : next_state = drawCar;
                gameOver : next_state = restart ? drawBackground : gameOver;

        endcase
end

always@(*)
begin : enable_signals

        case (current_state)

                reset : begin
                        go_drawBackground = 0;
                        go_drawCar = 0;
                        go_drawPerson = 0;
                        go_divide = 0;
                        go_clearScreen = 0;
                        go_updateCar = 0;
                        go_updatePerson = 0;
                        go_draw_gameOver = 0;
                    end

                drawBackground : begin
                        go_drawBackground = 1;
                        go_drawCar = 0;
                        go_drawPerson = 0;
                        go_divide = 0;
                        go_clearScreen = 0;
                        go_updateCar = 0;
                        go_updatePerson = 0;
                        go_draw_gameOver = 0;
                    end


                drawCar : begin
                        go_drawBackground = 0;
                        go_drawCar = 1;
```

```
                go_drawPerson = 0;
                go_divide = 0;
                go_clearScreen = 0;
                go_updateCar = 0;
                go_updatePerson = 0;

            end


drawPerson : begin
                go_drawBackground = 0;
                go_drawCar = 0;
                go_drawPerson = 1;
                go_divide = 0;
                go_clearScreen = 0;
                go_updateCar = 0;
                go_updatePerson = 0;

            end


waitDraw : begin
                go_drawBackground = 0;
                go_drawCar = 0;
                go_drawPerson = 0;
                go_divide = 1;
                go_clearScreen = 0;
                go_updateCar = 0;
                go_updatePerson = 0;

            end


clearScreen : begin
                go_drawBackground = 0;
                go_drawCar = 0;
                go_drawPerson = 0;
                go_divide = 0;
                go_clearScreen = 1;
                go_updateCar = 0;
                go_updatePerson = 0;

            end

updateCar : begin
                go_drawBackground = 0;
```

```
                                go_drawCar = 0;
                                go_drawPerson = 0;
                                go_divide = 0;
                                go_clearScreen = 0;
                                go_updateCar = 1;
                                go_updatePerson = 0;
                                go_draw_gameOver = 0;
                        end

                updatePerson : begin
                                go_drawBackground = 0;
                                go_drawCar = 0;
                                go_drawPerson = 0;
                                go_divide = 0;
                                go_clearScreen = 0;
                                go_updateCar = 0;
                                go_updatePerson = 1;
                                go_draw_gameOver = 0;
                        end

                gameOver : begin
                                go_drawBackground = 0;
                                go_drawCar = 0;
                                go_drawPerson = 0;
                                go_divide = 0;
                                go_clearScreen = 0;
                                go_updateCar = 0;
                                go_updatePerson = 0;
                                go_draw_gameOver = 1;
                        end

        endcase
end


always@ (posedge clock)
        begin : state_FFs
                if (resetn)
                        current_state<=reset;
                else if(game_over)
                        current_state<=gameOver;
                else
                        current_state<=next_state;
        end
endmodule
```

Appendix C: Datapath Verilog

```
//=============================================================
module drawBackground(
	input reset,gameOver,clock,go_drawBackground,go_clearScreen,
	output reg [7:0]x,
	output reg [6:0]y,
	output [2:0]colour,
	output reg done_drawBackground
);

	initial x = 0;
	initial y = 0;


parameter width = 160,
			height = 120;

wire [14:0]address;

lowVGA(x,y,address);
background back(address,clock,colour);

always@(posedge clock) begin
	if(reset||gameOver) begin
		x = 0;
		y = 0;
	end
	if(!go_drawBackground) done_drawBackground = 0;

	if(go_drawBackground || go_clearScreen) begin
		x = x+1;
		if(x == width) begin
			x = 0;
			y = y+1;
		end


		if(y == height) begin
			y = 0;
			done_drawBackground = 1;
		end
	end
end

endmodule
```

```
//============================================================
module drawCar(
	input reset,gameOver,clock,go_drawCar,
	input [2:0]carType,
	input [7:0]xShift,
	output signed [7:0]x,
	output [6:0]y,
	output reg [2:0]colour,
	output reg done_drawCar
);

