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

// RTL generated by Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC

// Version: 2017.2

// Copyright (C) 1986-2017 Xilinx, Inc. All Rights Reserved.

//

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

#include "matrixmul.h"

#include "AESL\_pkg.h"

using namespace std;

namespace ap\_rtl {

const sc\_logic matrixmul::ap\_const\_logic\_1 = sc\_dt::Log\_1;

const sc\_logic matrixmul::ap\_const\_logic\_0 = sc\_dt::Log\_0;

const sc\_lv<4> matrixmul::ap\_ST\_fsm\_state1 = "1";

const sc\_lv<4> matrixmul::ap\_ST\_fsm\_pp0\_stage0 = "10";

const sc\_lv<4> matrixmul::ap\_ST\_fsm\_pp0\_stage1 = "100";

const sc\_lv<4> matrixmul::ap\_ST\_fsm\_state6 = "1000";

const sc\_lv<32> matrixmul::ap\_const\_lv32\_0 = "00000000000000000000000000000000";

const bool matrixmul::ap\_const\_boolean\_1 = true;

const sc\_lv<32> matrixmul::ap\_const\_lv32\_2 = "10";

const bool matrixmul::ap\_const\_boolean\_0 = false;

const sc\_lv<1> matrixmul::ap\_const\_lv1\_0 = "0";

const sc\_lv<32> matrixmul::ap\_const\_lv32\_1 = "1";

const sc\_lv<1> matrixmul::ap\_const\_lv1\_1 = "1";

const sc\_lv<4> matrixmul::ap\_const\_lv4\_0 = "0000";

const sc\_lv<2> matrixmul::ap\_const\_lv2\_0 = "00";

const sc\_lv<64> matrixmul::ap\_const\_lv64\_0 = "0000000000000000000000000000000000000000000000000000000000000000";

const sc\_lv<64> matrixmul::ap\_const\_lv64\_1 = "1";

const sc\_lv<64> matrixmul::ap\_const\_lv64\_2 = "10";

const sc\_lv<4> matrixmul::ap\_const\_lv4\_9 = "1001";

const sc\_lv<4> matrixmul::ap\_const\_lv4\_1 = "1";

const sc\_lv<2> matrixmul::ap\_const\_lv2\_1 = "1";

const sc\_lv<2> matrixmul::ap\_const\_lv2\_3 = "11";

const sc\_lv<32> matrixmul::ap\_const\_lv32\_3 = "11";

matrixmul::matrixmul(sc\_module\_name name) : sc\_module(name), mVcdFile(0) {

matrixmul\_mux\_32\_bkb\_U1 = new matrixmul\_mux\_32\_bkb<1,1,8,8,8,2,8>("matrixmul\_mux\_32\_bkb\_U1");

matrixmul\_mux\_32\_bkb\_U1->din1(b\_0\_q0);

matrixmul\_mux\_32\_bkb\_U1->din2(b\_1\_q0);

matrixmul\_mux\_32\_bkb\_U1->din3(b\_2\_q0);

matrixmul\_mux\_32\_bkb\_U1->din4(j\_mid2\_reg\_446);

matrixmul\_mux\_32\_bkb\_U1->dout(grp\_fu\_238\_p5);

matrixmul\_mux\_32\_bkb\_U2 = new matrixmul\_mux\_32\_bkb<1,1,8,8,8,2,8>("matrixmul\_mux\_32\_bkb\_U2");

matrixmul\_mux\_32\_bkb\_U2->din1(b\_0\_q1);

matrixmul\_mux\_32\_bkb\_U2->din2(b\_1\_q1);

matrixmul\_mux\_32\_bkb\_U2->din3(b\_2\_q1);

matrixmul\_mux\_32\_bkb\_U2->din4(j\_mid2\_reg\_446);

matrixmul\_mux\_32\_bkb\_U2->dout(tmp\_9\_fu\_300\_p5);

matrixmul\_mac\_mulcud\_U3 = new matrixmul\_mac\_mulcud<1,1,8,8,16,16>("matrixmul\_mac\_mulcud\_U3");

matrixmul\_mac\_mulcud\_U3->din0(reg\_249);

matrixmul\_mac\_mulcud\_U3->din1(a\_0\_load\_reg\_476);

matrixmul\_mac\_mulcud\_U3->din2(grp\_fu\_376\_p2);

matrixmul\_mac\_mulcud\_U3->dout(grp\_fu\_376\_p3);

matrixmul\_mac\_muldEe\_U4 = new matrixmul\_mac\_muldEe<1,1,8,8,16,16>("matrixmul\_mac\_muldEe\_U4");

matrixmul\_mac\_muldEe\_U4->din0(reg\_249);

matrixmul\_mac\_muldEe\_U4->din1(a\_1\_load\_reg\_481);

matrixmul\_mac\_muldEe\_U4->din2(tmp1\_reg\_501);

matrixmul\_mac\_muldEe\_U4->dout(grp\_fu\_384\_p3);

SC\_METHOD(thread\_ap\_clk\_no\_reset\_);

dont\_initialize();

sensitive << ( ap\_clk.pos() );

SC\_METHOD(thread\_a\_0\_address0);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

sensitive << ( tmp\_mid2\_fu\_293\_p1 );

SC\_METHOD(thread\_a\_0\_ce0);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_a\_1\_address0);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

sensitive << ( tmp\_mid2\_fu\_293\_p1 );

SC\_METHOD(thread\_a\_1\_ce0);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_a\_2\_address0);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

sensitive << ( tmp\_mid2\_fu\_293\_p1 );

SC\_METHOD(thread\_a\_2\_ce0);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_ap\_CS\_fsm\_pp0\_stage0);

sensitive << ( ap\_CS\_fsm );

SC\_METHOD(thread\_ap\_CS\_fsm\_pp0\_stage1);

sensitive << ( ap\_CS\_fsm );

SC\_METHOD(thread\_ap\_CS\_fsm\_state1);

sensitive << ( ap\_CS\_fsm );

SC\_METHOD(thread\_ap\_CS\_fsm\_state6);

sensitive << ( ap\_CS\_fsm );

SC\_METHOD(thread\_ap\_block\_pp0\_stage0\_flag00000000);

SC\_METHOD(thread\_ap\_block\_pp0\_stage0\_flag00011001);

SC\_METHOD(thread\_ap\_block\_pp0\_stage0\_flag00011011);

SC\_METHOD(thread\_ap\_block\_pp0\_stage1\_flag00000000);

SC\_METHOD(thread\_ap\_block\_pp0\_stage1\_flag00011001);

