`include "axi\_bus.sv"

`include "apb\_bus.sv"

`include "debug\_bus.sv"

`include "config.sv"

module peripherals

#(

parameter AXI\_ADDR\_WIDTH = 32,

parameter AXI\_DATA\_WIDTH = 64,

parameter AXI\_USER\_WIDTH = 6,

parameter AXI\_SLAVE\_ID\_WIDTH = 6,

parameter AXI\_MASTER\_ID\_WIDTH = 6,

parameter ROM\_START\_ADDR = 32'h8000

)

(

// Clock and Reset

input logic clk\_i,

input logic rst\_n,

AXI\_BUS.Master axi\_spi\_master,

DEBUG\_BUS.Master debug,

input logic spi\_clk\_i,

input logic testmode\_i,

input logic spi\_cs\_i,

output logic [1:0] spi\_mode\_o,

output logic spi\_sdo0\_o,

output logic spi\_sdo1\_o,

output logic spi\_sdo2\_o,

output logic spi\_sdo3\_o,

input logic spi\_sdi0\_i,

input logic spi\_sdi1\_i,

input logic spi\_sdi2\_i,

input logic spi\_sdi3\_i,

AXI\_BUS.Slave slave,

output logic uart\_tx,

input logic uart\_rx,

output logic uart\_rts,

output logic uart\_dtr,

input logic uart\_cts,

input logic uart\_dsr,

output logic spi\_master\_clk,

output logic spi\_master\_csn0,

output logic spi\_master\_csn1,

output logic spi\_master\_csn2,

output logic spi\_master\_csn3,

output logic [1:0] spi\_master\_mode,

output logic spi\_master\_sdo0,

output logic spi\_master\_sdo1,

output logic spi\_master\_sdo2,

output logic spi\_master\_sdo3,

input logic spi\_master\_sdi0,

input logic spi\_master\_sdi1,

input logic spi\_master\_sdi2,

input logic spi\_master\_sdi3,

input logic scl\_pad\_i,

output logic scl\_pad\_o,

output logic scl\_padoen\_o,

input logic sda\_pad\_i,

output logic sda\_pad\_o,

output logic sda\_padoen\_o,

input logic [31:0] gpio\_in,

output logic [31:0] gpio\_out,

output logic [31:0] gpio\_dir,

output logic [31:0] [5:0] gpio\_padcfg,

input logic core\_busy\_i,

output logic [31:0] irq\_o,

input logic fetch\_enable\_i,

output logic fetch\_enable\_o,

output logic clk\_gate\_core\_o,

output logic fll1\_req\_o,

output logic fll1\_wrn\_o,

output logic [1:0] fll1\_add\_o,

output logic [31:0] fll1\_wdata\_o,

input logic fll1\_ack\_i,

input logic [31:0] fll1\_rdata\_i,

input logic fll1\_lock\_i,

output logic [31:0] [5:0] pad\_cfg\_o,

output logic [31:0] pad\_mux\_o,

output logic [31:0] boot\_addr\_o

);

localparam APB\_ADDR\_WIDTH = 32;

localparam APB\_NUM\_SLAVES = 8;

APB\_BUS s\_apb\_bus();

APB\_BUS s\_uart\_bus();

APB\_BUS s\_gpio\_bus();

APB\_BUS s\_spi\_bus();

APB\_BUS s\_timer\_bus();

APB\_BUS s\_event\_unit\_bus();

APB\_BUS s\_i2c\_bus();

APB\_BUS s\_fll\_bus();

APB\_BUS s\_soc\_ctrl\_bus();

APB\_BUS s\_debug\_bus();

logic [1:0] s\_spim\_event;

logic [3:0] timer\_irq;

logic [31:0] peripheral\_clock\_gate\_ctrl;

logic [31:0] clk\_int;

logic s\_uart\_event;

logic i2c\_event;

logic s\_gpio\_event;

