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

// 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<5> matrixmul::ap\_ST\_fsm\_state1 = "1";

const sc\_lv<5> matrixmul::ap\_ST\_fsm\_state2 = "10";

const sc\_lv<5> matrixmul::ap\_ST\_fsm\_pp0\_stage0 = "100";

const sc\_lv<5> matrixmul::ap\_ST\_fsm\_pp0\_stage1 = "1000";

const sc\_lv<5> matrixmul::ap\_ST\_fsm\_state6 = "10000";

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\_1 = "1";

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

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

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

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

const sc\_lv<32> matrixmul::ap\_const\_lv32\_4 = "100";

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<16> matrixmul::ap\_const\_lv16\_0 = "0000000000000000";

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";

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

matrixmul\_mac\_mulbkb\_U1 = new matrixmul\_mac\_mulbkb<1,1,8,8,16,16>("matrixmul\_mac\_mulbkb\_U1");

matrixmul\_mac\_mulbkb\_U1->din0(b\_load\_reg\_346);

matrixmul\_mac\_mulbkb\_U1->din1(a\_load\_reg\_341);

matrixmul\_mac\_mulbkb\_U1->din2(res\_q0);

matrixmul\_mac\_mulbkb\_U1->dout(grp\_fu\_279\_p3);

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

dont\_initialize();

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

SC\_METHOD(thread\_a\_address0);

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

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

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

sensitive << ( tmp\_11\_cast\_fu\_235\_p1 );

SC\_METHOD(thread\_a\_ce0);

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

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

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

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\_state2);

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\_state3\_pp0\_stage0\_iter0);

SC\_METHOD(thread\_ap\_block\_state4\_pp0\_stage1\_iter0);

SC\_METHOD(thread\_ap\_block\_state5\_pp0\_stage0\_iter1);

SC\_METHOD(thread\_ap\_condition\_pp0\_exit\_iter0\_state3);

sensitive << ( exitcond\_fu\_214\_p2 );

SC\_METHOD(thread\_ap\_done);

sensitive << ( exitcond\_flatten\_fu\_137\_p2 );

sensitive << ( ap\_CS\_fsm\_state2 );

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 << ( exitcond\_flatten\_fu\_137\_p2 );

sensitive << ( ap\_CS\_fsm\_state2 );

SC\_METHOD(thread\_b\_address0);

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

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

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

sensitive << ( tmp\_14\_cast\_fu\_263\_p1 );

SC\_METHOD(thread\_b\_ce0);

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

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

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

SC\_METHOD(thread\_exitcond1\_fu\_155\_p2);

sensitive << ( exitcond\_flatten\_fu\_137\_p2 );

sensitive << ( ap\_CS\_fsm\_state2 );

sensitive << ( j\_reg\_115 );

SC\_METHOD(thread\_exitcond\_flatten\_fu\_137\_p2);

sensitive << ( ap\_CS\_fsm\_state2 );

sensitive << ( indvar\_flatten\_reg\_93 );

SC\_METHOD(thread\_exitcond\_fu\_214\_p2);

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

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

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

sensitive << ( k\_phi\_fu\_130\_p4 );

SC\_METHOD(thread\_i\_1\_fu\_149\_p2);

sensitive << ( i\_reg\_104 );

SC\_METHOD(thread\_indvar\_flatten\_next\_fu\_143\_p2);

sensitive << ( indvar\_flatten\_reg\_93 );

SC\_METHOD(thread\_j\_1\_fu\_274\_p2);

sensitive << ( j\_mid2\_reg\_297 );

SC\_METHOD(thread\_j\_mid2\_fu\_161\_p3);

sensitive << ( j\_reg\_115 );

sensitive << ( exitcond1\_fu\_155\_p2 );

SC\_METHOD(thread\_k\_1\_fu\_220\_p2);

sensitive << ( k\_phi\_fu\_130\_p4 );

SC\_METHOD(thread\_k\_phi\_fu\_130\_p4);

sensitive << ( k\_reg\_126 );

sensitive << ( exitcond\_reg\_322 );

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

sensitive << ( k\_1\_reg\_326 );

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

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

SC\_METHOD(thread\_p\_shl1\_cast\_fu\_248\_p1);

sensitive << ( tmp\_4\_fu\_240\_p3 );

