`include "axi\_bus.sv"

`include "debug\_bus.sv"

`define AXI\_ADDR\_WIDTH 32

`define AXI\_DATA\_WIDTH 32

`define AXI\_ID\_MASTER\_WIDTH 2

`define AXI\_ID\_SLAVE\_WIDTH 4

`define AXI\_USER\_WIDTH 1

module chip\_two\_top

#(

parameter USE\_ZERO\_RISCY = 0,

parameter RISCY\_RV32F = 0,

parameter ZERO\_RV32M = 1,

parameter ZERO\_RV32E = 0

)

(

// Clock and Reset

input logic clk /\*verilator clocker\*/,

input logic rst\_n,

input logic clk\_sel\_i,

input logic clk\_standalone\_i,

input logic testmode\_i,

input logic fetch\_enable\_i,

input logic scan\_enable\_i,

//SPI Slave

input logic spi\_clk\_i /\*verilator clocker\*/,

input logic spi\_cs\_i /\*verilator clocker\*/,

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,

//SPI Master

output logic spi\_master\_clk\_o,

output logic spi\_master\_csn0\_o,

output logic spi\_master\_csn1\_o,

output logic spi\_master\_csn2\_o,

output logic spi\_master\_csn3\_o,

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

output logic spi\_master\_sdo0\_o,

output logic spi\_master\_sdo1\_o,

output logic spi\_master\_sdo2\_o,

output logic spi\_master\_sdo3\_o,

input logic spi\_master\_sdi0\_i,

input logic spi\_master\_sdi1\_i,

input logic spi\_master\_sdi2\_i,

input logic spi\_master\_sdi3\_i,

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,

output logic uart\_tx,

input logic uart\_rx,

output logic uart\_rts,

output logic uart\_dtr,

input logic uart\_cts,

input logic uart\_dsr,

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,

// JTAG signals

input logic tck\_i,

input logic trstn\_i,

input logic tms\_i,

input logic tdi\_i,

output logic tdo\_o,

// PULPino specific pad config

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

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

);

logic clk\_int;

logic fetch\_enable\_int;

logic core\_busy\_int;

logic clk\_gate\_core\_int;

logic [31:0] irq\_to\_core\_int;

logic lock\_fll\_int;

logic cfgreq\_fll\_int;

logic cfgack\_fll\_int;

logic [1:0] cfgad\_fll\_int;

logic [31:0] cfgd\_fll\_int;

logic [31:0] cfgq\_fll\_int;

logic cfgweb\_n\_fll\_int;

logic rstn\_int;

logic [31:0] boot\_addr\_int;

AXI\_BUS

#(

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

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

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

.AXI\_USER\_WIDTH ( `AXI\_USER\_WIDTH )

)

slaves[2:0]();

AXI\_BUS

#(

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

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

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

.AXI\_USER\_WIDTH ( `AXI\_USER\_WIDTH )

)

masters[2:0]();

DEBUG\_BUS

debug();

//----------------------------------------------------------------------------//

// Clock and reset generation

//----------------------------------------------------------------------------//

clk\_rst\_gen

clk\_rst\_gen\_i

(

.clk\_i ( clk ),

.rstn\_i ( rst\_n ),

.clk\_sel\_i ( clk\_sel\_i ),

.clk\_standalone\_i ( clk\_standalone\_i ),

.testmode\_i ( testmode\_i ),

.scan\_i ( 1'b0 ),

.scan\_o ( ),

.scan\_en\_i ( scan\_enable\_i ),

.fll\_req\_i ( cfgreq\_fll\_int ),

.fll\_wrn\_i ( cfgweb\_n\_fll\_int ),

.fll\_add\_i ( cfgad\_fll\_int ),

.fll\_data\_i ( cfgd\_fll\_int ),

.fll\_ack\_o ( cfgack\_fll\_int ),

.fll\_r\_data\_o ( cfgq\_fll\_int ),

.fll\_lock\_o ( lock\_fll\_int ),

.clk\_o ( clk\_int ),

.rstn\_o ( rstn\_int )