SC\_METHOD(thread\_ap\_block\_pp0\_stage1\_flag00011011);

SC\_METHOD(thread\_ap\_block\_state2\_pp0\_stage0\_iter0);

SC\_METHOD(thread\_ap\_block\_state3\_pp0\_stage1\_iter0);

SC\_METHOD(thread\_ap\_block\_state4\_pp0\_stage0\_iter1);

SC\_METHOD(thread\_ap\_block\_state5\_pp0\_stage1\_iter1);

SC\_METHOD(thread\_ap\_condition\_pp0\_exit\_iter0\_state2);

sensitive << ( exitcond\_flatten\_fu\_253\_p2 );

SC\_METHOD(thread\_ap\_done);

sensitive << ( ap\_CS\_fsm\_state6 );

SC\_METHOD(thread\_ap\_enable\_pp0);

sensitive << ( ap\_idle\_pp0 );

SC\_METHOD(thread\_ap\_idle);

sensitive << ( ap\_start );

sensitive << ( ap\_CS\_fsm\_state1 );

SC\_METHOD(thread\_ap\_idle\_pp0);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

SC\_METHOD(thread\_ap\_ready);

sensitive << ( ap\_CS\_fsm\_state6 );

SC\_METHOD(thread\_b\_0\_address0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00000000 );

SC\_METHOD(thread\_b\_0\_address1);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_b\_0\_ce0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00011001 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_b\_0\_ce1);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_b\_1\_address0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00000000 );

SC\_METHOD(thread\_b\_1\_address1);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_b\_1\_ce0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00011001 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_b\_1\_ce1);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_b\_2\_address0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00000000 );

SC\_METHOD(thread\_b\_2\_address1);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_b\_2\_ce0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00011001 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_b\_2\_ce1);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

SC\_METHOD(thread\_exitcond\_flatten\_fu\_253\_p2);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

sensitive << ( indvar\_flatten\_phi\_fu\_209\_p4 );

SC\_METHOD(thread\_exitcond\_fu\_271\_p2);

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011001 );

sensitive << ( exitcond\_flatten\_fu\_253\_p2 );

sensitive << ( j\_phi\_fu\_231\_p4 );

SC\_METHOD(thread\_grp\_fu\_376\_p2);

sensitive << ( exitcond\_flatten\_reg\_437 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

sensitive << ( tmp\_7\_2\_fu\_329\_p0 );

sensitive << ( tmp\_7\_2\_fu\_329\_p1 );

SC\_METHOD(thread\_i\_1\_fu\_265\_p2);

sensitive << ( i\_phi\_fu\_220\_p4 );

SC\_METHOD(thread\_i\_phi\_fu\_220\_p4);

sensitive << ( i\_reg\_216 );

sensitive << ( exitcond\_flatten\_reg\_437 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( tmp\_mid2\_v\_reg\_454 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_indvar\_flatten\_next\_fu\_259\_p2);

sensitive << ( indvar\_flatten\_phi\_fu\_209\_p4 );

SC\_METHOD(thread\_indvar\_flatten\_phi\_fu\_209\_p4);

sensitive << ( indvar\_flatten\_reg\_205 );

sensitive << ( exitcond\_flatten\_reg\_437 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( indvar\_flatten\_next\_reg\_441 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_j\_1\_fu\_311\_p2);

sensitive << ( j\_mid2\_reg\_446 );

SC\_METHOD(thread\_j\_mid2\_fu\_277\_p3);

sensitive << ( j\_phi\_fu\_231\_p4 );

sensitive << ( exitcond\_fu\_271\_p2 );

SC\_METHOD(thread\_j\_phi\_fu\_231\_p4);

sensitive << ( j\_reg\_227 );

sensitive << ( exitcond\_flatten\_reg\_437 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( j\_1\_reg\_496 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_p\_shl\_cast\_fu\_345\_p1);

sensitive << ( tmp\_fu\_338\_p3 );

SC\_METHOD(thread\_res\_address0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( tmp\_10\_cast\_fu\_364\_p1 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00000000 );

SC\_METHOD(thread\_res\_ce0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00011001 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

SC\_METHOD(thread\_res\_d0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00000000 );

sensitive << ( grp\_fu\_384\_p3 );

SC\_METHOD(thread\_res\_we0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00011001 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( ap\_reg\_pp0\_iter1\_exitcond\_flatten\_reg\_437 );

SC\_METHOD(thread\_tmp\_10\_cast\_fu\_364\_p1);

sensitive << ( tmp\_s\_fu\_358\_p2 );

SC\_METHOD(thread\_tmp\_1\_fu\_349\_p2);

sensitive << ( p\_shl\_cast\_fu\_345\_p1 );

sensitive << ( tmp\_mid2\_cast\_fu\_335\_p1 );

SC\_METHOD(thread\_tmp\_2\_cast\_fu\_355\_p1);

sensitive << ( ap\_reg\_pp0\_iter1\_j\_mid2\_reg\_446 );

SC\_METHOD(thread\_tmp\_7\_2\_fu\_329\_p0);

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( tmp\_9\_reg\_491 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_tmp\_7\_2\_fu\_329\_p1);

sensitive << ( ap\_CS\_fsm\_pp0\_stage0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( a\_2\_load\_reg\_486 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00000000 );

SC\_METHOD(thread\_tmp\_fu\_338\_p3);

sensitive << ( ap\_reg\_pp0\_iter1\_tmp\_mid2\_v\_reg\_454 );

SC\_METHOD(thread\_tmp\_mid2\_cast\_fu\_335\_p1);

sensitive << ( ap\_reg\_pp0\_iter1\_tmp\_mid2\_v\_reg\_454 );

SC\_METHOD(thread\_tmp\_mid2\_fu\_293\_p1);

sensitive << ( tmp\_mid2\_v\_fu\_285\_p3 );

SC\_METHOD(thread\_tmp\_mid2\_v\_fu\_285\_p3);

sensitive << ( i\_phi\_fu\_220\_p4 );

sensitive << ( exitcond\_fu\_271\_p2 );

sensitive << ( i\_1\_fu\_265\_p2 );

SC\_METHOD(thread\_tmp\_s\_fu\_358\_p2);

sensitive << ( tmp\_1\_fu\_349\_p2 );

sensitive << ( tmp\_2\_cast\_fu\_355\_p1 );