SC\_METHOD(thread\_p\_shl\_cast\_fu\_189\_p1);

sensitive << ( tmp\_fu\_181\_p3 );

SC\_METHOD(thread\_res\_address0);

sensitive << ( ap\_CS\_fsm\_state2 );

sensitive << ( res\_addr\_reg\_317 );

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

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

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

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

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

sensitive << ( tmp\_10\_cast\_fu\_209\_p1 );

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

SC\_METHOD(thread\_res\_ce0);

sensitive << ( ap\_CS\_fsm\_state2 );

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

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

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

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\_state2 );

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

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

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

sensitive << ( grp\_fu\_279\_p3 );

SC\_METHOD(thread\_res\_we0);

sensitive << ( exitcond\_flatten\_fu\_137\_p2 );

sensitive << ( ap\_CS\_fsm\_state2 );

sensitive << ( exitcond\_reg\_322 );

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

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

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

SC\_METHOD(thread\_tmp\_10\_cast\_fu\_209\_p1);

sensitive << ( tmp\_s\_fu\_203\_p2 );

SC\_METHOD(thread\_tmp\_10\_fu\_252\_p2);

sensitive << ( tmp\_4\_cast\_fu\_226\_p1 );

sensitive << ( p\_shl1\_cast\_fu\_248\_p1 );

SC\_METHOD(thread\_tmp\_11\_cast\_fu\_235\_p1);

sensitive << ( tmp\_2\_fu\_230\_p2 );

SC\_METHOD(thread\_tmp\_11\_fu\_258\_p2);

sensitive << ( tmp\_2\_cast\_reg\_312 );

sensitive << ( tmp\_10\_fu\_252\_p2 );

SC\_METHOD(thread\_tmp\_14\_cast\_fu\_263\_p1);

sensitive << ( tmp\_11\_fu\_258\_p2 );

SC\_METHOD(thread\_tmp\_1\_fu\_193\_p2);

sensitive << ( p\_shl\_cast\_fu\_189\_p1 );

sensitive << ( tmp\_mid2\_cast\_fu\_177\_p1 );

SC\_METHOD(thread\_tmp\_2\_cast\_fu\_199\_p1);

sensitive << ( j\_mid2\_fu\_161\_p3 );

SC\_METHOD(thread\_tmp\_2\_fu\_230\_p2);

sensitive << ( tmp\_1\_reg\_307 );

sensitive << ( tmp\_4\_cast\_fu\_226\_p1 );

SC\_METHOD(thread\_tmp\_4\_cast\_fu\_226\_p1);

sensitive << ( k\_phi\_fu\_130\_p4 );

SC\_METHOD(thread\_tmp\_4\_fu\_240\_p3);

sensitive << ( k\_phi\_fu\_130\_p4 );

SC\_METHOD(thread\_tmp\_fu\_181\_p3);

sensitive << ( tmp\_mid2\_v\_fu\_169\_p3 );

SC\_METHOD(thread\_tmp\_mid2\_cast\_fu\_177\_p1);

sensitive << ( tmp\_mid2\_v\_fu\_169\_p3 );

SC\_METHOD(thread\_tmp\_mid2\_v\_fu\_169\_p3);

sensitive << ( i\_reg\_104 );

sensitive << ( exitcond1\_fu\_155\_p2 );

sensitive << ( i\_1\_fu\_149\_p2 );

SC\_METHOD(thread\_tmp\_s\_fu\_203\_p2);

sensitive << ( tmp\_1\_fu\_193\_p2 );

sensitive << ( tmp\_2\_cast\_fu\_199\_p1 );

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

sensitive << ( ap\_start );

sensitive << ( ap\_CS\_fsm );

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

sensitive << ( exitcond\_flatten\_fu\_137\_p2 );

sensitive << ( ap\_CS\_fsm\_state2 );

sensitive << ( exitcond\_fu\_214\_p2 );

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

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

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

SC\_THREAD(thread\_hdltv\_gen);

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

ap\_CS\_fsm = "00001";

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\_address0, "(port)a\_address0");

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

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

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

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

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

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");

sc\_trace(mVcdFile, res\_q0, "(port)res\_q0");