reg[5:0] width = 40;
reg[5:0] height;
reg[6:0]yShift;


wire [10:0]address;
wire
[2:0]colourRaceCar,colourPolice,colourTruck,colourAnotherRace,colourTank,colourJiao,colo
urJail,colourDig;
reg [5:0]xAddress;
reg [4:0]yAddress;
racecarSprite(xAddress,yAddress,address);
racecar (address,clock,colourRaceCar);
policecar (address,clock,colourPolice);
truck (address,clock,colourTruck);
another (address,clock,colourAnotherRace);
tank (address,clock,colourTank);
jiao (address,clock,colourJiao);
jail (address,clock,colourJail);
dig (address,clock,colourDig);



always@(posedge clock) begin
	if(carType == 3'b000) begin
		height <= 10;
		colour <= colourRaceCar;
		yShift <= 61;
	end
	else if(carType == 3'b001) begin
		height <= 20;
		colour <= colourPolice;
		yShift <= 52;
	end
```

```
        else if(carType == 3'b010) begin
                height <= 30;
                colour <= colourDig;
                yShift <= 42;

        end
        else if(carType == 3'b011) begin
                height <= 30;
                colour <= colourTank;
                yShift <= 42;
        end
        else if(carType == 3'b100) begin
                height <= 15;
                colour <= colourAnotherRace;
                yShift <= 58;
        end
        else if(carType == 3'b101) begin
                height <= 30;
                colour <= colourJiao;
                yShift <= 45;
        end
        else if(carType == 3'b110) begin
                height <= 30;
                colour <= colourJail;
                yShift <= 45;
        end
        else if(carType == 3'b111) begin
                height <= 20;
                colour <= colourTruck;
                yShift <= 52;
        end

end



always@(posedge clock) begin
        if(reset||gameOver) begin
                        xAddress = 0;
                        yAddress = 0;
                end

        else begin
                if(!go_drawCar) done_drawCar = 0;

                if(go_drawCar) begin
```

```
                        xAddress = xAddress+1;
                        if(xAddress == width) begin
                                xAddress = 0;
                                yAddress = yAddress+1;
                        end

                        if(yAddress == height) begin
                                yAddress = 0;
                                done_drawCar = 1;
                        end
                end
        end
end

assign x = xShift -40 + xAddress;
assign y = yShift + yAddress;


endmodule
//============================================================

module drawPerson(
        input reset,gameOver,clock,go_drawPerson,
        input [6:0]yShift,
        output [7:0]x,
        output [6:0]y,
        output [2:0]colour,
        output reg done_drawPerson
);

parameter width = 24,
                         height = 24;

wire [10:0]address;
reg [4:0]xAddress;
reg [4:0]yAddress;
wire [2:0]colourStand;
wire [2:0]colourJump;

standSprite(xAddress,yAddress,address);
jump tiao(address,clock,colourJump);
stand zhan(address,clock,colourStand);

assign colour = (yShift >= 45 && yShift <=50) ? colourStand : colourJump;

always@(posedge clock) begin
```

```
	if(reset||gameOver) begin
			xAddress = 0;
			yAddress = 0;
		end
	if(!go_drawPerson) done_drawPerson = 0;

	if(go_drawPerson) begin
		xAddress = xAddress+1;
		if(xAddress == width) begin
			xAddress = 0;
			yAddress = yAddress+1;
		end

		if(yAddress == height) begin
			yAddress = 0;
			done_drawPerson = 1;
		end
	end
end

assign x = 120 + xAddress;
assign y = yShift + yAddress;

endmodule
//============================================================

module waitDraw(
	input clock,go_divide,jump,
	input [2:0]difficulty,
	output reg done_divide
	);
	reg [26:0] limit;
	reg [26:0] Q;

	always@(*) begin
		case (difficulty)
			3'b000: limit <= 808888;
			3'b001: limit <= 708888;
			3'b010: limit <= 608888;
			3'b011: limit <= 508888;
			3'b100: limit <= 408888;
			3'b101: limit <= 308888;
			3'b110: limit <= 208888;
			3'b111: limit <= 148888;
		endcase
	end
```

