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

// RK05 Emulator

// Top Level definition

// File Name: RK05\_emulator\_top\_v06.v

// Version 2.9

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

//

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

// //

// This FPGA firmware and related modules included by the top level module and files used to //

// build the software are provided on an as-is basis. No warrantees or guarantees are provided //

// or implied. Users of the RK05 Emulator or RK05 Tester shall not hold the developers of this //

// software, firmware, hardware, or related documentation liable for any damages caused by //

// any type of malfunction of the product including malfunctions caused by defects in the design //

// or operation of the software, firmware, hardware or use of related documentation or any //

// combination thereof. //

// //

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

//

`include "bus\_disk\_read.v"

`include "bus\_disk\_write.v"

`include "bus\_outputs.v"

`include "clock\_and\_reset.v"

`include "drive\_select.v"

`include "sdram\_controller.v"

`include "sector\_and\_index.v"

`include "seek\_to\_cylinder.v"

`include "spi\_interface.v"

`include "timing\_gen.v"

//================================= TOP LEVEL INPUT-OUTPUT DEFINITIONS =====================================

module RK05\_emulator\_top (

// BUS Connector Inputs, 20 signals

input wire BUS\_RK11D\_L, // RK11 mode for Drive Select BUS\_RK11D\_L

input wire [3:0] BUS\_SEL\_DR\_L, // Drive Select signals

input wire [7:0] BUS\_CYL\_ADD\_L, // Cylinder Address

input wire BUS\_STROBE\_L, // Strobe to enable movement, gates the Cylinder Address or Restore line

input wire BUS\_HEAD\_SELECT\_L, // Head Select

input wire BUS\_WT\_PROTECT\_L, // Write Protect

input wire BUS\_WT\_DATA\_CLK\_L, // Composite write data and write clock

input wire BUS\_WT\_GATE\_L, // Write gate, when active enables write circuitry

input wire BUS\_RESTORE\_L, // Restore, moves heads to cylinder 0

input wire BUS\_RD\_GATE\_L, // Read gate, when active enables read circuitry

// BUS Connector Outputs, 17 signals

output wire BUS\_FILE\_READY\_L, // data copied from microSD to SDRAM, drive is ready

output wire BUS\_RWS\_RDY\_L, // Read, Write or Seek ready and on-cylinder

output wire BUS\_ADDRESS\_ACCEPTED\_L, // Address accepted. 5 usec pulse indicates that the drive accepted a Seek command

output wire BUS\_ADDRESS\_INVALID\_L, // Address invalid. 5 usec pulse indicates that the drive received a nonexecutable Seek command, alternative ATTEN mode function is ATTEN4

output wire BUS\_SEEK\_INCOMPLETE\_L, // Seek Incomplete, indicates some malfunction, connected but not currently used

output wire BUS\_WT\_PROT\_STATUS\_L, // Write Protect status

output wire BUS\_WT\_CHK\_L, // Write Check, indicates a write malfunction, connected but not currently used

output wire BUS\_RD\_DATA\_L, // Read data pulses, 160 ns pulse

output wire BUS\_RD\_CLK\_L, // Read clock pulses, 160 ns pulse

output wire [4:0] BUS\_SEC\_CNTR\_L, // Sector Counter, sector counter bit 4 is pin 73 when not using Servo\_Pulse\_FPGA\_pin

output wire BUS\_SEC\_PLS\_L, // 2 us negative pulse each time a sector slot passes the transducer

output wire BUS\_INDX\_PLS\_L, // 2 us negative pulse for each revolution of the disk, 600 usec after the sector pulse

output wire BUS\_DC\_LO\_L, // DC low indicator, not used for the emulator, connected but not currently used

output wire BUS\_RK05\_HIGH\_DENSITY\_L, // All RK05's are high density, alternative ATTEN mode function is ATTEN2

// SDRAM I/O. 39 signals. 16 - DQ, 13 - Address: A0-A12, 2 - Addr Select: BS0 BS1, 8 - SDRAM Control: WE# CAS# RAS# CS# CLK CKE DQML DQMH