#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, k\_reg\_126, "k\_reg\_126");

sc\_trace(mVcdFile, exitcond\_flatten\_fu\_137\_p2, "exitcond\_flatten\_fu\_137\_p2");

sc\_trace(mVcdFile, ap\_CS\_fsm\_state2, "ap\_CS\_fsm\_state2");

sc\_trace(mVcdFile, indvar\_flatten\_next\_fu\_143\_p2, "indvar\_flatten\_next\_fu\_143\_p2");

sc\_trace(mVcdFile, indvar\_flatten\_next\_reg\_292, "indvar\_flatten\_next\_reg\_292");

sc\_trace(mVcdFile, j\_mid2\_fu\_161\_p3, "j\_mid2\_fu\_161\_p3");

sc\_trace(mVcdFile, j\_mid2\_reg\_297, "j\_mid2\_reg\_297");

sc\_trace(mVcdFile, tmp\_mid2\_v\_fu\_169\_p3, "tmp\_mid2\_v\_fu\_169\_p3");

sc\_trace(mVcdFile, tmp\_mid2\_v\_reg\_302, "tmp\_mid2\_v\_reg\_302");

sc\_trace(mVcdFile, tmp\_1\_fu\_193\_p2, "tmp\_1\_fu\_193\_p2");

sc\_trace(mVcdFile, tmp\_1\_reg\_307, "tmp\_1\_reg\_307");

sc\_trace(mVcdFile, tmp\_2\_cast\_fu\_199\_p1, "tmp\_2\_cast\_fu\_199\_p1");

sc\_trace(mVcdFile, tmp\_2\_cast\_reg\_312, "tmp\_2\_cast\_reg\_312");

sc\_trace(mVcdFile, res\_addr\_reg\_317, "res\_addr\_reg\_317");

sc\_trace(mVcdFile, exitcond\_fu\_214\_p2, "exitcond\_fu\_214\_p2");

sc\_trace(mVcdFile, exitcond\_reg\_322, "exitcond\_reg\_322");

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

sc\_trace(mVcdFile, ap\_block\_state3\_pp0\_stage0\_iter0, "ap\_block\_state3\_pp0\_stage0\_iter0");

sc\_trace(mVcdFile, ap\_block\_state5\_pp0\_stage0\_iter1, "ap\_block\_state5\_pp0\_stage0\_iter1");

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

sc\_trace(mVcdFile, k\_1\_fu\_220\_p2, "k\_1\_fu\_220\_p2");

sc\_trace(mVcdFile, k\_1\_reg\_326, "k\_1\_reg\_326");

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

sc\_trace(mVcdFile, a\_load\_reg\_341, "a\_load\_reg\_341");

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

sc\_trace(mVcdFile, ap\_block\_state4\_pp0\_stage1\_iter0, "ap\_block\_state4\_pp0\_stage1\_iter0");

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

sc\_trace(mVcdFile, b\_load\_reg\_346, "b\_load\_reg\_346");

sc\_trace(mVcdFile, j\_1\_fu\_274\_p2, "j\_1\_fu\_274\_p2");

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

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

sc\_trace(mVcdFile, ap\_condition\_pp0\_exit\_iter0\_state3, "ap\_condition\_pp0\_exit\_iter0\_state3");

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

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

sc\_trace(mVcdFile, indvar\_flatten\_reg\_93, "indvar\_flatten\_reg\_93");

sc\_trace(mVcdFile, i\_reg\_104, "i\_reg\_104");

sc\_trace(mVcdFile, j\_reg\_115, "j\_reg\_115");

sc\_trace(mVcdFile, k\_phi\_fu\_130\_p4, "k\_phi\_fu\_130\_p4");

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

sc\_trace(mVcdFile, tmp\_10\_cast\_fu\_209\_p1, "tmp\_10\_cast\_fu\_209\_p1");

sc\_trace(mVcdFile, tmp\_11\_cast\_fu\_235\_p1, "tmp\_11\_cast\_fu\_235\_p1");

sc\_trace(mVcdFile, tmp\_14\_cast\_fu\_263\_p1, "tmp\_14\_cast\_fu\_263\_p1");