```
	always@(posedge clock) begin
		Q <= limit;
		 if(go_divide) begin
			if(Q<=0) begin
					done_divide <= 1;
					Q<=limit;
			end
			else begin
				done_divide <= 0;
				Q<=Q-1;
			end
		end
	end

endmodule
//=============================================================

module updatePerson(
	input reset,gameOver,clock,go_update_person,jump,
	output reg [6:0]y,
	output reg enable
);
	parameter  yMin=1,
				yMax=50;
	reg up = 0, down = 1;
	initial y=yMax;



	always@(posedge clock) begin
			if(reset||gameOver) begin
				y = yMax;
				enable = 0;
			end

			if(jump) enable = 1;

			if(go_update_person) begin
				if(down && y>yMin) y=y-1;
				if(y==yMin && down) begin
					down=0;
					up=1;
					y=y-1;
				end
```

```
                        if(y<yMax && up) y=y+1;
                        if(y==yMax) begin
                                down=1;
                                up=0;
                                enable = 0;
                        end
                end
        end


endmodule

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

module updateCar(
        input reset,gameOver,clock,go_update_car,
        output reg  [7:0]x,
        output reg  [2:0]carType
);

        parameter  xMin=0,
                           xMax=199;



        always@(posedge clock) begin
                if(reset||gameOver) begin
                        x <= xMin;
                        carType = 3'b000;
                end

                if(go_update_car) begin
                        if(x<xMax) x<=x+1;
                        if(x==xMax) begin
                                x<=xMin;
                                carType <= carType + 1;
                        end
                end
        end

endmodule
//=============================================================

module hex_decoder(hex_digit, segments);
   input [3:0] hex_digit;
   output reg [6:0] segments;
```

```
    always @(*)
        case (hex_digit)
            4'h0: segments = 7'b100_0000;
            4'h1: segments = 7'b111_1001;
            4'h2: segments = 7'b010_0100;
            4'h3: segments = 7'b011_0000;
            4'h4: segments = 7'b001_1001;
            4'h5: segments = 7'b001_0010;
            4'h6: segments = 7'b000_0010;
            4'h7: segments = 7'b111_1000;
            4'h8: segments = 7'b000_0000;
            4'h9: segments = 7'b001_1000;
            4'hA: segments = 7'b000_1000;
            4'hB: segments = 7'b000_0011;
            4'hC: segments = 7'b100_0110;
            4'hD: segments = 7'b010_0001;
            4'hE: segments = 7'b000_0110;
            4'hF: segments = 7'b000_1110;
            default: segments = 7'h7f;
        endcase
endmodule
//================================================================
module drawGameOver(
        input reset,gameOver,clock,go_draw_gameOver,
        output reg [7:0]x,
        output reg [6:0]y,
        output [2:0]colour
);

        initial x = 0;
        initial y = 0;


parameter width = 160,
                        height = 120;

wire [14:0]address;

lowVGA(x,y,address);
gameover over(address,clock,colour);

always@(posedge clock) begin
        if(reset||gameOver) begin
                x = 0;
                y = 0;
```

```
        end

        if(go_draw_gameOver) begin
                x = x+1;
                if(x == width) begin
                        x = 0;
                        y = y+1;
                end

                if(y == height) begin
                        y = 0;
                end
        end
end

endmodule
```

Appendix D: Race Car Sprite Address Translator

```
module racecarSprite(x, y, mem_address);

        parameter RESOLUTION = "40x10";
        input [5:0] x;
        input [4:0] y;
        output [10:0] mem_address;

        assign mem_address =({1'b0, y, 5'd0} + {1'b0, y, 3'd0} + {1'b0, x});

endmodule
```

Appendix E: Background & Gameover Image Address Translator

```
module lowVGA(x,y,mem_address);

        parameter RESOLUTION = "160x120";
        input [7:0] x;
        input [6:0] y;
        output [14:0] mem_address;

        assign mem_address = ({1'b0, y, 7'd0} + {1'b0, y, 5'd0} + {1'b0, x});

endmodule
```

Appendix F: Person Sprite Address Translator

```
module standSprite(x, y, mem_address);

	parameter RESOLUTION = "24x24";
	input [4:0] x;
	input [4:0] y;
	output [10:0] mem_address;

	assign mem_address =({1'b0, y, 4'd0} + {1'b0, y, 3'd0} + {1'b0, x});


endmodule
```

Appendix G: VGA PLL, Controller, Address Translator & Adapter

```
// 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 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

