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

// 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.

//

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

#ifndef \_matrixmul\_HH\_

#define \_matrixmul\_HH\_

#include "systemc.h"

#include "AESL\_pkg.h"

#include "matrixmul\_mac\_mulbkb.h"

namespace ap\_rtl {

struct matrixmul : public sc\_module {

// Port declarations 17

sc\_in\_clk ap\_clk;

sc\_in< sc\_logic > ap\_rst;

sc\_in< sc\_logic > ap\_start;

sc\_out< sc\_logic > ap\_done;

sc\_out< sc\_logic > ap\_idle;

sc\_out< sc\_logic > ap\_ready;

sc\_out< sc\_lv<4> > a\_address0;

sc\_out< sc\_logic > a\_ce0;

sc\_in< sc\_lv<8> > a\_q0;

sc\_out< sc\_lv<4> > b\_address0;

sc\_out< sc\_logic > b\_ce0;

sc\_in< sc\_lv<8> > b\_q0;

sc\_out< sc\_lv<4> > res\_address0;

sc\_out< sc\_logic > res\_ce0;

sc\_out< sc\_logic > res\_we0;

sc\_out< sc\_lv<16> > res\_d0;

sc\_in< sc\_lv<16> > res\_q0;

// Module declarations

matrixmul(sc\_module\_name name);

SC\_HAS\_PROCESS(matrixmul);

~matrixmul();

sc\_trace\_file\* mVcdFile;

ofstream mHdltvinHandle;

ofstream mHdltvoutHandle;

matrixmul\_mac\_mulbkb<1,1,8,8,16,16>\* matrixmul\_mac\_mulbkb\_U1;

sc\_signal< sc\_lv<5> > ap\_CS\_fsm;

sc\_signal< sc\_logic > ap\_CS\_fsm\_state1;

sc\_signal< sc\_lv<2> > k\_reg\_126;

sc\_signal< sc\_lv<1> > exitcond\_flatten\_fu\_137\_p2;

sc\_signal< sc\_logic > ap\_CS\_fsm\_state2;

sc\_signal< sc\_lv<4> > indvar\_flatten\_next\_fu\_143\_p2;

sc\_signal< sc\_lv<4> > indvar\_flatten\_next\_reg\_292;

sc\_signal< sc\_lv<2> > j\_mid2\_fu\_161\_p3;

sc\_signal< sc\_lv<2> > j\_mid2\_reg\_297;

sc\_signal< sc\_lv<2> > tmp\_mid2\_v\_fu\_169\_p3;

sc\_signal< sc\_lv<2> > tmp\_mid2\_v\_reg\_302;

sc\_signal< sc\_lv<5> > tmp\_1\_fu\_193\_p2;

sc\_signal< sc\_lv<5> > tmp\_1\_reg\_307;

sc\_signal< sc\_lv<5> > tmp\_2\_cast\_fu\_199\_p1;

sc\_signal< sc\_lv<5> > tmp\_2\_cast\_reg\_312;

sc\_signal< sc\_lv<4> > res\_addr\_reg\_317;

sc\_signal< sc\_lv<1> > exitcond\_fu\_214\_p2;

sc\_signal< sc\_lv<1> > exitcond\_reg\_322;

sc\_signal< sc\_logic > ap\_CS\_fsm\_pp0\_stage0;

sc\_signal< bool > ap\_block\_state3\_pp0\_stage0\_iter0;

sc\_signal< bool > ap\_block\_state5\_pp0\_stage0\_iter1;

sc\_signal< bool > ap\_block\_pp0\_stage0\_flag00011001;

sc\_signal< sc\_lv<2> > k\_1\_fu\_220\_p2;

sc\_signal< sc\_lv<2> > k\_1\_reg\_326;

sc\_signal< sc\_logic > ap\_enable\_reg\_pp0\_iter0;

sc\_signal< sc\_lv<8> > a\_load\_reg\_341;