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

sc\_trace(mVcdFile, grp\_fu\_279\_p3, "grp\_fu\_279\_p3");

sc\_trace(mVcdFile, exitcond1\_fu\_155\_p2, "exitcond1\_fu\_155\_p2");

sc\_trace(mVcdFile, i\_1\_fu\_149\_p2, "i\_1\_fu\_149\_p2");

sc\_trace(mVcdFile, tmp\_fu\_181\_p3, "tmp\_fu\_181\_p3");

sc\_trace(mVcdFile, p\_shl\_cast\_fu\_189\_p1, "p\_shl\_cast\_fu\_189\_p1");

sc\_trace(mVcdFile, tmp\_mid2\_cast\_fu\_177\_p1, "tmp\_mid2\_cast\_fu\_177\_p1");

sc\_trace(mVcdFile, tmp\_s\_fu\_203\_p2, "tmp\_s\_fu\_203\_p2");

sc\_trace(mVcdFile, tmp\_4\_cast\_fu\_226\_p1, "tmp\_4\_cast\_fu\_226\_p1");

sc\_trace(mVcdFile, tmp\_2\_fu\_230\_p2, "tmp\_2\_fu\_230\_p2");

sc\_trace(mVcdFile, tmp\_4\_fu\_240\_p3, "tmp\_4\_fu\_240\_p3");

sc\_trace(mVcdFile, p\_shl1\_cast\_fu\_248\_p1, "p\_shl1\_cast\_fu\_248\_p1");

sc\_trace(mVcdFile, tmp\_10\_fu\_252\_p2, "tmp\_10\_fu\_252\_p2");

sc\_trace(mVcdFile, tmp\_11\_fu\_258\_p2, "tmp\_11\_fu\_258\_p2");

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\_mac\_mulbkb\_U1;

}

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\_state3.read()))) {

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

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read()) &&

esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.read(), ap\_const\_lv1\_0))) {

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\_condition\_pp0\_exit\_iter0\_state3.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\_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\_block\_pp0\_stage1\_flag00011011.read(), ap\_const\_boolean\_0))))) {

ap\_enable\_reg\_pp0\_iter1 = (ap\_condition\_pp0\_exit\_iter0\_state3.read() ^ ap\_const\_logic\_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\_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\_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\_state2.read()) &&

esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.read(), ap\_const\_lv1\_0))) {

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

}

}

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

i\_reg\_104 = tmp\_mid2\_v\_reg\_302.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\_104 = ap\_const\_lv2\_0;

}

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

indvar\_flatten\_reg\_93 = indvar\_flatten\_next\_reg\_292.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\_93 = ap\_const\_lv4\_0;

}

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

j\_reg\_115 = j\_1\_fu\_274\_p2.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\_115 = ap\_const\_lv2\_0;

}

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\_reg\_322.read()) &&

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

k\_reg\_126 = k\_1\_reg\_326.read();

} else if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read()) &&

esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.read(), ap\_const\_lv1\_0))) {

k\_reg\_126 = 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>(ap\_const\_lv1\_0, exitcond\_reg\_322.read()))) {

a\_load\_reg\_341 = a\_q0.read();

b\_load\_reg\_346 = b\_q0.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))) {

exitcond\_reg\_322 = exitcond\_fu\_214\_p2.read();

}

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read())) {

indvar\_flatten\_next\_reg\_292 = indvar\_flatten\_next\_fu\_143\_p2.read();

}

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read()) && esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.read(), ap\_const\_lv1\_0))) {

j\_mid2\_reg\_297 = j\_mid2\_fu\_161\_p3.read();

res\_addr\_reg\_317 = (sc\_lv<4>) (tmp\_10\_cast\_fu\_209\_p1.read());

tmp\_1\_reg\_307 = tmp\_1\_fu\_193\_p2.read();

tmp\_2\_cast\_reg\_312 = tmp\_2\_cast\_fu\_199\_p1.read();

tmp\_mid2\_v\_reg\_302 = tmp\_mid2\_v\_fu\_169\_p3.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\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()))) {

k\_1\_reg\_326 = k\_1\_fu\_220\_p2.read();

}

}

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

a\_address0 = (sc\_lv<4>) (tmp\_11\_cast\_fu\_235\_p1.read());