//input [15:0] SDRAM\_DQ\_in, // input from DQ signal receivers

//output [15:0] SDRAM\_DQ\_output, // outputs to DQ signal drivers

//output SDRAM\_DQ\_enable, // DQ output enable, active high

output wire [12:0] SDRAM\_Address,// SDRAM Address

output wire SDRAM\_BS0, // SDRAM Bank Select 0

output wire SDRAM\_BS1, // SDRAM Bank Select 1

output wire SDRAM\_WE\_n, // SDRAM Write

output wire SDRAM\_CAS\_n, // SDRAM Column Address Select

output wire SDRAM\_RAS\_n, // SDRAM Row Address Select

output wire SDRAM\_CS\_n, // SDRAM Chip Select

output wire SDRAM\_CLK, // SDRAM Clock

output wire SDRAM\_CKE, // SDRAM Clock Enable

output wire SDRAM\_DQML, // SDRAM DQ Mask for Lower Byte

output wire SDRAM\_DQMH, // SDRAM DQ Mask for Upper (High) Byte

inout wire [15:0] SDRAM\_DQ, //SDRAM multiplexed Data Input & Output

// ESP32 CPU SPI Port 6 signals: MISO MOSI CLK CS 2-bit-register-select

output wire CPU\_SPI\_MISO, // SPI Controller Input Peripheral Output

input wire CPU\_SPI\_MOSI, // SPI Controller Output Peripheral Input

input wire CPU\_SPI\_CLK, // SPI Controller Clock

input wire CPU\_SPI\_CS\_n, // SPI Controller Chip Select, active low

// Front Panel from FPGA, 4 signals. WT\_PROT\_indicator WT\_indicator RD\_indicator ON\_CYL\_indicator

output wire FPANEL\_WT\_PROT\_indicator, // Write Protect indicator

output wire FPANEL\_WT\_indicator, // Write indicator

output wire FPANEL\_RD\_indicator, // Read indicator

output wire FPANEL\_ON\_CYL\_indicator, // On Cylinder indicator

// Clock and Reset external pins: pin\_clock pin\_reset

input wire clock,

input wire pin\_reset\_n,

// Tester Outputs, new in Emulator v1 hardware

output wire TESTER\_OUTPUT\_1\_L, // this is pin 87

output wire TESTER\_OUTPUT\_2\_L, // this is pin 44, alternative ATTEN mode function is ATTEN1

output wire TESTER\_OUTPUT\_3\_L, // this is pin 45, alternative ATTEN mode function is ATTEN3

output wire Servo\_Pulse\_FPGA\_pin, // this is pin 73 when not using sector counter bit 4

input wire SPARE\_PIO1\_24, // this is pin 74, the spare input used in interface test mode to monitor the sector counter bit 4 output on bus pin AV1 driven by the tester

output wire SELECTED\_RDY\_LED\_N, // this is pin 75

output wire CMD\_INTERRUPT // this is pin 76

);

// ATTEN Mode summary:

// TESTER\_OUTPUT\_2\_L, FPGA pin 44, alternative ATTEN mode function is ATTEN1

// BUS\_RK05\_HIGH\_DENSITY\_L, alternative ATTEN mode function is ATTEN2

// TESTER\_OUTPUT\_3\_L, FPGA pin 45, alternative ATTEN mode function is ATTEN3

// BUS\_ADDRESS\_INVALID\_L, alternative ATTEN mode function is ATTEN4

//============================ Internal Connections ==================================

wire [7:0] MAJOR\_VERSION;

assign MAJOR\_VERSION = 2;

wire [7:0] MINOR\_VERSION;

assign MINOR\_VERSION = 9;

wire reset;

wire [7:0] test\_reg\_1;

wire [7:0] test\_reg\_2;

wire [11:0] test\_reg\_3;

//wire [4:0] number\_of\_sectors;

wire [4:0] number\_of\_sectors\_minus\_1;

wire [7:0] bitclockdivider\_clockphase;

wire [7:0] bitclockdivider\_dataphase;

wire [7:0] bitpulse\_width;

wire [15:0] microseconds\_per\_sector;

wire cpu\_dc\_low;

wire atten\_mode\_enable;

wire clkenbl\_sector;

wire clkenbl\_index;

wire bus\_sector\_pulse;

wire bus\_index\_pulse;

wire File\_Ready;

wire Write\_Protect;

wire Fault\_Latch;

wire [2:0] Drive\_Address;

wire Selected\_Ready;

wire [7:0] Cylinder\_Address;

wire Head\_Select;

wire [4:0] Sector\_Address;

wire [4:0] Sector\_Address\_Safe;

wire oncylinder\_indicator;

wire write\_indicator;

wire read\_indicator;

wire BUS\_RD\_DATA\_H;

wire BUS\_RD\_CLK\_H;

wire load\_address\_spi;

wire load\_address\_busread;

wire load\_address\_buswrite;

wire dram\_read\_enbl\_spi;

wire dram\_read\_enbl\_busread;