SC\_METHOD(thread\_ap\_NS\_fsm);

sensitive << ( ap\_start );

sensitive << ( ap\_CS\_fsm );

sensitive << ( ap\_CS\_fsm\_state1 );

sensitive << ( ap\_CS\_fsm\_pp0\_stage1 );

sensitive << ( ap\_enable\_reg\_pp0\_iter0 );

sensitive << ( ap\_enable\_reg\_pp0\_iter1 );

sensitive << ( exitcond\_flatten\_fu\_253\_p2 );

sensitive << ( ap\_block\_pp0\_stage0\_flag00011011 );

sensitive << ( ap\_block\_pp0\_stage1\_flag00011011 );

SC\_THREAD(thread\_hdltv\_gen);

sensitive << ( ap\_clk.pos() );

ap\_CS\_fsm = "0001";

ap\_enable\_reg\_pp0\_iter0 = SC\_LOGIC\_0;

ap\_enable\_reg\_pp0\_iter1 = SC\_LOGIC\_0;

static int apTFileNum = 0;

stringstream apTFilenSS;

apTFilenSS << "matrixmul\_sc\_trace\_" << apTFileNum ++;

string apTFn = apTFilenSS.str();

mVcdFile = sc\_create\_vcd\_trace\_file(apTFn.c\_str());

mVcdFile->set\_time\_unit(1, SC\_PS);

if (1) {

#ifdef \_\_HLS\_TRACE\_LEVEL\_PORT\_\_

sc\_trace(mVcdFile, ap\_clk, "(port)ap\_clk");

sc\_trace(mVcdFile, ap\_rst, "(port)ap\_rst");

sc\_trace(mVcdFile, ap\_start, "(port)ap\_start");

sc\_trace(mVcdFile, ap\_done, "(port)ap\_done");

sc\_trace(mVcdFile, ap\_idle, "(port)ap\_idle");

sc\_trace(mVcdFile, ap\_ready, "(port)ap\_ready");

sc\_trace(mVcdFile, a\_0\_address0, "(port)a\_0\_address0");

sc\_trace(mVcdFile, a\_0\_ce0, "(port)a\_0\_ce0");

sc\_trace(mVcdFile, a\_0\_q0, "(port)a\_0\_q0");

sc\_trace(mVcdFile, a\_1\_address0, "(port)a\_1\_address0");

sc\_trace(mVcdFile, a\_1\_ce0, "(port)a\_1\_ce0");

sc\_trace(mVcdFile, a\_1\_q0, "(port)a\_1\_q0");

sc\_trace(mVcdFile, a\_2\_address0, "(port)a\_2\_address0");

sc\_trace(mVcdFile, a\_2\_ce0, "(port)a\_2\_ce0");

sc\_trace(mVcdFile, a\_2\_q0, "(port)a\_2\_q0");

sc\_trace(mVcdFile, b\_0\_address0, "(port)b\_0\_address0");

sc\_trace(mVcdFile, b\_0\_ce0, "(port)b\_0\_ce0");

sc\_trace(mVcdFile, b\_0\_q0, "(port)b\_0\_q0");

sc\_trace(mVcdFile, b\_0\_address1, "(port)b\_0\_address1");

sc\_trace(mVcdFile, b\_0\_ce1, "(port)b\_0\_ce1");

sc\_trace(mVcdFile, b\_0\_q1, "(port)b\_0\_q1");

sc\_trace(mVcdFile, b\_1\_address0, "(port)b\_1\_address0");

sc\_trace(mVcdFile, b\_1\_ce0, "(port)b\_1\_ce0");

sc\_trace(mVcdFile, b\_1\_q0, "(port)b\_1\_q0");

sc\_trace(mVcdFile, b\_1\_address1, "(port)b\_1\_address1");

sc\_trace(mVcdFile, b\_1\_ce1, "(port)b\_1\_ce1");

sc\_trace(mVcdFile, b\_1\_q1, "(port)b\_1\_q1");

sc\_trace(mVcdFile, b\_2\_address0, "(port)b\_2\_address0");

sc\_trace(mVcdFile, b\_2\_ce0, "(port)b\_2\_ce0");

sc\_trace(mVcdFile, b\_2\_q0, "(port)b\_2\_q0");

sc\_trace(mVcdFile, b\_2\_address1, "(port)b\_2\_address1");

sc\_trace(mVcdFile, b\_2\_ce1, "(port)b\_2\_ce1");

sc\_trace(mVcdFile, b\_2\_q1, "(port)b\_2\_q1");

sc\_trace(mVcdFile, res\_address0, "(port)res\_address0");

sc\_trace(mVcdFile, res\_ce0, "(port)res\_ce0");

sc\_trace(mVcdFile, res\_we0, "(port)res\_we0");

sc\_trace(mVcdFile, res\_d0, "(port)res\_d0");

#endif

#ifdef \_\_HLS\_TRACE\_LEVEL\_INT\_\_

sc\_trace(mVcdFile, ap\_CS\_fsm, "ap\_CS\_fsm");

sc\_trace(mVcdFile, ap\_CS\_fsm\_state1, "ap\_CS\_fsm\_state1");

sc\_trace(mVcdFile, indvar\_flatten\_reg\_205, "indvar\_flatten\_reg\_205");

sc\_trace(mVcdFile, i\_reg\_216, "i\_reg\_216");

sc\_trace(mVcdFile, j\_reg\_227, "j\_reg\_227");

sc\_trace(mVcdFile, grp\_fu\_238\_p5, "grp\_fu\_238\_p5");

sc\_trace(mVcdFile, reg\_249, "reg\_249");

sc\_trace(mVcdFile, ap\_CS\_fsm\_pp0\_stage1, "ap\_CS\_fsm\_pp0\_stage1");

sc\_trace(mVcdFile, ap\_enable\_reg\_pp0\_iter0, "ap\_enable\_reg\_pp0\_iter0");

sc\_trace(mVcdFile, ap\_block\_state3\_pp0\_stage1\_iter0, "ap\_block\_state3\_pp0\_stage1\_iter0");

sc\_trace(mVcdFile, ap\_block\_state5\_pp0\_stage1\_iter1, "ap\_block\_state5\_pp0\_stage1\_iter1");

sc\_trace(mVcdFile, ap\_block\_pp0\_stage1\_flag00011001, "ap\_block\_pp0\_stage1\_flag00011001");

sc\_trace(mVcdFile, exitcond\_flatten\_reg\_437, "exitcond\_flatten\_reg\_437");

sc\_trace(mVcdFile, ap\_CS\_fsm\_pp0\_stage0, "ap\_CS\_fsm\_pp0\_stage0");

sc\_trace(mVcdFile, ap\_enable\_reg\_pp0\_iter1, "ap\_enable\_reg\_pp0\_iter1");