}

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

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\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()))) {

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

} else {

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

}

}

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

ap\_CS\_fsm\_pp0\_stage0 = ap\_CS\_fsm.read()[2];

}

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

ap\_CS\_fsm\_pp0\_stage1 = ap\_CS\_fsm.read()[3];

}

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

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

}

void matrixmul::thread\_ap\_CS\_fsm\_state2() {

ap\_CS\_fsm\_state2 = ap\_CS\_fsm.read()[1];

}

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

ap\_CS\_fsm\_state6 = ap\_CS\_fsm.read()[4];

}

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\_state3\_pp0\_stage0\_iter0() {

ap\_block\_state3\_pp0\_stage0\_iter0 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_state4\_pp0\_stage1\_iter0() {

ap\_block\_state4\_pp0\_stage1\_iter0 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_block\_state5\_pp0\_stage0\_iter1() {

ap\_block\_state5\_pp0\_stage0\_iter1 = !esl\_seteq<1,1,1>(ap\_const\_boolean\_1, ap\_const\_boolean\_1);

}

void matrixmul::thread\_ap\_condition\_pp0\_exit\_iter0\_state3() {

if (esl\_seteq<1,1,1>(exitcond\_fu\_214\_p2.read(), ap\_const\_lv1\_1)) {

ap\_condition\_pp0\_exit\_iter0\_state3 = ap\_const\_logic\_1;

} else {

ap\_condition\_pp0\_exit\_iter0\_state3 = ap\_const\_logic\_0;

}

}

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

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read()) &&

esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.read(), ap\_const\_lv1\_1))) {

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\_state2.read()) &&

esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.read(), ap\_const\_lv1\_1))) {

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

} else {

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

}

}

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

b\_address0 = (sc\_lv<4>) (tmp\_14\_cast\_fu\_263\_p1.read());

}

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

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\_logic\_1, ap\_enable\_reg\_pp0\_iter0.read()))) {

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

} else {

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

}

}

void matrixmul::thread\_exitcond1\_fu\_155\_p2() {

exitcond1\_fu\_155\_p2 = (!j\_reg\_115.read().is\_01() || !ap\_const\_lv2\_3.is\_01())? sc\_lv<1>(): sc\_lv<1>(j\_reg\_115.read() == ap\_const\_lv2\_3);

}

void matrixmul::thread\_exitcond\_flatten\_fu\_137\_p2() {

exitcond\_flatten\_fu\_137\_p2 = (!indvar\_flatten\_reg\_93.read().is\_01() || !ap\_const\_lv4\_9.is\_01())? sc\_lv<1>(): sc\_lv<1>(indvar\_flatten\_reg\_93.read() == ap\_const\_lv4\_9);

}

void matrixmul::thread\_exitcond\_fu\_214\_p2() {

exitcond\_fu\_214\_p2 = (!k\_phi\_fu\_130\_p4.read().is\_01() || !ap\_const\_lv2\_3.is\_01())? sc\_lv<1>(): sc\_lv<1>(k\_phi\_fu\_130\_p4.read() == ap\_const\_lv2\_3);

}

void matrixmul::thread\_i\_1\_fu\_149\_p2() {

i\_1\_fu\_149\_p2 = (!i\_reg\_104.read().is\_01() || !ap\_const\_lv2\_1.is\_01())? sc\_lv<2>(): (sc\_biguint<2>(i\_reg\_104.read()) + sc\_biguint<2>(ap\_const\_lv2\_1));

}

void matrixmul::thread\_indvar\_flatten\_next\_fu\_143\_p2() {

indvar\_flatten\_next\_fu\_143\_p2 = (!indvar\_flatten\_reg\_93.read().is\_01() || !ap\_const\_lv4\_1.is\_01())? sc\_lv<4>(): (sc\_biguint<4>(indvar\_flatten\_reg\_93.read()) + sc\_biguint<4>(ap\_const\_lv4\_1));

}

void matrixmul::thread\_j\_1\_fu\_274\_p2() {

j\_1\_fu\_274\_p2 = (!j\_mid2\_reg\_297.read().is\_01() || !ap\_const\_lv2\_1.is\_01())? sc\_lv<2>(): (sc\_biguint<2>(j\_mid2\_reg\_297.read()) + sc\_biguint<2>(ap\_const\_lv2\_1));