wire dram\_addr\_incr\_buswrite;

wire dram\_write\_enbl\_spi;

wire dram\_write\_enbl\_buswrite;

wire dram\_readack;

wire dram\_writeack;

wire [7:0] spi\_serpar\_reg;

wire [15:0] dram\_readdata;

wire [15:0] dram\_writedata\_spi;

wire [15:0] dram\_writedata\_buswrite;

wire [15:0] SDRAM\_DQ\_in;

wire [15:0] SDRAM\_DQ\_output;

wire SDRAM\_DQ\_enable;

wire clkenbl\_read\_bit;

wire clkenbl\_read\_data;

wire clock\_pulse;

wire data\_pulse;

wire clkenbl\_1usec;

wire [7:0] sdram\_debug;

wire [7:0] bdw\_test;

wire interface\_test\_mode;

wire strobe\_selected\_ready;

wire read\_selected\_ready;

wire write\_selected\_ready;

wire Servo\_Pulse\_FPGA;

//reg [7:0] test\_counter;

//wire [2:0] bdw\_state\_output; // for debugging for visibility of the bus write state

//============================ MISC TOP LEVEL LOGIC TO DRIVE THE INDICATORS ==================================

assign FPANEL\_WT\_PROT\_indicator = interface\_test\_mode ? ~test\_reg\_3[3] : ~Write\_Protect;

assign FPANEL\_WT\_indicator = interface\_test\_mode ? ~test\_reg\_3[1] : ~write\_indicator;

assign FPANEL\_RD\_indicator = interface\_test\_mode ? ~test\_reg\_3[0] : ~read\_indicator;

assign FPANEL\_ON\_CYL\_indicator = interface\_test\_mode ? ~test\_reg\_3[5] : ~(oncylinder\_indicator & File\_Ready);

//assign FPANEL\_WT\_indicator = ~(~BUS\_WT\_GATE\_L & Selected\_Ready);

//assign FPANEL\_RD\_indicator = ~(~BUS\_RD\_GATE\_L & Selected\_Ready);

//assign SPARE\_PIO1\_24 = sdram\_debug[0];

//assign SPARE\_PIO1\_24 = bdw\_test[6];

//assign SPARE\_PIO1\_24 = 1'b0;

assign SELECTED\_RDY\_LED\_N = ~Selected\_Ready;

assign TESTER\_OUTPUT\_1\_L = interface\_test\_mode ? ~test\_reg\_3[8] : ~1'b0; // this is pin 87

//assign TESTER\_OUTPUT\_2\_L = interface\_test\_mode ? ~test\_reg\_3[9] : ~(atten\_mode\_enable & ~Drive\_Address[2] & ~Drive\_Address[1] & ~Drive\_Address[0] & ~BUS\_RWS\_RDY\_H); // this is pin 44

assign TESTER\_OUTPUT\_2\_L = interface\_test\_mode ? ~test\_reg\_3[9] : ~(~Drive\_Address[2] & ~Drive\_Address[1] & ~Drive\_Address[0] & ~BUS\_RWS\_RDY\_H); // this is pin 44

// Don't qualify the ATTEN output with Selected\_Ready because the setting of the on\_cyl\_counter in seek\_to\_cylinder.v is already qualified by Selected\_Ready.

// We want the ATTEN to remain active in case the controller selects another drive immediately following a seek command (by activating Strobe)

// There are already individual outputs of ATTENx for each drive so they don't need to be qualified by the BUS\_SEL\_DR\_L address from the disk bus

// Instead, use Drive\_Address (the address selected by the dipswitches) to route the signal to the proper ATTENx output

assign TESTER\_OUTPUT\_3\_L = interface\_test\_mode ? ~test\_reg\_3[10] : ~1'b0; // this is pin 45

//assign TESTER\_OUTPUT\_1\_L = sdram\_debug[1]; // this is pin 87

//assign TESTER\_OUTPUT\_2\_L = sdram\_debug[2]; // this is pin 44

//assign TESTER\_OUTPUT\_3\_L = sdram\_debug[3]; // this is pin 45

//assign Servo\_Pulse\_FPGA\_pin = bdw\_test[7];

//assign Servo\_Pulse\_FPGA\_pin = Servo\_Pulse\_FPGA;

//assign Servo\_Pulse\_FPGA\_pin = ~Selected\_Ready;