//////////////////////////////////////////////////////////////////

/// ///

/// Peripheral Clock Gating ///

/// ///

//////////////////////////////////////////////////////////////////

generate

genvar i;

for (i = 0; i < APB\_NUM\_SLAVES; i = i + 1) begin

cluster\_clock\_gating core\_clock\_gate

(

.clk\_o ( clk\_int[i] ),

.en\_i ( peripheral\_clock\_gate\_ctrl[i] ),

.test\_en\_i ( testmode\_i ),

.clk\_i ( clk\_i )

);

end

endgenerate

//////////////////////////////////////////////////////////////////

/// ///

/// SPI Slave, AXI Master ///

/// ///

//////////////////////////////////////////////////////////////////

axi\_spi\_slave\_wrap

#(

.AXI\_ADDRESS\_WIDTH ( AXI\_ADDR\_WIDTH ),

.AXI\_DATA\_WIDTH ( AXI\_DATA\_WIDTH ),

.AXI\_USER\_WIDTH ( AXI\_USER\_WIDTH ),

.AXI\_ID\_WIDTH ( AXI\_MASTER\_ID\_WIDTH )

)

axi\_spi\_slave\_i

(

.clk\_i ( clk\_int[0] ),

.rst\_ni ( rst\_n ),

.test\_mode ( testmode\_i ),

.axi\_master ( axi\_spi\_master ),

.spi\_clk ( spi\_clk\_i ),

.spi\_cs ( spi\_cs\_i ),

.spi\_mode ( spi\_mode\_o ),

.spi\_sdo0 ( spi\_sdo0\_o ),

.spi\_sdo1 ( spi\_sdo1\_o ),

.spi\_sdo2 ( spi\_sdo2\_o ),

.spi\_sdo3 ( spi\_sdo3\_o ),

.spi\_sdi0 ( spi\_sdi0\_i ),

.spi\_sdi1 ( spi\_sdi1\_i ),

.spi\_sdi2 ( spi\_sdi2\_i ),

.spi\_sdi3 ( spi\_sdi3\_i )

);

//////////////////////////////////////////////////////////////////

/// ///

/// AXI2APB Bridge ///

/// ///

//////////////////////////////////////////////////////////////////

axi2apb\_wrap

#(

.AXI\_ADDR\_WIDTH ( AXI\_ADDR\_WIDTH ),

.AXI\_DATA\_WIDTH ( AXI\_DATA\_WIDTH ),

.AXI\_USER\_WIDTH ( AXI\_USER\_WIDTH ),

.AXI\_ID\_WIDTH ( AXI\_SLAVE\_ID\_WIDTH ),

.APB\_ADDR\_WIDTH ( APB\_ADDR\_WIDTH )

)

axi2apb\_i

(

.clk\_i ( clk\_i ),

.rst\_ni ( rst\_n ),

.test\_en\_i ( testmode\_i ),

.axi\_slave ( slave ),

.apb\_master( s\_apb\_bus )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Bus ///

/// ///

//////////////////////////////////////////////////////////////////

periph\_bus\_wrap

#(

.APB\_ADDR\_WIDTH( APB\_ADDR\_WIDTH ),

.APB\_DATA\_WIDTH( 32 )

)

periph\_bus\_i

(

.clk\_i ( clk\_i ),

.rst\_ni ( rst\_n ),

.apb\_slave ( s\_apb\_bus ),

.uart\_master ( s\_uart\_bus ),

.gpio\_master ( s\_gpio\_bus ),

.spi\_master ( s\_spi\_bus ),

.timer\_master ( s\_timer\_bus ),

.event\_unit\_master ( s\_event\_unit\_bus ),

.i2c\_master ( s\_i2c\_bus ),

.fll\_master ( s\_fll\_bus ),

.soc\_ctrl\_master ( s\_soc\_ctrl\_bus ),

.debug\_master ( s\_debug\_bus )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 0: APB UART interface ///