}

void matrixmul::thread\_j\_mid2\_fu\_161\_p3() {

j\_mid2\_fu\_161\_p3 = (!exitcond1\_fu\_155\_p2.read()[0].is\_01())? sc\_lv<2>(): ((exitcond1\_fu\_155\_p2.read()[0].to\_bool())? ap\_const\_lv2\_0: j\_reg\_115.read());

}

void matrixmul::thread\_k\_1\_fu\_220\_p2() {

k\_1\_fu\_220\_p2 = (!k\_phi\_fu\_130\_p4.read().is\_01() || !ap\_const\_lv2\_1.is\_01())? sc\_lv<2>(): (sc\_biguint<2>(k\_phi\_fu\_130\_p4.read()) + sc\_biguint<2>(ap\_const\_lv2\_1));

}

void matrixmul::thread\_k\_phi\_fu\_130\_p4() {

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

esl\_seteq<1,1,1>(ap\_const\_lv1\_0, exitcond\_reg\_322.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))) {

k\_phi\_fu\_130\_p4 = k\_1\_reg\_326.read();

} else {

k\_phi\_fu\_130\_p4 = k\_reg\_126.read();

}

}

void matrixmul::thread\_p\_shl1\_cast\_fu\_248\_p1() {

p\_shl1\_cast\_fu\_248\_p1 = esl\_zext<5,4>(tmp\_4\_fu\_240\_p3.read());

}

void matrixmul::thread\_p\_shl\_cast\_fu\_189\_p1() {

p\_shl\_cast\_fu\_189\_p1 = esl\_zext<5,4>(tmp\_fu\_181\_p3.read());

}

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

if (((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)) ||

(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\_stage1.read()) &&

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

res\_address0 = res\_addr\_reg\_317.read();

} else if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read())) {

res\_address0 = (sc\_lv<4>) (tmp\_10\_cast\_fu\_209\_p1.read());

} else {

res\_address0 = "XXXX";

}

}

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

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.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\_logic\_1, ap\_enable\_reg\_pp0\_iter1.read())) ||

(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\_stage1.read()) &&

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

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

} else {

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

}

}

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

if ((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))) {

res\_d0 = grp\_fu\_279\_p3.read();

} else if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read())) {

res\_d0 = ap\_const\_lv16\_0;

} else {

res\_d0 = (sc\_lv<16>) ("XXXXXXXXXXXXXXXX");

}

}

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

if (((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read()) &&

esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.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\_block\_pp0\_stage0\_flag00011001.read(), ap\_const\_boolean\_0) &&

esl\_seteq<1,1,1>(ap\_const\_lv1\_0, exitcond\_reg\_322.read()) &&

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

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

} else {

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

}

}

void matrixmul::thread\_tmp\_10\_cast\_fu\_209\_p1() {

tmp\_10\_cast\_fu\_209\_p1 = esl\_sext<64,5>(tmp\_s\_fu\_203\_p2.read());

}

void matrixmul::thread\_tmp\_10\_fu\_252\_p2() {

tmp\_10\_fu\_252\_p2 = (!p\_shl1\_cast\_fu\_248\_p1.read().is\_01() || !tmp\_4\_cast\_fu\_226\_p1.read().is\_01())? sc\_lv<5>(): (sc\_biguint<5>(p\_shl1\_cast\_fu\_248\_p1.read()) - sc\_biguint<5>(tmp\_4\_cast\_fu\_226\_p1.read()));

}

void matrixmul::thread\_tmp\_11\_cast\_fu\_235\_p1() {

tmp\_11\_cast\_fu\_235\_p1 = esl\_sext<64,5>(tmp\_2\_fu\_230\_p2.read());

}

void matrixmul::thread\_tmp\_11\_fu\_258\_p2() {

tmp\_11\_fu\_258\_p2 = (!tmp\_10\_fu\_252\_p2.read().is\_01() || !tmp\_2\_cast\_reg\_312.read().is\_01())? sc\_lv<5>(): (sc\_biguint<5>(tmp\_10\_fu\_252\_p2.read()) + sc\_biguint<5>(tmp\_2\_cast\_reg\_312.read()));