//assign TESTER\_OUTPUT\_1 = test\_counter[0]; // this is pin 87

//assign TESTER\_OUTPUT\_2 = test\_counter[1]; // this is pin 44

//assign TESTER\_OUTPUT\_3 = test\_counter[2]; // this is pin 45

//assign Servo\_Pulse\_FPGA\_pin = test\_counter[3]; // this is pin 73

//assign SPARE\_PIO1\_24 = test\_counter[4]; // this is pin 74

//assign SELECTED\_RDY\_LED\_N = test\_counter[5]; // this is pin 75

//assign CMD\_INTERRUPTX = test\_counter[6]; // this is pin 76

//always @ (posedge clock)

//begin : PINTESTLOOP // block name

//if(reset==1'b1) begin

//test\_counter <= 8'd0;

//end

//else begin

//test\_counter <= test\_counter + 1;

//end

//end

//============================ SDRAM Bidirectional I/O pins ==================================

SB\_IO DQ00

(

.PACKAGE\_PIN (SDRAM\_DQ[0]), // User's Pin signal name

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable), // Output Pin Tristate/Enable control

.D\_OUT\_0 (SDRAM\_DQ\_output[0]), // Data 0 - out to Pin

.D\_IN\_0 (SDRAM\_DQ\_in[0]), // Data 0 - Pin input

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ00.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ00.PULLUP = 1'b0;

// By default, the IO will have NO pull up.

// This parameter is used only on bank 0, 1, and 2. Ignored when it is placed at bank 3

defparam DQ00.NEG\_TRIGGER = 1'b0;

// Specify the polarity of all FFs in the IO to be falling edge when NEG\_TRIGGER = 1.

// Default is rising edge.

//defparam DQ00.IO\_STANDARD = "LVCMOS";

// Other IO standards are supported in bank 3 only: SB\_SSTL2\_CLASS\_2, SB\_SSTL2\_CLASS\_1,

// SB\_SSTL18\_FULL, SB\_SSTL18\_HALF, SB\_MDDR10,SB\_MDDR8, SB\_MDDR4, SB\_MDDR2 etc.

SB\_IO DQ01

(

.PACKAGE\_PIN (SDRAM\_DQ[1]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[1]),

.D\_IN\_0 (SDRAM\_DQ\_in[1]),

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ01.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ01.PULLUP = 1'b0;

defparam DQ01.NEG\_TRIGGER = 1'b0;

//defparam DQ01.IO\_STANDARD = "LVCMOS";

SB\_IO DQ02

(

.PACKAGE\_PIN (SDRAM\_DQ[2]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[2]),

.D\_IN\_0 (SDRAM\_DQ\_in[2]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ02.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ02.PULLUP = 1'b0;

defparam DQ02.NEG\_TRIGGER = 1'b0;

//defparam DQ02.IO\_STANDARD = "LVCMOS";

SB\_IO DQ03

(

.PACKAGE\_PIN (SDRAM\_DQ[3]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[3]),

.D\_IN\_0 (SDRAM\_DQ\_in[3]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ03.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ03.PULLUP = 1'b0;

defparam DQ03.NEG\_TRIGGER = 1'b0;

//defparam DQ03.IO\_STANDARD = "LVCMOS";

SB\_IO DQ04

(

.PACKAGE\_PIN (SDRAM\_DQ[4]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[4]),

.D\_IN\_0 (SDRAM\_DQ\_in[4]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ04.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ04.PULLUP = 1'b0;

defparam DQ04.NEG\_TRIGGER = 1'b0;

//defparam DQ04.IO\_STANDARD = "LVCMOS";

SB\_IO DQ05

(

.PACKAGE\_PIN (SDRAM\_DQ[5]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[5]),

.D\_IN\_0 (SDRAM\_DQ\_in[5]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ05.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ05.PULLUP = 1'b0;

defparam DQ05.NEG\_TRIGGER = 1'b0;

//defparam DQ05.IO\_STANDARD = "LVCMOS";

SB\_IO DQ06

(

.PACKAGE\_PIN (SDRAM\_DQ[6]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[6]),

.D\_IN\_0 (SDRAM\_DQ\_in[6]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ06.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ06.PULLUP = 1'b0;

defparam DQ06.NEG\_TRIGGER = 1'b0;

//defparam DQ06.IO\_STANDARD = "LVCMOS";

SB\_IO DQ07

(

.PACKAGE\_PIN (SDRAM\_DQ[7]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[7]),

.D\_IN\_0 (SDRAM\_DQ\_in[7]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ07.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ07.PULLUP = 1'b0;

defparam DQ07.NEG\_TRIGGER = 1'b0;

//defparam DQ07.IO\_STANDARD = "LVCMOS";

SB\_IO DQ08

(

.PACKAGE\_PIN (SDRAM\_DQ[8]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[8]),