sc\_trace(mVcdFile, ap\_block\_state2\_pp0\_stage0\_iter0, "ap\_block\_state2\_pp0\_stage0\_iter0");

sc\_trace(mVcdFile, ap\_block\_state4\_pp0\_stage0\_iter1, "ap\_block\_state4\_pp0\_stage0\_iter1");

sc\_trace(mVcdFile, ap\_block\_pp0\_stage0\_flag00011001, "ap\_block\_pp0\_stage0\_flag00011001");

sc\_trace(mVcdFile, exitcond\_flatten\_fu\_253\_p2, "exitcond\_flatten\_fu\_253\_p2");

sc\_trace(mVcdFile, ap\_reg\_pp0\_iter1\_exitcond\_flatten\_reg\_437, "ap\_reg\_pp0\_iter1\_exitcond\_flatten\_reg\_437");

sc\_trace(mVcdFile, indvar\_flatten\_next\_fu\_259\_p2, "indvar\_flatten\_next\_fu\_259\_p2");

sc\_trace(mVcdFile, indvar\_flatten\_next\_reg\_441, "indvar\_flatten\_next\_reg\_441");

sc\_trace(mVcdFile, j\_mid2\_fu\_277\_p3, "j\_mid2\_fu\_277\_p3");

sc\_trace(mVcdFile, j\_mid2\_reg\_446, "j\_mid2\_reg\_446");

sc\_trace(mVcdFile, ap\_reg\_pp0\_iter1\_j\_mid2\_reg\_446, "ap\_reg\_pp0\_iter1\_j\_mid2\_reg\_446");

sc\_trace(mVcdFile, tmp\_mid2\_v\_fu\_285\_p3, "tmp\_mid2\_v\_fu\_285\_p3");

sc\_trace(mVcdFile, tmp\_mid2\_v\_reg\_454, "tmp\_mid2\_v\_reg\_454");

sc\_trace(mVcdFile, ap\_reg\_pp0\_iter1\_tmp\_mid2\_v\_reg\_454, "ap\_reg\_pp0\_iter1\_tmp\_mid2\_v\_reg\_454");

sc\_trace(mVcdFile, a\_0\_load\_reg\_476, "a\_0\_load\_reg\_476");

sc\_trace(mVcdFile, a\_1\_load\_reg\_481, "a\_1\_load\_reg\_481");

sc\_trace(mVcdFile, a\_2\_load\_reg\_486, "a\_2\_load\_reg\_486");

sc\_trace(mVcdFile, tmp\_9\_fu\_300\_p5, "tmp\_9\_fu\_300\_p5");

sc\_trace(mVcdFile, tmp\_9\_reg\_491, "tmp\_9\_reg\_491");

sc\_trace(mVcdFile, j\_1\_fu\_311\_p2, "j\_1\_fu\_311\_p2");

sc\_trace(mVcdFile, j\_1\_reg\_496, "j\_1\_reg\_496");

sc\_trace(mVcdFile, grp\_fu\_376\_p3, "grp\_fu\_376\_p3");

sc\_trace(mVcdFile, tmp1\_reg\_501, "tmp1\_reg\_501");

sc\_trace(mVcdFile, ap\_block\_pp0\_stage0\_flag00011011, "ap\_block\_pp0\_stage0\_flag00011011");

sc\_trace(mVcdFile, ap\_condition\_pp0\_exit\_iter0\_state2, "ap\_condition\_pp0\_exit\_iter0\_state2");

sc\_trace(mVcdFile, ap\_block\_pp0\_stage1\_flag00011011, "ap\_block\_pp0\_stage1\_flag00011011");

sc\_trace(mVcdFile, indvar\_flatten\_phi\_fu\_209\_p4, "indvar\_flatten\_phi\_fu\_209\_p4");

sc\_trace(mVcdFile, ap\_block\_pp0\_stage0\_flag00000000, "ap\_block\_pp0\_stage0\_flag00000000");

sc\_trace(mVcdFile, i\_phi\_fu\_220\_p4, "i\_phi\_fu\_220\_p4");

sc\_trace(mVcdFile, j\_phi\_fu\_231\_p4, "j\_phi\_fu\_231\_p4");

sc\_trace(mVcdFile, tmp\_mid2\_fu\_293\_p1, "tmp\_mid2\_fu\_293\_p1");

sc\_trace(mVcdFile, tmp\_10\_cast\_fu\_364\_p1, "tmp\_10\_cast\_fu\_364\_p1");

sc\_trace(mVcdFile, ap\_block\_pp0\_stage1\_flag00000000, "ap\_block\_pp0\_stage1\_flag00000000");

sc\_trace(mVcdFile, grp\_fu\_384\_p3, "grp\_fu\_384\_p3");

sc\_trace(mVcdFile, exitcond\_fu\_271\_p2, "exitcond\_fu\_271\_p2");

sc\_trace(mVcdFile, i\_1\_fu\_265\_p2, "i\_1\_fu\_265\_p2");

sc\_trace(mVcdFile, tmp\_7\_2\_fu\_329\_p0, "tmp\_7\_2\_fu\_329\_p0");

sc\_trace(mVcdFile, tmp\_7\_2\_fu\_329\_p1, "tmp\_7\_2\_fu\_329\_p1");

sc\_trace(mVcdFile, tmp\_fu\_338\_p3, "tmp\_fu\_338\_p3");

sc\_trace(mVcdFile, p\_shl\_cast\_fu\_345\_p1, "p\_shl\_cast\_fu\_345\_p1");

sc\_trace(mVcdFile, tmp\_mid2\_cast\_fu\_335\_p1, "tmp\_mid2\_cast\_fu\_335\_p1");

sc\_trace(mVcdFile, tmp\_1\_fu\_349\_p2, "tmp\_1\_fu\_349\_p2");

sc\_trace(mVcdFile, tmp\_2\_cast\_fu\_355\_p1, "tmp\_2\_cast\_fu\_355\_p1");

sc\_trace(mVcdFile, tmp\_s\_fu\_358\_p2, "tmp\_s\_fu\_358\_p2");

sc\_trace(mVcdFile, grp\_fu\_376\_p2, "grp\_fu\_376\_p2");

sc\_trace(mVcdFile, ap\_CS\_fsm\_state6, "ap\_CS\_fsm\_state6");

sc\_trace(mVcdFile, ap\_NS\_fsm, "ap\_NS\_fsm");

sc\_trace(mVcdFile, ap\_idle\_pp0, "ap\_idle\_pp0");

sc\_trace(mVcdFile, ap\_enable\_pp0, "ap\_enable\_pp0");

