`include "apb\_bus.sv"

module periph\_bus\_wrap

#(

parameter APB\_ADDR\_WIDTH = 32,

parameter APB\_DATA\_WIDTH = 32

)

(

input logic clk\_i,

input logic rst\_ni,

APB\_BUS.Slave apb\_slave,

APB\_BUS.Master uart\_master,

APB\_BUS.Master gpio\_master,

APB\_BUS.Master spi\_master,

APB\_BUS.Master timer\_master,

APB\_BUS.Master event\_unit\_master,

APB\_BUS.Master i2c\_master,

APB\_BUS.Master fll\_master,

APB\_BUS.Master soc\_ctrl\_master,

APB\_BUS.Master debug\_master

);

localparam NB\_MASTER = `NB\_MASTER;

logic [NB\_MASTER-1:0][APB\_ADDR\_WIDTH-1:0] s\_start\_addr;

logic [NB\_MASTER-1:0][APB\_ADDR\_WIDTH-1:0] s\_end\_addr;

APB\_BUS

#(

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

.APB\_DATA\_WIDTH(APB\_DATA\_WIDTH)

)

s\_masters[NB\_MASTER-1:0]();

APB\_BUS

#(

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

.APB\_DATA\_WIDTH(APB\_DATA\_WIDTH)

)

s\_slave();

`APB\_ASSIGN\_SLAVE(s\_slave, apb\_slave);

`APB\_ASSIGN\_MASTER(s\_masters[0], uart\_master);

assign s\_start\_addr[0] = `UART\_START\_ADDR;

assign s\_end\_addr[0] = `UART\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[1], gpio\_master);

assign s\_start\_addr[1] = `GPIO\_START\_ADDR;

assign s\_end\_addr[1] = `GPIO\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[2], spi\_master);

assign s\_start\_addr[2] = `SPI\_START\_ADDR;

assign s\_end\_addr[2] = `SPI\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[3], timer\_master);

assign s\_start\_addr[3] = `TIMER\_START\_ADDR;

assign s\_end\_addr[3] = `TIMER\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[4], event\_unit\_master);

assign s\_start\_addr[4] = `EVENT\_UNIT\_START\_ADDR;

assign s\_end\_addr[4] = `EVENT\_UNIT\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[5], i2c\_master);

assign s\_start\_addr[5] = `I2C\_START\_ADDR;

assign s\_end\_addr[5] = `I2C\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[6], fll\_master);

assign s\_start\_addr[6] = `FLL\_START\_ADDR;

assign s\_end\_addr[6] = `FLL\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[7], soc\_ctrl\_master);

assign s\_start\_addr[7] = `SOC\_CTRL\_START\_ADDR;

assign s\_end\_addr[7] = `SOC\_CTRL\_END\_ADDR;

`APB\_ASSIGN\_MASTER(s\_masters[8], debug\_master);

assign s\_start\_addr[8] = `DEBUG\_START\_ADDR;

assign s\_end\_addr[8] = `DEBUG\_END\_ADDR;

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SOC BUS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

apb\_node\_wrap

#(

.NB\_MASTER ( NB\_MASTER ),

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

.APB\_DATA\_WIDTH ( APB\_DATA\_WIDTH )

)

apb\_node\_wrap\_i

(

.clk\_i ( clk\_i ),

.rst\_ni ( rst\_ni ),

.apb\_slave ( s\_slave ),

.apb\_masters ( s\_masters ),

.start\_addr\_i ( s\_start\_addr ),

.end\_addr\_i ( s\_end\_addr )

);

endmodule