sc\_signal< sc\_logic > ap\_CS\_fsm\_pp0\_stage1;

sc\_signal< bool > ap\_block\_state4\_pp0\_stage1\_iter0;

sc\_signal< bool > ap\_block\_pp0\_stage1\_flag00011001;

sc\_signal< sc\_lv<8> > b\_load\_reg\_346;

sc\_signal< sc\_lv<2> > j\_1\_fu\_274\_p2;

sc\_signal< sc\_logic > ap\_CS\_fsm\_state6;

sc\_signal< bool > ap\_block\_pp0\_stage0\_flag00011011;

sc\_signal< sc\_logic > ap\_condition\_pp0\_exit\_iter0\_state3;

sc\_signal< sc\_logic > ap\_enable\_reg\_pp0\_iter1;

sc\_signal< bool > ap\_block\_pp0\_stage1\_flag00011011;

sc\_signal< sc\_lv<4> > indvar\_flatten\_reg\_93;

sc\_signal< sc\_lv<2> > i\_reg\_104;

sc\_signal< sc\_lv<2> > j\_reg\_115;

sc\_signal< sc\_lv<2> > k\_phi\_fu\_130\_p4;

sc\_signal< bool > ap\_block\_pp0\_stage0\_flag00000000;

sc\_signal< sc\_lv<64> > tmp\_10\_cast\_fu\_209\_p1;

sc\_signal< sc\_lv<64> > tmp\_11\_cast\_fu\_235\_p1;

sc\_signal< sc\_lv<64> > tmp\_14\_cast\_fu\_263\_p1;

sc\_signal< bool > ap\_block\_pp0\_stage1\_flag00000000;

sc\_signal< sc\_lv<16> > grp\_fu\_279\_p3;

sc\_signal< sc\_lv<1> > exitcond1\_fu\_155\_p2;

sc\_signal< sc\_lv<2> > i\_1\_fu\_149\_p2;

sc\_signal< sc\_lv<4> > tmp\_fu\_181\_p3;

sc\_signal< sc\_lv<5> > p\_shl\_cast\_fu\_189\_p1;

sc\_signal< sc\_lv<5> > tmp\_mid2\_cast\_fu\_177\_p1;

sc\_signal< sc\_lv<5> > tmp\_s\_fu\_203\_p2;

sc\_signal< sc\_lv<5> > tmp\_4\_cast\_fu\_226\_p1;

sc\_signal< sc\_lv<5> > tmp\_2\_fu\_230\_p2;

sc\_signal< sc\_lv<4> > tmp\_4\_fu\_240\_p3;

sc\_signal< sc\_lv<5> > p\_shl1\_cast\_fu\_248\_p1;

sc\_signal< sc\_lv<5> > tmp\_10\_fu\_252\_p2;

sc\_signal< sc\_lv<5> > tmp\_11\_fu\_258\_p2;

sc\_signal< sc\_lv<5> > ap\_NS\_fsm;

sc\_signal< sc\_logic > ap\_idle\_pp0;

sc\_signal< sc\_logic > ap\_enable\_pp0;

static const sc\_logic ap\_const\_logic\_1;

static const sc\_logic ap\_const\_logic\_0;

static const sc\_lv<5> ap\_ST\_fsm\_state1;

static const sc\_lv<5> ap\_ST\_fsm\_state2;

static const sc\_lv<5> ap\_ST\_fsm\_pp0\_stage0;

static const sc\_lv<5> ap\_ST\_fsm\_pp0\_stage1;

static const sc\_lv<5> ap\_ST\_fsm\_state6;

static const sc\_lv<32> ap\_const\_lv32\_0;

static const bool ap\_const\_boolean\_1;

static const sc\_lv<32> ap\_const\_lv32\_1;

static const sc\_lv<1> ap\_const\_lv1\_0;

static const sc\_lv<32> ap\_const\_lv32\_2;

static const bool ap\_const\_boolean\_0;