);

//----------------------------------------------------------------------------//

// Core region

//----------------------------------------------------------------------------//

core\_region

#(

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

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

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

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

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

.USE\_ZERO\_RISCY ( USE\_ZERO\_RISCY ),

.RISCY\_RV32F ( RISCY\_RV32F ),

.ZERO\_RV32M ( ZERO\_RV32M ),

.ZERO\_RV32E ( ZERO\_RV32E )

)

core\_region\_i

(

.clk ( clk\_int ),

.rst\_n ( rstn\_int ),

.testmode\_i ( testmode\_i ),

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

.irq\_i ( irq\_to\_core\_int ),

.core\_busy\_o ( core\_busy\_int ),

.clock\_gating\_i ( clk\_gate\_core\_int ),

.boot\_addr\_i ( boot\_addr\_int ),

.core\_master ( masters[0] ),

.dbg\_master ( masters[1] ),

.data\_slave ( slaves[1] ),

.instr\_slave ( slaves[0] ),

.debug ( debug ),

.tck\_i ( tck\_i ),

.trstn\_i ( trstn\_i ),

.tms\_i ( tms\_i ),

.tdi\_i ( tdi\_i ),

.tdo\_o ( tdo\_o )

);

//----------------------------------------------------------------------------//

// Peripherals

//----------------------------------------------------------------------------//

peripherals

#(

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

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

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

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

.AXI\_USER\_WIDTH ( `AXI\_USER\_WIDTH )

)

peripherals\_i

(

.clk\_i ( clk\_int ),

.rst\_n ( rstn\_int ),

.axi\_spi\_master ( masters[2] ),

.debug ( debug ),

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

.testmode\_i ( testmode\_i ),

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

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

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

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

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

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

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

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

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

.spi\_sdi3\_i ( spi\_sdi3\_i ),

.slave ( slaves[2] ),

.uart\_tx ( uart\_tx ),

.uart\_rx ( uart\_rx ),

.uart\_rts ( uart\_rts ),

.uart\_dtr ( uart\_dtr ),

.uart\_cts ( uart\_cts ),

.uart\_dsr ( uart\_dsr ),

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

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

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

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

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

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

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

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

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

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

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

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

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

.spi\_master\_sdi3 ( spi\_master\_sdi3\_i ),

.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 ),

.gpio\_in ( gpio\_in ),

.gpio\_out ( gpio\_out ),

.gpio\_dir ( gpio\_dir ),

.gpio\_padcfg ( gpio\_padcfg ),

.core\_busy\_i ( core\_busy\_int ),

.irq\_o ( irq\_to\_core\_int ),

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

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

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

.fll1\_req\_o ( cfgreq\_fll\_int ),

.fll1\_wrn\_o ( cfgweb\_n\_fll\_int ),

.fll1\_add\_o ( cfgad\_fll\_int ),

.fll1\_wdata\_o ( cfgd\_fll\_int ),

.fll1\_ack\_i ( cfgack\_fll\_int ),

.fll1\_rdata\_i ( cfgq\_fll\_int ),

.fll1\_lock\_i ( lock\_fll\_int ),

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

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

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

);

//----------------------------------------------------------------------------//

// Axi node

//----------------------------------------------------------------------------//

axi\_node\_intf\_wrap

#(

.NB\_MASTER ( 3 ),

.NB\_SLAVE ( 3 ),

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

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

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

.AXI\_USER\_WIDTH ( `AXI\_USER\_WIDTH )

)

axi\_interconnect\_i

(

.clk ( clk\_int ),

.rst\_n ( rstn\_int ),

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

.master ( slaves ),

.slave ( masters ),

.start\_addr\_i ( { 32'h1A10\_0000, 32'h0010\_0000, 32'h0000\_0000 } ),

.end\_addr\_i ( { 32'h1A11\_FFFF, 32'h001F\_FFFF, 32'h000F\_FFFF } )

);

endmodule