/// ///

//////////////////////////////////////////////////////////////////

`ifndef VERILATOR

apb\_uart apb\_uart\_i (

.CLK ( clk\_int[1] ),

.RSTN ( rst\_n ),

.PSEL ( s\_uart\_bus.psel ),

.PENABLE ( s\_uart\_bus.penable ),

.PWRITE ( s\_uart\_bus.pwrite ),

.PADDR ( s\_uart\_bus.paddr[4:2] ),

.PWDATA ( s\_uart\_bus.pwdata ),

.PRDATA ( s\_uart\_bus.prdata ),

.PREADY ( s\_uart\_bus.pready ),

.PSLVERR ( s\_uart\_bus.pslverr ),

.INT ( s\_uart\_event ), //Interrupt output

.OUT1N (), //Output 1

.OUT2N (), //Output 2

.RTSN ( uart\_rts ), //RTS output

.DTRN ( uart\_dtr ), //DTR output

.CTSN ( uart\_cts ), //CTS input

.DSRN ( uart\_dsr ), //DSR input

.DCDN ( 1'b1 ), //DCD input

.RIN ( 1'b1 ), //RI input

.SIN ( uart\_rx ),

.SOUT ( uart\_tx )

);

`else

apb\_uart\_sv

#(

.APB\_ADDR\_WIDTH( 3 )

)

apb\_uart\_i

(

.CLK ( clk\_int[1] ),

.RSTN ( rst\_n ),

.PSEL ( s\_uart\_bus.psel ),

.PENABLE ( s\_uart\_bus.penable ),

.PWRITE ( s\_uart\_bus.pwrite ),

.PADDR ( s\_uart\_bus.paddr[4:2] ),

.PWDATA ( s\_uart\_bus.pwdata ),

.PRDATA ( s\_uart\_bus.prdata ),

.PREADY ( s\_uart\_bus.pready ),

.PSLVERR ( s\_uart\_bus.pslverr ),

.rx\_i ( uart\_rx ),

.tx\_o ( uart\_tx ),

.event\_o ( s\_uart\_event )

);

`endif

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 1: APB GPIO interface ///

/// ///

//////////////////////////////////////////////////////////////////

apb\_gpio apb\_gpio\_i

(

.HCLK ( clk\_int[2] ),

.HRESETn ( rst\_n ),

.PADDR ( s\_gpio\_bus.paddr[11:0]),

.PWDATA ( s\_gpio\_bus.pwdata ),

.PWRITE ( s\_gpio\_bus.pwrite ),

.PSEL ( s\_gpio\_bus.psel ),

.PENABLE ( s\_gpio\_bus.penable ),

.PRDATA ( s\_gpio\_bus.prdata ),

.PREADY ( s\_gpio\_bus.pready ),

.PSLVERR ( s\_gpio\_bus.pslverr ),

.gpio\_in ( gpio\_in ),

.gpio\_out ( gpio\_out ),

.gpio\_dir ( gpio\_dir ),

.gpio\_padcfg ( gpio\_padcfg ),

.interrupt ( s\_gpio\_event )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 2: APB SPI Master interface ///

/// ///

//////////////////////////////////////////////////////////////////

apb\_spi\_master

#(

.BUFFER\_DEPTH(8)

)