}

void matrixmul::thread\_tmp\_14\_cast\_fu\_263\_p1() {

tmp\_14\_cast\_fu\_263\_p1 = esl\_sext<64,5>(tmp\_11\_fu\_258\_p2.read());

}

void matrixmul::thread\_tmp\_1\_fu\_193\_p2() {

tmp\_1\_fu\_193\_p2 = (!p\_shl\_cast\_fu\_189\_p1.read().is\_01() || !tmp\_mid2\_cast\_fu\_177\_p1.read().is\_01())? sc\_lv<5>(): (sc\_biguint<5>(p\_shl\_cast\_fu\_189\_p1.read()) - sc\_biguint<5>(tmp\_mid2\_cast\_fu\_177\_p1.read()));

}

void matrixmul::thread\_tmp\_2\_cast\_fu\_199\_p1() {

tmp\_2\_cast\_fu\_199\_p1 = esl\_zext<5,2>(j\_mid2\_fu\_161\_p3.read());

}

void matrixmul::thread\_tmp\_2\_fu\_230\_p2() {

tmp\_2\_fu\_230\_p2 = (!tmp\_1\_reg\_307.read().is\_01() || !tmp\_4\_cast\_fu\_226\_p1.read().is\_01())? sc\_lv<5>(): (sc\_biguint<5>(tmp\_1\_reg\_307.read()) + sc\_biguint<5>(tmp\_4\_cast\_fu\_226\_p1.read()));

}

void matrixmul::thread\_tmp\_4\_cast\_fu\_226\_p1() {

tmp\_4\_cast\_fu\_226\_p1 = esl\_zext<5,2>(k\_phi\_fu\_130\_p4.read());

}

void matrixmul::thread\_tmp\_4\_fu\_240\_p3() {

tmp\_4\_fu\_240\_p3 = esl\_concat<2,2>(k\_phi\_fu\_130\_p4.read(), ap\_const\_lv2\_0);

}

void matrixmul::thread\_tmp\_fu\_181\_p3() {

tmp\_fu\_181\_p3 = esl\_concat<2,2>(tmp\_mid2\_v\_fu\_169\_p3.read(), ap\_const\_lv2\_0);

}

void matrixmul::thread\_tmp\_mid2\_cast\_fu\_177\_p1() {

tmp\_mid2\_cast\_fu\_177\_p1 = esl\_zext<5,2>(tmp\_mid2\_v\_fu\_169\_p3.read());

}

void matrixmul::thread\_tmp\_mid2\_v\_fu\_169\_p3() {

tmp\_mid2\_v\_fu\_169\_p3 = (!exitcond1\_fu\_155\_p2.read()[0].is\_01())? sc\_lv<2>(): ((exitcond1\_fu\_155\_p2.read()[0].to\_bool())? i\_1\_fu\_149\_p2.read(): i\_reg\_104.read());

}

void matrixmul::thread\_tmp\_s\_fu\_203\_p2() {

tmp\_s\_fu\_203\_p2 = (!tmp\_1\_fu\_193\_p2.read().is\_01() || !tmp\_2\_cast\_fu\_199\_p1.read().is\_01())? sc\_lv<5>(): (sc\_biguint<5>(tmp\_1\_fu\_193\_p2.read()) + sc\_biguint<5>(tmp\_2\_cast\_fu\_199\_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\_state2;

} else {

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

}

break;

case 2 :

if ((esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state2.read()) && esl\_seteq<1,1,1>(exitcond\_flatten\_fu\_137\_p2.read(), ap\_const\_lv1\_1))) {

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

} else {

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

}

break;

case 4 :

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\_fu\_214\_p2.read(), ap\_const\_lv1\_1)))) {

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\_fu\_214\_p2.read(), ap\_const\_lv1\_1))) {

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

} else {

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

}

break;

case 8 :

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

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

} else {

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

}

break;

case 16 :

ap\_NS\_fsm = ap\_ST\_fsm\_state2;

break;

default :

ap\_NS\_fsm = (sc\_lv<5>) ("XXXXX");

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\_address0\" : \"" << a\_address0.read() << "\" ";

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

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

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

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

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

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

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

ap\_cycleNo++;

}

}

}