.D\_IN\_0 (SDRAM\_DQ\_in[8]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ08.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ08.PULLUP = 1'b0;

defparam DQ08.NEG\_TRIGGER = 1'b0;

//defparam DQ08.IO\_STANDARD = "LVCMOS";

SB\_IO DQ09

(

.PACKAGE\_PIN (SDRAM\_DQ[9]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[9]),

.D\_IN\_0 (SDRAM\_DQ\_in[9]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ09.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ09.PULLUP = 1'b0;

defparam DQ09.NEG\_TRIGGER = 1'b0;

//defparam DQ09.IO\_STANDARD = "LVCMOS";

SB\_IO DQ10

(

.PACKAGE\_PIN (SDRAM\_DQ[10]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[10]),

.D\_IN\_0 (SDRAM\_DQ\_in[10]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ10.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ10.PULLUP = 1'b0;

defparam DQ10.NEG\_TRIGGER = 1'b0;

//defparam DQ10.IO\_STANDARD = "LVCMOS";

SB\_IO DQ11

(

.PACKAGE\_PIN (SDRAM\_DQ[11]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[11]),

.D\_IN\_0 (SDRAM\_DQ\_in[11]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ11.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ11.PULLUP = 1'b0;

defparam DQ11.NEG\_TRIGGER = 1'b0;

//defparam DQ11.IO\_STANDARD = "LVCMOS";

SB\_IO DQ12

(

.PACKAGE\_PIN (SDRAM\_DQ[12]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[12]),

.D\_IN\_0 (SDRAM\_DQ\_in[12]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ12.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ12.PULLUP = 1'b0;

defparam DQ12.NEG\_TRIGGER = 1'b0;

//defparam DQ12.IO\_STANDARD = "LVCMOS";

SB\_IO DQ13

(

.PACKAGE\_PIN (SDRAM\_DQ[13]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[13]),

.D\_IN\_0 (SDRAM\_DQ\_in[13]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ13.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ13.PULLUP = 1'b0;

defparam DQ13.NEG\_TRIGGER = 1'b0;

//defparam DQ13.IO\_STANDARD = "LVCMOS";

SB\_IO DQ14

(

.PACKAGE\_PIN (SDRAM\_DQ[14]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[14]),

.D\_IN\_0 (SDRAM\_DQ\_in[14]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ14.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ14.PULLUP = 1'b0;

defparam DQ14.NEG\_TRIGGER = 1'b0;

//defparam DQ14.IO\_STANDARD = "LVCMOS";

SB\_IO DQ15

(

.PACKAGE\_PIN (SDRAM\_DQ[15]),

.OUTPUT\_ENABLE (SDRAM\_DQ\_enable),

.D\_OUT\_0 (SDRAM\_DQ\_output[15]),

.D\_IN\_0 (SDRAM\_DQ\_in[15]),

// n.c. pins we don't use

.LATCH\_INPUT\_VALUE (), .CLOCK\_ENABLE (), .INPUT\_CLK (), .OUTPUT\_CLK (), .D\_OUT\_1 (), .D\_IN\_1 ()

);

defparam DQ15.PIN\_TYPE = 6'b101001; // non-latched bi-directional I/O buffer

defparam DQ15.PULLUP = 1'b0;

defparam DQ15.NEG\_TRIGGER = 1'b0;

//defparam DQ15.IO\_STANDARD = "LVCMOS";

//================================= MODULES =====================================

// ======== Module ======== clock\_and\_reset =====

clock\_and\_reset i\_clock\_and\_reset (

// Inputs

.clock (clock),

.pin\_reset\_n (pin\_reset\_n),

// Outputs

.reset (reset)

);

// ======== Module ======== bus\_disk\_read =====

bus\_disk\_read i\_bus\_disk\_read (

// Inputs

.clock (clock),

.reset (reset),

.BUS\_RD\_GATE\_L (BUS\_RD\_GATE\_L),

.clkenbl\_read\_bit (clkenbl\_read\_bit),

.clkenbl\_read\_data (clkenbl\_read\_data),

.clock\_pulse (clock\_pulse),

.data\_pulse (data\_pulse),

.dram\_readdata (dram\_readdata),

.Selected\_Ready (Selected\_Ready),

.clkenbl\_sector (clkenbl\_sector),

.dram\_readack (dram\_readack),

// Outputs

.dram\_read\_enbl\_busread (dram\_read\_enbl\_busread),

.BUS\_RD\_DATA\_H (BUS\_RD\_DATA\_H),

.BUS\_RD\_CLK\_H (BUS\_RD\_CLK\_H),

.load\_address\_busread (load\_address\_busread),

.read\_indicator (read\_indicator),

.read\_selected\_ready (read\_selected\_ready)