apb\_spi\_master\_i

(

.HCLK ( clk\_int[3] ),

.HRESETn ( rst\_n ),

.PADDR ( s\_spi\_bus.paddr[11:0]),

.PWDATA ( s\_spi\_bus.pwdata ),

.PWRITE ( s\_spi\_bus.pwrite ),

.PSEL ( s\_spi\_bus.psel ),

.PENABLE ( s\_spi\_bus.penable ),

.PRDATA ( s\_spi\_bus.prdata ),

.PREADY ( s\_spi\_bus.pready ),

.PSLVERR ( s\_spi\_bus.pslverr ),

.events\_o ( s\_spim\_event ),

.spi\_clk ( spi\_master\_clk ),

.spi\_csn0 ( spi\_master\_csn0 ),

.spi\_csn1 ( spi\_master\_csn1 ),

.spi\_csn2 ( spi\_master\_csn2 ),

.spi\_csn3 ( spi\_master\_csn3 ),

.spi\_mode ( spi\_master\_mode ),

.spi\_sdo0 ( spi\_master\_sdo0 ),

.spi\_sdo1 ( spi\_master\_sdo1 ),

.spi\_sdo2 ( spi\_master\_sdo2 ),

.spi\_sdo3 ( spi\_master\_sdo3 ),

.spi\_sdi0 ( spi\_master\_sdi0 ),

.spi\_sdi1 ( spi\_master\_sdi1 ),

.spi\_sdi2 ( spi\_master\_sdi2 ),

.spi\_sdi3 ( spi\_master\_sdi3 )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 3: Timer Unit ///

/// ///

//////////////////////////////////////////////////////////////////

apb\_timer

apb\_timer\_i

(

.HCLK ( clk\_int[4] ),

.HRESETn ( rst\_n ),

.PADDR ( s\_timer\_bus.paddr[11:0]),

.PWDATA ( s\_timer\_bus.pwdata ),

.PWRITE ( s\_timer\_bus.pwrite ),

.PSEL ( s\_timer\_bus.psel ),

.PENABLE ( s\_timer\_bus.penable ),

.PRDATA ( s\_timer\_bus.prdata ),

.PREADY ( s\_timer\_bus.pready ),

.PSLVERR ( s\_timer\_bus.pslverr ),

.irq\_o ( timer\_irq )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 4: Event Unit ///

/// ///

//////////////////////////////////////////////////////////////////

apb\_event\_unit

apb\_event\_unit\_i

(

.clk\_i ( clk\_i ),

.HCLK ( clk\_int[5] ),

.HRESETn ( rst\_n ),

.PADDR ( s\_event\_unit\_bus.paddr[11:0]),

.PWDATA ( s\_event\_unit\_bus.pwdata ),

.PWRITE ( s\_event\_unit\_bus.pwrite ),

.PSEL ( s\_event\_unit\_bus.psel ),

.PENABLE ( s\_event\_unit\_bus.penable ),

.PRDATA ( s\_event\_unit\_bus.prdata ),

.PREADY ( s\_event\_unit\_bus.pready ),

.PSLVERR ( s\_event\_unit\_bus.pslverr ),

.irq\_i ( {timer\_irq, s\_spim\_event, s\_gpio\_event, s\_uart\_event, i2c\_event, 23'b0} ),

.event\_i ( {timer\_irq, s\_spim\_event, s\_gpio\_event, s\_uart\_event, i2c\_event, 23'b0} ),

.irq\_o ( irq\_o ),

.fetch\_enable\_i ( fetch\_enable\_i ),

.fetch\_enable\_o ( fetch\_enable\_o ),

.clk\_gate\_core\_o ( clk\_gate\_core\_o ),

.core\_busy\_i ( core\_busy\_i )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 5: I2C ///

/// ///

//////////////////////////////////////////////////////////////////

apb\_i2c

apb\_i2c\_i

(

.HCLK ( clk\_int[6] ),

.HRESETn ( rst\_n ),

.PADDR ( s\_i2c\_bus.paddr[11:0] ),

.PWDATA ( s\_i2c\_bus.pwdata ),

.PWRITE ( s\_i2c\_bus.pwrite ),

.PSEL ( s\_i2c\_bus.psel ),

.PENABLE ( s\_i2c\_bus.penable ),

.PRDATA ( s\_i2c\_bus.prdata ),

.PREADY ( s\_i2c\_bus.pready ),

.PSLVERR ( s\_i2c\_bus.pslverr ),

.interrupt\_o ( i2c\_event ),

.scl\_pad\_i ( scl\_pad\_i ),

.scl\_pad\_o ( scl\_pad\_o ),

.scl\_padoen\_o ( scl\_padoen\_o ),

.sda\_pad\_i ( sda\_pad\_i ),

.sda\_pad\_o ( sda\_pad\_o ),

.sda\_padoen\_o ( sda\_padoen\_o )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 6: FLL Ctrl ///