#endif

}

mHdltvinHandle.open("matrixmul.hdltvin.dat");

mHdltvoutHandle.open("matrixmul.hdltvout.dat");

}

matrixmul::~matrixmul() {

if (mVcdFile)

sc\_close\_vcd\_trace\_file(mVcdFile);

mHdltvinHandle << "] " << endl;

mHdltvoutHandle << "] " << endl;

mHdltvinHandle.close();

mHdltvoutHandle.close();

delete matrixmul\_mux\_32\_bkb\_U1;

delete matrixmul\_mux\_32\_bkb\_U2;

delete matrixmul\_mac\_mulcud\_U3;

delete matrixmul\_mac\_muldEe\_U4;

}

void matrixmul::thread\_ap\_clk\_no\_reset\_() {

if ( ap\_rst.read() == ap\_const\_logic\_1) {

ap\_CS\_fsm = ap\_ST\_fsm\_state1;

} else {

ap\_CS\_fsm = ap\_NS\_fsm.read();

}

if ( ap\_rst.read() == ap\_const\_logic\_1) {

ap\_enable\_reg\_pp0\_iter0 = ap\_const\_logic\_0;

} else {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011011.read(), ap\_const\_boolean\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_condition\_pp0\_exit\_iter0\_state2.read()))) {

ap\_enable\_reg\_pp0\_iter0 = ap\_const\_logic\_0;

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state1.read()) &&

esl\_seteq<1,1,1>(ap\_start.read(), ap\_const\_logic\_1))) {

ap\_enable\_reg\_pp0\_iter0 = ap\_const\_logic\_1;

}

}

if ( ap\_rst.read() == ap\_const\_logic\_1) {

ap\_enable\_reg\_pp0\_iter1 = ap\_const\_logic\_0;

} else {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_condition\_pp0\_exit\_iter0\_state2.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011011.read(), ap\_const\_boolean\_0))) {

ap\_enable\_reg\_pp0\_iter1 = (ap\_condition\_pp0\_exit\_iter0\_state2.read() ^ ap\_const\_logic\_1);

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011011.read(), ap\_const\_boolean\_0))) {

ap\_enable\_reg\_pp0\_iter1 = ap\_enable\_reg\_pp0\_iter0.read();

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state1.read()) &&

esl\_seteq<1,1,1>(ap\_start.read(), ap\_const\_logic\_1))) {

ap\_enable\_reg\_pp0\_iter1 = ap\_const\_logic\_0;

}

}

if ((esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

i\_reg\_216 = tmp\_mid2\_v\_reg\_454.read();

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state1.read()) &&

esl\_seteq<1,1,1>(ap\_start.read(), ap\_const\_logic\_1))) {

i\_reg\_216 = ap\_const\_lv2\_0;

}

if ((esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

indvar\_flatten\_reg\_205 = indvar\_flatten\_next\_reg\_441.read();

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state1.read()) &&

esl\_seteq<1,1,1>(ap\_start.read(), ap\_const\_logic\_1))) {

indvar\_flatten\_reg\_205 = ap\_const\_lv4\_0;

}

if ((esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

j\_reg\_227 = j\_1\_reg\_496.read();

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state1.read()) &&

esl\_seteq<1,1,1>(ap\_start.read(), ap\_const\_logic\_1))) {

j\_reg\_227 = ap\_const\_lv2\_0;

}

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0))) {

a\_0\_load\_reg\_476 = a\_0\_q0.read();

a\_1\_load\_reg\_481 = a\_1\_q0.read();

a\_2\_load\_reg\_486 = a\_2\_q0.read();

tmp\_9\_reg\_491 = tmp\_9\_fu\_300\_p5.read();

}

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

ap\_reg\_pp0\_iter1\_exitcond\_flatten\_reg\_437 = exitcond\_flatten\_reg\_437.read();

ap\_reg\_pp0\_iter1\_j\_mid2\_reg\_446 = j\_mid2\_reg\_446.read();

ap\_reg\_pp0\_iter1\_tmp\_mid2\_v\_reg\_454 = tmp\_mid2\_v\_reg\_454.read();

exitcond\_flatten\_reg\_437 = exitcond\_flatten\_fu\_253\_p2.read();

}

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) && esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

indvar\_flatten\_next\_reg\_441 = indvar\_flatten\_next\_fu\_259\_p2.read();

}

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) && esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0))) {

j\_1\_reg\_496 = j\_1\_fu\_311\_p2.read();

}

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(ap\_const\_lv1\_0, exitcond\_flatten\_fu\_253\_p2.read()))) {

j\_mid2\_reg\_446 = j\_mid2\_fu\_277\_p3.read();

}

if (((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0) &&

esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0)) || (esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0)))) {

reg\_249 = grp\_fu\_238\_p5.read();

}

if ((esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) && esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) && esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

tmp1\_reg\_501 = grp\_fu\_376\_p3.read();

}

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) && esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(ap\_const\_lv1\_0, exitcond\_flatten\_fu\_253\_p2.read()))) {

tmp\_mid2\_v\_reg\_454 = tmp\_mid2\_v\_fu\_285\_p3.read();

}

}

void matrixmul::thread\_a\_0\_address0() {

a\_0\_address0 = (sc\_lv<2>) (tmp\_mid2\_fu\_293\_p1.read());

}

void matrixmul::thread\_a\_0\_ce0() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

a\_0\_ce0 = ap\_const\_logic\_1;

} else {

a\_0\_ce0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_a\_1\_address0() {

a\_1\_address0 = (sc\_lv<2>) (tmp\_mid2\_fu\_293\_p1.read());

}

void matrixmul::thread\_a\_1\_ce0() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

a\_1\_ce0 = ap\_const\_logic\_1;

} else {

a\_1\_ce0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_a\_2\_address0() {

a\_2\_address0 = (sc\_lv<2>) (tmp\_mid2\_fu\_293\_p1.read());

}

void matrixmul::thread\_a\_2\_ce0() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

a\_2\_ce0 = ap\_const\_logic\_1;

} else {

a\_2\_ce0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_ap\_CS\_fsm\_pp0\_stage0() {

ap\_CS\_fsm\_pp0\_stage0 = ap\_CS\_fsm.read()[1];

}

void matrixmul::thread\_ap\_CS\_fsm\_pp0\_stage1() {

ap\_CS\_fsm\_pp0\_stage1 = ap\_CS\_fsm.read()[2];

}

void matrixmul::thread\_ap\_CS\_fsm\_state1() {

ap\_CS\_fsm\_state1 = ap\_CS\_fsm.read()[0];