);

// ======== Module ======== bus\_disk\_write =====

bus\_disk\_write i\_bus\_disk\_write (

// Inputs

.clock (clock),

.reset (reset),

.BUS\_WT\_GATE\_L (BUS\_WT\_GATE\_L),

.BUS\_WT\_DATA\_CLK\_L (BUS\_WT\_DATA\_CLK\_L),

.Selected\_Ready (Selected\_Ready),

.clkenbl\_sector (clkenbl\_sector),

.clkenbl\_read\_data (clkenbl\_read\_data),

.dram\_readack (dram\_readack),

.dram\_writeack (dram\_writeack),

// Outputs

.dram\_write\_enbl\_buswrite (dram\_write\_enbl\_buswrite),

.dram\_writedata\_buswrite (dram\_writedata\_buswrite),

.load\_address\_buswrite (load\_address\_buswrite),

.dram\_addr\_incr\_buswrite (dram\_addr\_incr\_buswrite),

.write\_indicator (write\_indicator),

.bdw\_debug (bdw\_test),

.write\_selected\_ready (write\_selected\_ready)

//.bdw\_state\_output (bdw\_state\_output)

);

// ======== Module ======== bus\_outputs =====

bus\_outputs i\_bus\_outputs (

// Inputs

.Selected (Selected),

.File\_Ready (File\_Ready),

.Fault\_Latch (Fault\_Latch),

.BUS\_RWS\_RDY\_H (BUS\_RWS\_RDY\_H),

.BUS\_ADDRESS\_ACCEPTED\_H (BUS\_ADDRESS\_ACCEPTED\_H),

.BUS\_ADDRESS\_INVALID\_H (BUS\_ADDRESS\_INVALID\_H),

//.BUS\_SEEK\_INCOMPLETE\_H (BUS\_SEEK\_INCOMPLETE\_H),

.Write\_Protect (Write\_Protect),

.BUS\_RD\_DATA\_H (BUS\_RD\_DATA\_H),

.BUS\_RD\_CLK\_H (BUS\_RD\_CLK\_H),

.BUS\_RD\_GATE\_L (BUS\_RD\_GATE\_L),

.Sector\_Address (Sector\_Address),

.bus\_sector\_pulse (bus\_sector\_pulse),

.bus\_index\_pulse (bus\_index\_pulse),

.cpu\_dc\_low (cpu\_dc\_low),

//.cpu\_dc\_low (File\_Ready), // for debugging to have visibility of File\_Ready

//.cpu\_dc\_low (~SDRAM\_DQ\_output[0]), // for debugging SDRAM\_DQ[0]

.interface\_test\_mode (interface\_test\_mode),

.test\_reg\_1 (test\_reg\_1),

.test\_reg\_2 (test\_reg\_2),

.test\_reg\_3 (test\_reg\_3),

// Outputs

.BUS\_FILE\_READY\_L (BUS\_FILE\_READY\_L),

.BUS\_RWS\_RDY\_L (BUS\_RWS\_RDY\_L),

.BUS\_ADDRESS\_ACCEPTED\_L (BUS\_ADDRESS\_ACCEPTED\_L),

.BUS\_ADDRESS\_INVALID\_L (BUS\_ADDRESS\_INVALID\_L),

.BUS\_SEEK\_INCOMPLETE\_L (BUS\_SEEK\_INCOMPLETE\_L),

.BUS\_WT\_PROT\_STATUS\_L (BUS\_WT\_PROT\_STATUS\_L),

.BUS\_WT\_CHK\_L (BUS\_WT\_CHK\_L),

.BUS\_RD\_DATA\_L (BUS\_RD\_DATA\_L),

.BUS\_RD\_CLK\_L (BUS\_RD\_CLK\_L),

.BUS\_SEC\_CNTR\_L (BUS\_SEC\_CNTR\_L),

.BUS\_SEC\_PLS\_L (BUS\_SEC\_PLS\_L),

.BUS\_INDX\_PLS\_L (BUS\_INDX\_PLS\_L),

//.BUS\_AC\_LO\_L (BUS\_AC\_LO\_L),

.BUS\_DC\_LO\_L (BUS\_DC\_LO\_L),

.BUS\_RK05\_HIGH\_DENSITY\_L (BUS\_RK05\_HIGH\_DENSITY\_L),

.Selected\_Ready (Selected\_Ready)