/* 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 = "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.
         */

        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
```

```
/* 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 = "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.
	 */

	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
```

Appendix H: ROMs

```
// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
```

```
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: background.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 Lite 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 (
        address,
        clock,
        q);

        input    [14:0]  address;
        input      clock;
        output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
```

```
`endif
	tri1	clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/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 = 19200,
		altsyncram_component.operation_mode = "ROM",
		altsyncram_component.outdata_aclr_a = "NONE",
		altsyncram_component.outdata_reg_a = "CLOCK0",
		altsyncram_component.widthad_a = 15,
```

```
		altsyncram_component.width_a = 3,
		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 "8"
// 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 "../../../bmp/bmp2mif/background.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "19200"
// Retrieval info: PRIVATE: RAM_BLOCK_TYPE NUMERIC "0"
// Retrieval info: PRIVATE: RegAddr NUMERIC "1"
// Retrieval info: PRIVATE: RegOutput NUMERIC "1"
// 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 "3"
// 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 "../../../bmp/bmp2mif/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 "19200"
```

```
// Retrieval info: CONSTANT: OPERATION_MODE STRING "ROM"
// Retrieval info: CONSTANT: OUTDATA_ACLR_A STRING "NONE"
// Retrieval info: CONSTANT: OUTDATA_REG_A STRING "CLOCK0"
// Retrieval info: CONSTANT: WIDTHAD_A NUMERIC "15"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// 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 3 0 OUTPUT NODEFVAL "q[2..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 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL background.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL background.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL background.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL background.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL background_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL background_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf


// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: stand.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 Lite 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 stand (
	address,
	clock,
	q);

	input	[10:0]  address;
	input	  clock;
	output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	  clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/stand.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 = 1056,
                altsyncram_component.operation_mode = "ROM",
                altsyncram_component.outdata_aclr_a = "NONE",
                altsyncram_component.outdata_reg_a = "UNREGISTERED",
                altsyncram_component.widthad_a = 11,
                altsyncram_component.width_a = 3,
                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 "8"
// 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 "../../../bmp/bmp2mif/stand.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "1056"
// 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 "11"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../../bmp/bmp2mif/stand.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 "1056"
// 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 "11"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 11 0 INPUT NODEFVAL "address[10..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 11 0 address 0 0 11 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL stand.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL stand.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL stand.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL stand.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL stand_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL stand_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
```

```
// MODULE: altsyncram

// ============================================================
// File Name: gameover.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 Lite 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 gameover (
        address,
        clock,
        q);

        input    [14:0]  address;
        input      clock;
        output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
```

```
	tri1	clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/gameover.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 = 19200,
		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 = 3,
```

```
		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 "8"
// 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 "../../../bmp/bmp2mif/gameover.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "19200"
// 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 "3"
// 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 "../../../bmp/bmp2mif/gameover.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 "19200"
// 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 "3"
// 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 3 0 OUTPUT NODEFVAL "q[2..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 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL gameover.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL gameover.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL gameover.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL gameover.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL gameover_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL gameover_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: racecar.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 Lite 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 racecar (
	address,
	clock,
	q);

	input	[8:0]  address;
	input	  clock;
	output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	  clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/racecar.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 = 400,
                altsyncram_component.operation_mode = "ROM",
                altsyncram_component.outdata_aclr_a = "NONE",
                altsyncram_component.outdata_reg_a = "UNREGISTERED",
                altsyncram_component.widthad_a = 9,
                altsyncram_component.width_a = 3,
                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 "8"
// 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 "../../../bmp/bmp2mif/racecar.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "400"
// 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 "9"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../../bmp/bmp2mif/racecar.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 "400"
// 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 "9"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 9 0 INPUT NODEFVAL "address[8..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 9 0 address 0 0 9 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL racecar.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL racecar.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL racecar.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL racecar.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL racecar_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL racecar_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram
```