}

void matrixmul::thread\_ap\_CS\_fsm\_state6() {

ap\_CS\_fsm\_state6 = ap\_CS\_fsm.read()[3];

}

void matrixmul::thread\_ap\_block\_pp0\_stage0\_flag00000000() {

ap\_block\_pp0\_stage0\_flag00000000 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_pp0\_stage0\_flag00011001() {

ap\_block\_pp0\_stage0\_flag00011001 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_pp0\_stage0\_flag00011011() {

ap\_block\_pp0\_stage0\_flag00011011 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_pp0\_stage1\_flag00000000() {

ap\_block\_pp0\_stage1\_flag00000000 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_pp0\_stage1\_flag00011001() {

ap\_block\_pp0\_stage1\_flag00011001 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_pp0\_stage1\_flag00011011() {

ap\_block\_pp0\_stage1\_flag00011011 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_state2\_pp0\_stage0\_iter0() {

ap\_block\_state2\_pp0\_stage0\_iter0 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_state3\_pp0\_stage1\_iter0() {

ap\_block\_state3\_pp0\_stage1\_iter0 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_state4\_pp0\_stage0\_iter1() {

ap\_block\_state4\_pp0\_stage0\_iter1 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_state5\_pp0\_stage1\_iter1() {

ap\_block\_state5\_pp0\_stage1\_iter1 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_condition\_pp0\_exit\_iter0\_state2() {

if (esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_253\_p2.read(), ap\_const\_lv1\_1)) {

ap\_condition\_pp0\_exit\_iter0\_state2 = ap\_const\_logic\_1;

} else {

ap\_condition\_pp0\_exit\_iter0\_state2 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_ap\_done() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state6.read())) {

ap\_done = ap\_const\_logic\_1;

} else {

ap\_done = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_ap\_enable\_pp0() {

ap\_enable\_pp0 = (ap\_idle\_pp0.read() ^ ap\_const\_logic\_1);

}

void matrixmul::thread\_ap\_idle() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_0, ap\_start.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state1.read()))) {

ap\_idle = ap\_const\_logic\_1;

} else {

ap\_idle = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_ap\_idle\_pp0() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_0, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_0, ap\_enable\_reg\_pp0\_iter1.read()))) {

ap\_idle\_pp0 = ap\_const\_logic\_1;

} else {

ap\_idle\_pp0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_ap\_ready() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state6.read())) {

ap\_ready = ap\_const\_logic\_1;

} else {

ap\_ready = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_b\_0\_address0() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read())) {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00000000.read(), ap\_const\_boolean\_0))) {

b\_0\_address0 = (sc\_lv<2>) (ap\_const\_lv64\_1);

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00000000.read(), ap\_const\_boolean\_0))) {

b\_0\_address0 = (sc\_lv<2>) (ap\_const\_lv64\_0);

} else {

b\_0\_address0 = "XX";

}

} else {

b\_0\_address0 = "XX";

}

}

void matrixmul::thread\_b\_0\_address1() {

b\_0\_address1 = (sc\_lv<2>) (ap\_const\_lv64\_2);

}

void matrixmul::thread\_b\_0\_ce0() {

if (((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0)) ||

(esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0)))) {

b\_0\_ce0 = ap\_const\_logic\_1;

} else {

b\_0\_ce0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_b\_0\_ce1() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

b\_0\_ce1 = ap\_const\_logic\_1;

} else {

b\_0\_ce1 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_b\_1\_address0() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read())) {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00000000.read(), ap\_const\_boolean\_0))) {

b\_1\_address0 = (sc\_lv<2>) (ap\_const\_lv64\_1);

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00000000.read(), ap\_const\_boolean\_0))) {

b\_1\_address0 = (sc\_lv<2>) (ap\_const\_lv64\_0);

} else {

b\_1\_address0 = "XX";

}

} else {

b\_1\_address0 = "XX";

}

}

void matrixmul::thread\_b\_1\_address1() {

b\_1\_address1 = (sc\_lv<2>) (ap\_const\_lv64\_2);

}

void matrixmul::thread\_b\_1\_ce0() {

if (((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0)) ||

(esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0)))) {

b\_1\_ce0 = ap\_const\_logic\_1;

} else {

b\_1\_ce0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_b\_1\_ce1() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

b\_1\_ce1 = ap\_const\_logic\_1;

} else {

b\_1\_ce1 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_b\_2\_address0() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read())) {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00000000.read(), ap\_const\_boolean\_0))) {

b\_2\_address0 = (sc\_lv<2>) (ap\_const\_lv64\_1);

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00000000.read(), ap\_const\_boolean\_0))) {

b\_2\_address0 = (sc\_lv<2>) (ap\_const\_lv64\_0);

} else {

b\_2\_address0 = "XX";

}

} else {

b\_2\_address0 = "XX";

}

}

void matrixmul::thread\_b\_2\_address1() {

b\_2\_address1 = (sc\_lv<2>) (ap\_const\_lv64\_2);

}

void matrixmul::thread\_b\_2\_ce0() {

if (((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0)) ||

(esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0)))) {

b\_2\_ce0 = ap\_const\_logic\_1;

} else {

b\_2\_ce0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_b\_2\_ce1() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0))) {

b\_2\_ce1 = ap\_const\_logic\_1;

} else {

b\_2\_ce1 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_exitcond\_flatten\_fu\_253\_p2() {

exitcond\_flatten\_fu\_253\_p2 = (!indvar\_flatten\_phi\_fu\_209\_p4.read().is\_01() || !ap\_const\_lv4\_9.is\_01())? sc\_lv<1>(): sc\_lv<1>(indvar\_flatten\_phi\_fu\_209\_p4.read() == ap\_const\_lv4\_9);

}

void matrixmul::thread\_exitcond\_fu\_271\_p2() {

exitcond\_fu\_271\_p2 = (!j\_phi\_fu\_231\_p4.read().is\_01() || !ap\_const\_lv2\_3.is\_01())? sc\_lv<1>(): sc\_lv<1>(j\_phi\_fu\_231\_p4.read() == ap\_const\_lv2\_3);

}

void matrixmul::thread\_grp\_fu\_376\_p2() {

grp\_fu\_376\_p2 = (!tmp\_7\_2\_fu\_329\_p0.read().is\_01() || !tmp\_7\_2\_fu\_329\_p1.read().is\_01())? sc\_lv<16>(): sc\_bigint<8>(tmp\_7\_2\_fu\_329\_p0.read()) \* sc\_bigint<8>(tmp\_7\_2\_fu\_329\_p1.read());