);

// ======== Module ======== drive\_select =====

drive\_select i\_drive\_select (

// Inputs

.BUS\_RK11D\_L (BUS\_RK11D\_L),

.BUS\_SEL\_DR\_L (BUS\_SEL\_DR\_L),

.Drive\_Address (Drive\_Address),

.RK05F\_Mode (1'b0),

// Outputs

.Selected (Selected)

);

// ======== Module ======== sdram\_controller =====

sdram\_controller i\_sdram\_controller (

// Inputs

.clock (clock),

.reset (reset),

.load\_address\_spi (load\_address\_spi),

.load\_address\_busread (load\_address\_busread),

.load\_address\_buswrite (load\_address\_buswrite),

.dram\_read\_enbl\_spi (dram\_read\_enbl\_spi),

.dram\_read\_enbl\_busread (dram\_read\_enbl\_busread),

.dram\_addr\_incr\_buswrite (dram\_addr\_incr\_buswrite),

.dram\_write\_enbl\_spi (dram\_write\_enbl\_spi),

.dram\_write\_enbl\_buswrite (dram\_write\_enbl\_buswrite),

.dram\_writedata\_spi (dram\_writedata\_spi),

.dram\_writedata\_buswrite (dram\_writedata\_buswrite),

.Write\_Protect (Write\_Protect),

.spi\_serpar\_reg (spi\_serpar\_reg),

.Sector\_Address\_Safe (Sector\_Address\_Safe),

.Cylinder\_Address (Cylinder\_Address),

.Head\_Select (Head\_Select),

.number\_of\_sectors\_minus\_1 (number\_of\_sectors\_minus\_1),

.SDRAM\_DQ\_in (SDRAM\_DQ\_in),

// Outputs

.dram\_readdata (dram\_readdata),

.dram\_readack (dram\_readack),

.dram\_writeack (dram\_writeack),

.SDRAM\_DQ\_output (SDRAM\_DQ\_output),

.SDRAM\_DQ\_enable (SDRAM\_DQ\_enable),

.SDRAM\_Address (SDRAM\_Address),

.SDRAM\_BS0 (SDRAM\_BS0),

.SDRAM\_BS1 (SDRAM\_BS1),

.SDRAM\_WE\_n (SDRAM\_WE\_n),

.SDRAM\_CAS\_n (SDRAM\_CAS\_n),

.SDRAM\_RAS\_n (SDRAM\_RAS\_n),

.SDRAM\_CS\_n (SDRAM\_CS\_n),

.SDRAM\_CLK (SDRAM\_CLK),

.SDRAM\_CKE (SDRAM\_CKE),

.SDRAM\_DQML (SDRAM\_DQML),

.SDRAM\_DQMH (SDRAM\_DQMH),

.sdram\_debug (sdram\_debug)

);

// ======== Module ======== sector\_and\_index =====

sector\_and\_index i\_sector\_and\_index (

// Inputs

.clock (clock),

.reset (reset),

.clkenbl\_1usec (clkenbl\_1usec),

.number\_of\_sectors\_minus\_1 (number\_of\_sectors\_minus\_1),

.microseconds\_per\_sector (microseconds\_per\_sector),

// Outputs

.clkenbl\_sector (clkenbl\_sector),

.clkenbl\_index (clkenbl\_index),

.bus\_sector\_pulse (bus\_sector\_pulse),

.bus\_index\_pulse (bus\_index\_pulse),

.Sector\_Address (Sector\_Address),

.Sector\_Address\_Safe (Sector\_Address\_Safe)

);

// ======== Module ======== seek\_to\_cylinder =====

seek\_to\_cylinder i\_seek\_to\_cylinder (

// Inputs

.clock (clock),

.reset (reset),

.Selected\_Ready (Selected\_Ready),

.BUS\_STROBE\_L (BUS\_STROBE\_L),

.BUS\_CYL\_ADD\_L (BUS\_CYL\_ADD\_L),

.BUS\_RESTORE\_L (BUS\_RESTORE\_L),

.BUS\_HEAD\_SELECT\_L (BUS\_HEAD\_SELECT\_L),

.clkenbl\_index (clkenbl\_index),

// Outputs

.Cylinder\_Address (Cylinder\_Address),

.Head\_Select (Head\_Select),

.BUS\_ADDRESS\_ACCEPTED\_H (BUS\_ADDRESS\_ACCEPTED\_H),

.BUS\_ADDRESS\_INVALID\_H (BUS\_ADDRESS\_INVALID\_H),

//.BUS\_SEEK\_INCOMPLETE\_H (BUS\_SEEK\_INCOMPLETE\_H),

.BUS\_RWS\_RDY\_H (BUS\_RWS\_RDY\_H),

.oncylinder\_indicator (oncylinder\_indicator),

.strobe\_selected\_ready (strobe\_selected\_ready)