```
// ============================================================
// File Name: jump.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 Lite 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 jump (
        address,
        clock,
        q);

        input    [9:0]  address;
        input      clock;
        output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
        tri1        clock;
`ifndef ALTERA_RESERVED_QIS
```

```
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/jump.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 = 576,
		altsyncram_component.operation_mode = "ROM",
		altsyncram_component.outdata_aclr_a = "NONE",
		altsyncram_component.outdata_reg_a = "UNREGISTERED",
		altsyncram_component.widthad_a = 10,
		altsyncram_component.width_a = 3,
		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 "8"
// 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 "../../../bmp/bmp2mif/jump.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "576"
// 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 "10"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../../bmp/bmp2mif/jump.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 "576"
// 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 "10"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 10 0 INPUT NODEFVAL "address[9..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 10 0 address 0 0 10 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL jump.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL jump.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jump.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jump.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jump_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jump_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: cover.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 Lite 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 cover (
	address,
	clock,
	q);

	input	[14:0]  address;
	input	  clock;
	output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	  clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../cover.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 = 19200,
                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 = 3,
                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 "8"
// 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 "../cover.mif"
```

```
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "19200"
// 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 "3"
// 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 "../cover.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 "19200"
// 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 "3"
// 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 3 0 OUTPUT NODEFVAL "q[2..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 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL cover.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL cover.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL cover.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL cover.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL cover_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL cover_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: policecar.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 Lite 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 policecar (
        address,
        clock,
        q);

        input    [9:0]  address;
        input      clock;
        output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
        tri1       clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif
```

```
	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/policecar.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 = 800,
		altsyncram_component.operation_mode = "ROM",
		altsyncram_component.outdata_aclr_a = "NONE",
		altsyncram_component.outdata_reg_a = "UNREGISTERED",
		altsyncram_component.widthad_a = 10,
		altsyncram_component.width_a = 3,
		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 "8"
// 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 "../../../bmp/bmp2mif/policecar.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "800"
// 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 "10"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../../bmp/bmp2mif/policecar.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 "800"
// 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 "10"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
```

```
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 10 0 INPUT NODEFVAL "address[9..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 10 0 address 0 0 10 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL policecar.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL policecar.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL policecar.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL policecar.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL policecar_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL policecar_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: truck.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 Lite 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 truck (
	address,
	clock,
	q);

	input	[9:0]  address;
	input	  clock;
	output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	  clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/truck.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 = 800,
        altsyncram_component.operation_mode = "ROM",
        altsyncram_component.outdata_aclr_a = "NONE",
        altsyncram_component.outdata_reg_a = "UNREGISTERED",
        altsyncram_component.widthad_a = 10,
        altsyncram_component.width_a = 3,
        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 "8"
// 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 "../../../bmp/bmp2mif/truck.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "800"
// 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 "10"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../../bmp/bmp2mif/truck.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 "800"
// 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 "10"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 10 0 INPUT NODEFVAL "address[9..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 10 0 address 0 0 10 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL truck.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL truck.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL truck.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL truck.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL truck_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL truck_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: another.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 Lite 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 another (
        address,
        clock,
        q);

        input    [9:0]  address;
        input      clock;
        output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
        tri1       clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

        wire [2:0] sub_wire0;
```

```
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/another.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 = 600,
		altsyncram_component.operation_mode = "ROM",
		altsyncram_component.outdata_aclr_a = "NONE",
		altsyncram_component.outdata_reg_a = "UNREGISTERED",
		altsyncram_component.widthad_a = 10,
		altsyncram_component.width_a = 3,
		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 "8"
// 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 "../../../bmp/bmp2mif/another.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "600"
// 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 "10"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../../bmp/bmp2mif/another.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 "600"
// 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 "10"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 10 0 INPUT NODEFVAL "address[9..0]"
```