}

void matrixmul::thread\_i\_1\_fu\_265\_p2() {

i\_1\_fu\_265\_p2 = (!i\_phi\_fu\_220\_p4.read().is\_01() || !ap\_const\_lv2\_1.is\_01())? sc\_lv<2>(): (sc\_biguint<2>(i\_phi\_fu\_220\_p4.read()) + sc\_biguint<2>(ap\_const\_lv2\_1));

}

void matrixmul::thread\_i\_phi\_fu\_220\_p4() {

if ((esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00000000.read(), ap\_const\_boolean\_0))) {

i\_phi\_fu\_220\_p4 = tmp\_mid2\_v\_reg\_454.read();

} else {

i\_phi\_fu\_220\_p4 = i\_reg\_216.read();

}

}

void matrixmul::thread\_indvar\_flatten\_next\_fu\_259\_p2() {

indvar\_flatten\_next\_fu\_259\_p2 = (!indvar\_flatten\_phi\_fu\_209\_p4.read().is\_01() || !ap\_const\_lv4\_1.is\_01())? sc\_lv<4>(): (sc\_biguint<4>(indvar\_flatten\_phi\_fu\_209\_p4.read()) + sc\_biguint<4>(ap\_const\_lv4\_1));

}

void matrixmul::thread\_indvar\_flatten\_phi\_fu\_209\_p4() {

if ((esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00000000.read(), ap\_const\_boolean\_0))) {

indvar\_flatten\_phi\_fu\_209\_p4 = indvar\_flatten\_next\_reg\_441.read();

} else {

indvar\_flatten\_phi\_fu\_209\_p4 = indvar\_flatten\_reg\_205.read();

}

}

void matrixmul::thread\_j\_1\_fu\_311\_p2() {

j\_1\_fu\_311\_p2 = (!j\_mid2\_reg\_446.read().is\_01() || !ap\_const\_lv2\_1.is\_01())? sc\_lv<2>(): (sc\_biguint<2>(j\_mid2\_reg\_446.read()) + sc\_biguint<2>(ap\_const\_lv2\_1));

}

void matrixmul::thread\_j\_mid2\_fu\_277\_p3() {

j\_mid2\_fu\_277\_p3 = (!exitcond\_fu\_271\_p2.read()[0].is\_01())? sc\_lv<2>(): ((exitcond\_fu\_271\_p2.read()[0].to\_bool())? ap\_const\_lv2\_0: j\_phi\_fu\_231\_p4.read());

}

void matrixmul::thread\_j\_phi\_fu\_231\_p4() {

if ((esl\_seteq<1,1,1>(exitcond\_flatten\_reg\_437.read(), ap\_const\_lv1\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage0.read()) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00000000.read(), ap\_const\_boolean\_0))) {

j\_phi\_fu\_231\_p4 = j\_1\_reg\_496.read();

} else {

j\_phi\_fu\_231\_p4 = j\_reg\_227.read();

}

}

void matrixmul::thread\_p\_shl\_cast\_fu\_345\_p1() {

p\_shl\_cast\_fu\_345\_p1 = esl\_zext<5,4>(tmp\_fu\_338\_p3.read());

}

void matrixmul::thread\_res\_address0() {

res\_address0 = (sc\_lv<4>) (tmp\_10\_cast\_fu\_364\_p1.read());

}

void matrixmul::thread\_res\_ce0() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()))) {

res\_ce0 = ap\_const\_logic\_1;

} else {

res\_ce0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_res\_d0() {

res\_d0 = grp\_fu\_384\_p3.read();

}

void matrixmul::thread\_res\_we0() {

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) &&

esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011001.read(), ap\_const\_boolean\_0) &&

esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) &&

esl\_seteq<1,1,1>(ap\_const\_lv1\_0, ap\_reg\_pp0\_iter1\_exitcond\_flatten\_reg\_437.read()))) {

res\_we0 = ap\_const\_logic\_1;

} else {

res\_we0 = ap\_const\_logic\_0;

}

}

void matrixmul::thread\_tmp\_10\_cast\_fu\_364\_p1() {

tmp\_10\_cast\_fu\_364\_p1 = esl\_sext<64,5>(tmp\_s\_fu\_358\_p2.read());

}

void matrixmul::thread\_tmp\_1\_fu\_349\_p2() {

tmp\_1\_fu\_349\_p2 = (!p\_shl\_cast\_fu\_345\_p1.read().is\_01() || !tmp\_mid2\_cast\_fu\_335\_p1.read().is\_01())? sc\_lv<5>(): (sc\_biguint<5>(p\_shl\_cast\_fu\_345\_p1.read()) - sc\_biguint<5>(tmp\_mid2\_cast\_fu\_335\_p1.read()));

}

void matrixmul::thread\_tmp\_2\_cast\_fu\_355\_p1() {

tmp\_2\_cast\_fu\_355\_p1 = esl\_zext<5,2>(ap\_reg\_pp0\_iter1\_j\_mid2\_reg\_446.read());

}

void matrixmul::thread\_tmp\_7\_2\_fu\_329\_p0() {

tmp\_7\_2\_fu\_329\_p0 = tmp\_9\_reg\_491.read();

}

void matrixmul::thread\_tmp\_7\_2\_fu\_329\_p1() {

tmp\_7\_2\_fu\_329\_p1 = a\_2\_load\_reg\_486.read();

}

void matrixmul::thread\_tmp\_fu\_338\_p3() {

tmp\_fu\_338\_p3 = esl\_concat<2,2>(ap\_reg\_pp0\_iter1\_tmp\_mid2\_v\_reg\_454.read(), ap\_const\_lv2\_0);

}

void matrixmul::thread\_tmp\_mid2\_cast\_fu\_335\_p1() {

tmp\_mid2\_cast\_fu\_335\_p1 = esl\_zext<5,2>(ap\_reg\_pp0\_iter1\_tmp\_mid2\_v\_reg\_454.read());

}

void matrixmul::thread\_tmp\_mid2\_fu\_293\_p1() {

tmp\_mid2\_fu\_293\_p1 = esl\_zext<64,2>(tmp\_mid2\_v\_fu\_285\_p3.read());

}

void matrixmul::thread\_tmp\_mid2\_v\_fu\_285\_p3() {

tmp\_mid2\_v\_fu\_285\_p3 = (!exitcond\_fu\_271\_p2.read()[0].is\_01())? sc\_lv<2>(): ((exitcond\_fu\_271\_p2.read()[0].to\_bool())? i\_1\_fu\_265\_p2.read(): i\_phi\_fu\_220\_p4.read());