/// ///

//////////////////////////////////////////////////////////////////

apb\_fll\_if apb\_fll\_if\_i

(

.HCLK ( clk\_int[7] ),

.HRESETn ( rst\_n ),

.PADDR ( s\_fll\_bus.paddr[11:0]),

.PWDATA ( s\_fll\_bus.pwdata ),

.PWRITE ( s\_fll\_bus.pwrite ),

.PSEL ( s\_fll\_bus.psel ),

.PENABLE ( s\_fll\_bus.penable ),

.PRDATA ( s\_fll\_bus.prdata ),

.PREADY ( s\_fll\_bus.pready ),

.PSLVERR ( s\_fll\_bus.pslverr ),

.fll1\_req ( fll1\_req\_o ),

.fll1\_wrn ( fll1\_wrn\_o ),

.fll1\_add ( fll1\_add\_o ),

.fll1\_data ( fll1\_wdata\_o ),

.fll1\_ack ( fll1\_ack\_i ),

.fll1\_r\_data ( fll1\_rdata\_i ),

.fll1\_lock ( fll1\_lock\_i ),

.fll2\_req ( ),

.fll2\_wrn ( ),

.fll2\_add ( ),

.fll2\_data ( ),

.fll2\_ack ( 1'b0 ),

.fll2\_r\_data ( '0 ),

.fll2\_lock ( 1'b0 )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 7: PULPino control ///

/// ///

//////////////////////////////////////////////////////////////////

apb\_pulpino

#(

.BOOT\_ADDR ( ROM\_START\_ADDR )

)

apb\_pulpino\_i

(

.HCLK ( clk\_i ),

.HRESETn ( rst\_n ),

.PADDR ( s\_soc\_ctrl\_bus.paddr[11:0]),

.PWDATA ( s\_soc\_ctrl\_bus.pwdata ),

.PWRITE ( s\_soc\_ctrl\_bus.pwrite ),

.PSEL ( s\_soc\_ctrl\_bus.psel ),

.PENABLE ( s\_soc\_ctrl\_bus.penable ),

.PRDATA ( s\_soc\_ctrl\_bus.prdata ),

.PREADY ( s\_soc\_ctrl\_bus.pready ),

.PSLVERR ( s\_soc\_ctrl\_bus.pslverr ),

.pad\_cfg\_o ( pad\_cfg\_o ),

.clk\_gate\_o ( peripheral\_clock\_gate\_ctrl ),

.pad\_mux\_o ( pad\_mux\_o ),

.boot\_addr\_o ( boot\_addr\_o )

);

//////////////////////////////////////////////////////////////////

/// ///

/// APB Slave 8: APB2PER for debug ///

/// ///

//////////////////////////////////////////////////////////////////

apb2per

#(

.PER\_ADDR\_WIDTH ( 15 ),

.APB\_ADDR\_WIDTH ( APB\_ADDR\_WIDTH )

)

apb2per\_debug\_i

(

.clk\_i ( clk\_i ),

.rst\_ni ( rst\_n ),

.PADDR ( s\_debug\_bus.paddr ),

.PWDATA ( s\_debug\_bus.pwdata ),

.PWRITE ( s\_debug\_bus.pwrite ),

.PSEL ( s\_debug\_bus.psel ),

.PENABLE ( s\_debug\_bus.penable ),

.PRDATA ( s\_debug\_bus.prdata ),

.PREADY ( s\_debug\_bus.pready ),

.PSLVERR ( s\_debug\_bus.pslverr ),

.per\_master\_req\_o ( debug.req ),

.per\_master\_add\_o ( debug.addr ),

.per\_master\_we\_o ( debug.we ),

.per\_master\_wdata\_o ( debug.wdata ),

.per\_master\_be\_o ( ),

.per\_master\_gnt\_i ( debug.gnt ),

.per\_master\_r\_valid\_i ( debug.rvalid ),

.per\_master\_r\_opc\_i ( '0 ),

.per\_master\_r\_rdata\_i ( debug.rdata )

);

endmodule