```
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 10 0 address 0 0 10 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL another.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL another.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL another.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL another.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL another_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL another_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: tank.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 Lite 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 tank (
	address,
	clock,
	q);

	input	[10:0]  address;
	input	 clock;
	output	[2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	 clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../bmp/bmp2mif/tank.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 = 1200,
        altsyncram_component.operation_mode = "ROM",
        altsyncram_component.outdata_aclr_a = "NONE",
        altsyncram_component.outdata_reg_a = "UNREGISTERED",
        altsyncram_component.widthad_a = 11,
        altsyncram_component.width_a = 3,
        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 "8"
// 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 "../../bmp/bmp2mif/tank.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "1200"
// 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 "11"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../bmp/bmp2mif/tank.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 "1200"
// 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 "11"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 11 0 INPUT NODEFVAL "address[10..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 11 0 address 0 0 11 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL tank.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL tank.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL tank.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL tank.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL tank_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL tank_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: jiao.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 Lite 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 jiao (
        address,
        clock,
        q);

        input    [10:0]  address;
        input      clock;
        output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
        tri1        clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

        wire [2:0] sub_wire0;
        wire [2:0] q = sub_wire0[2:0];
```

```
	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 ({3{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 = "../../bmp/bmp2mif/jiao.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 = 1200,
		altsyncram_component.operation_mode = "ROM",
		altsyncram_component.outdata_aclr_a = "NONE",
		altsyncram_component.outdata_reg_a = "UNREGISTERED",
		altsyncram_component.widthad_a = 11,
		altsyncram_component.width_a = 3,
		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 "8"
// 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 "../../bmp/bmp2mif/jiao.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "1200"
// 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 "11"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../bmp/bmp2mif/jiao.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 "1200"
// 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 "11"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 11 0 INPUT NODEFVAL "address[10..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"

```
// Retrieval info: CONNECT: @address_a 0 0 11 0 address 0 0 11 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL jiao.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL jiao.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jiao.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jiao.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jiao_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jiao_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: jail.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 Lite 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 jail (
	address,
	clock,
	q);

	input	[10:0]  address;
	input	  clock;
	output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	  clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../bmp/bmp2mif/jail.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 = 1200,
                altsyncram_component.operation_mode = "ROM",
                altsyncram_component.outdata_aclr_a = "NONE",
                altsyncram_component.outdata_reg_a = "UNREGISTERED",
                altsyncram_component.widthad_a = 11,
                altsyncram_component.width_a = 3,
                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 "8"
// 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 "../../bmp/bmp2mif/jail.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "1200"
// 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 "11"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../bmp/bmp2mif/jail.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 "1200"
// 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 "11"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 11 0 INPUT NODEFVAL "address[10..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 11 0 address 0 0 11 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL jail.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL jail.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jail.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jail.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jail_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL jail_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf

// megafunction wizard: %ROM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: dig.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 Lite 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 dig (
	address,
	clock,
	q);

	input	[10:0]  address;
	input	  clock;
	output  [2:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
	tri1	  clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

	wire [2:0] sub_wire0;
	wire [2:0] q = sub_wire0[2:0];

	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 ({3{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 = "../../../bmp/bmp2mif/dig.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 = 1200,
                altsyncram_component.operation_mode = "ROM",
                altsyncram_component.outdata_aclr_a = "NONE",
                altsyncram_component.outdata_reg_a = "UNREGISTERED",
                altsyncram_component.widthad_a = 11,
                altsyncram_component.width_a = 3,
                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 "8"
// 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 "../../../bmp/bmp2mif/dig.mif"
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "1200"
// 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 "11"
// Retrieval info: PRIVATE: WidthData NUMERIC "3"
// 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 "../../../bmp/bmp2mif/dig.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 "1200"
// 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 "11"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "3"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 11 0 INPUT NODEFVAL "address[10..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: q 0 0 3 0 OUTPUT NODEFVAL "q[2..0]"
// Retrieval info: CONNECT: @address_a 0 0 11 0 address 0 0 11 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
```

// Retrieval info: CONNECT: q 0 0 3 0 @q_a 0 0 3 0
// Retrieval info: GEN_FILE: TYPE_NORMAL dig.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL dig.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL dig.cmp FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL dig.bsf FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL dig_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL dig_bb.v TRUE
// Retrieval info: LIB_FILE: altera_mf