}

void matrixmul::thread\_tmp\_s\_fu\_358\_p2() {

tmp\_s\_fu\_358\_p2 = (!tmp\_1\_fu\_349\_p2.read().is\_01() || !tmp\_2\_cast\_fu\_355\_p1.read().is\_01())? sc\_lv<5>(): (sc\_biguint<5>(tmp\_1\_fu\_349\_p2.read()) + sc\_biguint<5>(tmp\_2\_cast\_fu\_355\_p1.read()));

}

void matrixmul::thread\_ap\_NS\_fsm() {

switch (ap\_CS\_fsm.read().to\_uint64()) {

case 1 :

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state1.read()) && esl\_seteq<1,1,1>(ap\_start.read(), ap\_const\_logic\_1))) {

ap\_NS\_fsm = ap\_ST\_fsm\_pp0\_stage0;

} else {

ap\_NS\_fsm = ap\_ST\_fsm\_state1;

}

break;

case 2 :

if ((esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011011.read(), ap\_const\_boolean\_0) && !(esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011011.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_253\_p2.read(), ap\_const\_lv1\_1) && esl\_seteq<1,1,1>(ap\_enable\_reg\_pp0\_iter1.read(), ap\_const\_logic\_0)))) {

ap\_NS\_fsm = ap\_ST\_fsm\_pp0\_stage1;

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage0\_flag00011011.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_253\_p2.read(), ap\_const\_lv1\_1) && esl\_seteq<1,1,1>(ap\_enable\_reg\_pp0\_iter1.read(), ap\_const\_logic\_0))) {

ap\_NS\_fsm = ap\_ST\_fsm\_state6;

} else {

ap\_NS\_fsm = ap\_ST\_fsm\_pp0\_stage0;

}

break;

case 4 :

if ((esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011011.read(), ap\_const\_boolean\_0) && !(esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) && esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011011.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(ap\_enable\_reg\_pp0\_iter0.read(), ap\_const\_logic\_0)))) {

ap\_NS\_fsm = ap\_ST\_fsm\_pp0\_stage0;

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_pp0\_stage1.read()) && esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read()) && esl\_seteq<1,1,1>(ap\_block\_pp0\_stage1\_flag00011011.read(), ap\_const\_boolean\_0) && esl\_seteq<1,1,1>(ap\_enable\_reg\_pp0\_iter0.read(), ap\_const\_logic\_0))) {

ap\_NS\_fsm = ap\_ST\_fsm\_state6;

} else {

ap\_NS\_fsm = ap\_ST\_fsm\_pp0\_stage1;

}

break;

case 8 :

ap\_NS\_fsm = ap\_ST\_fsm\_state1;

break;

default :

ap\_NS\_fsm = (sc\_lv<4>) ("XXXX");

break;

}

}

void matrixmul::thread\_hdltv\_gen() {

const char\* dump\_tv = std::getenv("AP\_WRITE\_TV");

if (!(dump\_tv && string(dump\_tv) == "on")) return;

wait();

mHdltvinHandle << "[ " << endl;

mHdltvoutHandle << "[ " << endl;

int ap\_cycleNo = 0;

while (1) {

wait();

const char\* mComma = ap\_cycleNo == 0 ? " " : ", " ;

mHdltvinHandle << mComma << "{" << " \"ap\_rst\" : \"" << ap\_rst.read() << "\" ";

mHdltvinHandle << " , " << " \"ap\_start\" : \"" << ap\_start.read() << "\" ";

mHdltvoutHandle << mComma << "{" << " \"ap\_done\" : \"" << ap\_done.read() << "\" ";

mHdltvoutHandle << " , " << " \"ap\_idle\" : \"" << ap\_idle.read() << "\" ";

mHdltvoutHandle << " , " << " \"ap\_ready\" : \"" << ap\_ready.read() << "\" ";

mHdltvoutHandle << " , " << " \"a\_0\_address0\" : \"" << a\_0\_address0.read() << "\" ";

mHdltvoutHandle << " , " << " \"a\_0\_ce0\" : \"" << a\_0\_ce0.read() << "\" ";

mHdltvinHandle << " , " << " \"a\_0\_q0\" : \"" << a\_0\_q0.read() << "\" ";

mHdltvoutHandle << " , " << " \"a\_1\_address0\" : \"" << a\_1\_address0.read() << "\" ";

mHdltvoutHandle << " , " << " \"a\_1\_ce0\" : \"" << a\_1\_ce0.read() << "\" ";

mHdltvinHandle << " , " << " \"a\_1\_q0\" : \"" << a\_1\_q0.read() << "\" ";

mHdltvoutHandle << " , " << " \"a\_2\_address0\" : \"" << a\_2\_address0.read() << "\" ";

mHdltvoutHandle << " , " << " \"a\_2\_ce0\" : \"" << a\_2\_ce0.read() << "\" ";

mHdltvinHandle << " , " << " \"a\_2\_q0\" : \"" << a\_2\_q0.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_0\_address0\" : \"" << b\_0\_address0.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_0\_ce0\" : \"" << b\_0\_ce0.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_0\_q0\" : \"" << b\_0\_q0.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_0\_address1\" : \"" << b\_0\_address1.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_0\_ce1\" : \"" << b\_0\_ce1.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_0\_q1\" : \"" << b\_0\_q1.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_1\_address0\" : \"" << b\_1\_address0.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_1\_ce0\" : \"" << b\_1\_ce0.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_1\_q0\" : \"" << b\_1\_q0.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_1\_address1\" : \"" << b\_1\_address1.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_1\_ce1\" : \"" << b\_1\_ce1.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_1\_q1\" : \"" << b\_1\_q1.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_2\_address0\" : \"" << b\_2\_address0.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_2\_ce0\" : \"" << b\_2\_ce0.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_2\_q0\" : \"" << b\_2\_q0.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_2\_address1\" : \"" << b\_2\_address1.read() << "\" ";

mHdltvoutHandle << " , " << " \"b\_2\_ce1\" : \"" << b\_2\_ce1.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_2\_q1\" : \"" << b\_2\_q1.read() << "\" ";

mHdltvoutHandle << " , " << " \"res\_address0\" : \"" << res\_address0.read() << "\" ";

mHdltvoutHandle << " , " << " \"res\_ce0\" : \"" << res\_ce0.read() << "\" ";

mHdltvoutHandle << " , " << " \"res\_we0\" : \"" << res\_we0.read() << "\" ";

mHdltvoutHandle << " , " << " \"res\_d0\" : \"" << res\_d0.read() << "\" ";

mHdltvinHandle << "}" << std::endl;

mHdltvoutHandle << "}" << std::endl;

ap\_cycleNo++;

}

}

}