static const sc\_lv<32> ap\_const\_lv32\_3;

static const sc\_lv<32> ap\_const\_lv32\_4;

static const sc\_lv<1> ap\_const\_lv1\_1;

static const sc\_lv<4> ap\_const\_lv4\_0;

static const sc\_lv<2> ap\_const\_lv2\_0;

static const sc\_lv<16> ap\_const\_lv16\_0;

static const sc\_lv<4> ap\_const\_lv4\_9;

static const sc\_lv<4> ap\_const\_lv4\_1;

static const sc\_lv<2> ap\_const\_lv2\_1;

static const sc\_lv<2> ap\_const\_lv2\_3;

// Thread declarations

void thread\_ap\_clk\_no\_reset\_();

void thread\_a\_address0();

void thread\_a\_ce0();

void thread\_ap\_CS\_fsm\_pp0\_stage0();

void thread\_ap\_CS\_fsm\_pp0\_stage1();

void thread\_ap\_CS\_fsm\_state1();

void thread\_ap\_CS\_fsm\_state2();

void thread\_ap\_CS\_fsm\_state6();

void thread\_ap\_block\_pp0\_stage0\_flag00000000();

void thread\_ap\_block\_pp0\_stage0\_flag00011001();

void thread\_ap\_block\_pp0\_stage0\_flag00011011();

void thread\_ap\_block\_pp0\_stage1\_flag00000000();

void thread\_ap\_block\_pp0\_stage1\_flag00011001();

void thread\_ap\_block\_pp0\_stage1\_flag00011011();

void thread\_ap\_block\_state3\_pp0\_stage0\_iter0();

void thread\_ap\_block\_state4\_pp0\_stage1\_iter0();

void thread\_ap\_block\_state5\_pp0\_stage0\_iter1();

void thread\_ap\_condition\_pp0\_exit\_iter0\_state3();

void thread\_ap\_done();

void thread\_ap\_enable\_pp0();

void thread\_ap\_idle();

void thread\_ap\_idle\_pp0();

void thread\_ap\_ready();

void thread\_b\_address0();

void thread\_b\_ce0();

void thread\_exitcond1\_fu\_155\_p2();

void thread\_exitcond\_flatten\_fu\_137\_p2();

void thread\_exitcond\_fu\_214\_p2();

void thread\_i\_1\_fu\_149\_p2();

void thread\_indvar\_flatten\_next\_fu\_143\_p2();

void thread\_j\_1\_fu\_274\_p2();

void thread\_j\_mid2\_fu\_161\_p3();

void thread\_k\_1\_fu\_220\_p2();

void thread\_k\_phi\_fu\_130\_p4();

void thread\_p\_shl1\_cast\_fu\_248\_p1();

void thread\_p\_shl\_cast\_fu\_189\_p1();

void thread\_res\_address0();

void thread\_res\_ce0();

void thread\_res\_d0();

void thread\_res\_we0();

void thread\_tmp\_10\_cast\_fu\_209\_p1();

void thread\_tmp\_10\_fu\_252\_p2();

void thread\_tmp\_11\_cast\_fu\_235\_p1();

void thread\_tmp\_11\_fu\_258\_p2();

void thread\_tmp\_14\_cast\_fu\_263\_p1();

void thread\_tmp\_1\_fu\_193\_p2();

void thread\_tmp\_2\_cast\_fu\_199\_p1();

void thread\_tmp\_2\_fu\_230\_p2();

void thread\_tmp\_4\_cast\_fu\_226\_p1();

void thread\_tmp\_4\_fu\_240\_p3();

void thread\_tmp\_fu\_181\_p3();

void thread\_tmp\_mid2\_cast\_fu\_177\_p1();

void thread\_tmp\_mid2\_v\_fu\_169\_p3();

void thread\_tmp\_s\_fu\_203\_p2();

void thread\_ap\_NS\_fsm();

void thread\_hdltv\_gen();

};

}

using namespace ap\_rtl;

#endif