);

// ======== Module ======== spi\_interface =====

spi\_interface i\_spi\_interface (

// Inputs

.clock (clock),

.reset (reset),

.spi\_clk (CPU\_SPI\_CLK),

.spi\_cs\_n (CPU\_SPI\_CS\_n),

.spi\_mosi (CPU\_SPI\_MOSI),

.dram\_readdata (dram\_readdata),

.dram\_readack (dram\_readack),

.dram\_writeack (dram\_writeack),

.BUS\_WT\_PROTECT\_L (BUS\_WT\_PROTECT\_L),

.Cylinder\_Address (Cylinder\_Address),

.Head\_Select (Head\_Select),

.Selected\_Ready (Selected\_Ready),

.BUS\_RK11D\_L (BUS\_RK11D\_L),

.BUS\_SEL\_DR\_L (BUS\_SEL\_DR\_L),

.BUS\_CYL\_ADD\_L (BUS\_CYL\_ADD\_L),

.BUS\_STROBE\_L (BUS\_STROBE\_L),

.BUS\_HEAD\_SELECT\_L (BUS\_HEAD\_SELECT\_L),

//.BUS\_WT\_PROTECT\_L (BUS\_WT\_PROTECT\_L),

.BUS\_WT\_DATA\_CLK\_L (BUS\_WT\_DATA\_CLK\_L),

.BUS\_WT\_GATE\_L (BUS\_WT\_GATE\_L),

.BUS\_RESTORE\_L (BUS\_RESTORE\_L),

.BUS\_RD\_GATE\_L (BUS\_RD\_GATE\_L),

.major\_version (MAJOR\_VERSION),

.minor\_version (MINOR\_VERSION),

.Sector\_Address (Sector\_Address),

.strobe\_selected\_ready (strobe\_selected\_ready),

.read\_selected\_ready (read\_selected\_ready),

.write\_selected\_ready (write\_selected\_ready),

.clkenbl\_1usec (clkenbl\_1usec),

.SPARE\_PIO1\_24 (SPARE\_PIO1\_24),

// Outputs

.spi\_miso (CPU\_SPI\_MISO),

.load\_address\_spi (load\_address\_spi),

.spi\_serpar\_reg (spi\_serpar\_reg),

.dram\_read\_enbl\_spi (dram\_read\_enbl\_spi),

.dram\_write\_enbl\_spi (dram\_write\_enbl\_spi),

.dram\_writedata\_spi (dram\_writedata\_spi),

.Drive\_Address (Drive\_Address),

.File\_Ready (File\_Ready),

.Write\_Protect (Write\_Protect),

.Fault\_Latch (Fault\_Latch),

.cpu\_dc\_low (cpu\_dc\_low),

.test\_reg\_1 (test\_reg\_1),

.test\_reg\_2 (test\_reg\_2),

.test\_reg\_3 (test\_reg\_3),

.number\_of\_sectors\_minus\_1 (number\_of\_sectors\_minus\_1),

.bitclockdivider\_clockphase (bitclockdivider\_clockphase),

.bitclockdivider\_dataphase (bitclockdivider\_dataphase),

.bitpulse\_width (bitpulse\_width),

.microseconds\_per\_sector (microseconds\_per\_sector),

.interface\_test\_mode (interface\_test\_mode),

.command\_interrupt (CMD\_INTERRUPT),

.Servo\_Pulse\_FPGA (Servo\_Pulse\_FPGA),

.atten\_mode\_enable (atten\_mode\_enable)

);

// ======== Module ======== timing\_gen =====

timing\_gen i\_timing\_gen (

// Inputs

.clock (clock),

.reset (reset),

.bitclockdivider\_clockphase (bitclockdivider\_clockphase),

.bitclockdivider\_dataphase (bitclockdivider\_dataphase),

.bitpulse\_width (bitpulse\_width),

// Outputs

.clkenbl\_read\_bit (clkenbl\_read\_bit),

.clkenbl\_read\_data (clkenbl\_read\_data),

.clock\_pulse (clock\_pulse),

.data\_pulse (data\_pulse),

.clkenbl\_1usec (clkenbl\_1usec)

);

endmodule // RK05\_emulator\_top