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

// 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 "simple.h"

#include "AESL\_pkg.h"

using namespace std;

namespace ap\_rtl {

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

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

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

const sc\_lv<4> simple::ap\_ST\_fsm\_state2 = "10";

const sc\_lv<4> simple::ap\_ST\_fsm\_state3 = "100";

const sc\_lv<4> simple::ap\_ST\_fsm\_state4 = "1000";

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

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

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

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

simple\_fadd\_32ns\_bkb\_U1 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U1");

simple\_fadd\_32ns\_bkb\_U1->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U1->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U1->din0(a\_0);

simple\_fadd\_32ns\_bkb\_U1->din1(b\_0);

simple\_fadd\_32ns\_bkb\_U1->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U1->dout(grp\_fu\_212\_p2);

simple\_fadd\_32ns\_bkb\_U2 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U2");

simple\_fadd\_32ns\_bkb\_U2->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U2->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U2->din0(a\_1);

simple\_fadd\_32ns\_bkb\_U2->din1(b\_1);

simple\_fadd\_32ns\_bkb\_U2->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U2->dout(grp\_fu\_219\_p2);

simple\_fadd\_32ns\_bkb\_U3 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U3");

simple\_fadd\_32ns\_bkb\_U3->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U3->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U3->din0(a\_2);

simple\_fadd\_32ns\_bkb\_U3->din1(b\_2);

simple\_fadd\_32ns\_bkb\_U3->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U3->dout(grp\_fu\_226\_p2);

simple\_fadd\_32ns\_bkb\_U4 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U4");

simple\_fadd\_32ns\_bkb\_U4->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U4->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U4->din0(a\_3);

simple\_fadd\_32ns\_bkb\_U4->din1(b\_3);

simple\_fadd\_32ns\_bkb\_U4->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U4->dout(grp\_fu\_233\_p2);

simple\_fadd\_32ns\_bkb\_U5 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U5");

simple\_fadd\_32ns\_bkb\_U5->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U5->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U5->din0(a\_4);

simple\_fadd\_32ns\_bkb\_U5->din1(b\_4);

simple\_fadd\_32ns\_bkb\_U5->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U5->dout(grp\_fu\_240\_p2);

simple\_fadd\_32ns\_bkb\_U6 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U6");

simple\_fadd\_32ns\_bkb\_U6->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U6->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U6->din0(a\_5);

simple\_fadd\_32ns\_bkb\_U6->din1(b\_5);

simple\_fadd\_32ns\_bkb\_U6->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U6->dout(grp\_fu\_247\_p2);

simple\_fadd\_32ns\_bkb\_U7 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U7");

simple\_fadd\_32ns\_bkb\_U7->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U7->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U7->din0(a\_6);

simple\_fadd\_32ns\_bkb\_U7->din1(b\_6);

simple\_fadd\_32ns\_bkb\_U7->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U7->dout(grp\_fu\_254\_p2);

simple\_fadd\_32ns\_bkb\_U8 = new simple\_fadd\_32ns\_bkb<1,4,32,32,32>("simple\_fadd\_32ns\_bkb\_U8");

simple\_fadd\_32ns\_bkb\_U8->clk(ap\_clk);

simple\_fadd\_32ns\_bkb\_U8->reset(ap\_rst);

simple\_fadd\_32ns\_bkb\_U8->din0(a\_7);

simple\_fadd\_32ns\_bkb\_U8->din1(b\_7);

simple\_fadd\_32ns\_bkb\_U8->ce(ap\_var\_for\_const0);

simple\_fadd\_32ns\_bkb\_U8->dout(grp\_fu\_261\_p2);

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

dont\_initialize();

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

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

sensitive << ( ap\_CS\_fsm );

SC\_METHOD(thread\_ap\_CS\_fsm\_state4);

sensitive << ( ap\_CS\_fsm );

SC\_METHOD(thread\_ap\_done);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_ap\_idle);

sensitive << ( ap\_start );

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

SC\_METHOD(thread\_ap\_ready);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_ap\_return);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_0);

sensitive << ( grp\_fu\_212\_p2 );

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_0\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_1);

sensitive << ( ap\_CS\_fsm\_state4 );

sensitive << ( grp\_fu\_219\_p2 );

SC\_METHOD(thread\_c\_1\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_2);

sensitive << ( ap\_CS\_fsm\_state4 );

sensitive << ( grp\_fu\_226\_p2 );

SC\_METHOD(thread\_c\_2\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_3);

sensitive << ( ap\_CS\_fsm\_state4 );

sensitive << ( grp\_fu\_233\_p2 );

SC\_METHOD(thread\_c\_3\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_4);

sensitive << ( ap\_CS\_fsm\_state4 );

sensitive << ( grp\_fu\_240\_p2 );

SC\_METHOD(thread\_c\_4\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_5);

sensitive << ( ap\_CS\_fsm\_state4 );

sensitive << ( grp\_fu\_247\_p2 );

SC\_METHOD(thread\_c\_5\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_6);

sensitive << ( ap\_CS\_fsm\_state4 );

sensitive << ( grp\_fu\_254\_p2 );

SC\_METHOD(thread\_c\_6\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

SC\_METHOD(thread\_c\_7);

sensitive << ( ap\_CS\_fsm\_state4 );

sensitive << ( grp\_fu\_261\_p2 );

SC\_METHOD(thread\_c\_7\_ap\_vld);

sensitive << ( ap\_CS\_fsm\_state4 );

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

sensitive << ( ap\_start );

sensitive << ( ap\_CS\_fsm );

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

SC\_THREAD(thread\_hdltv\_gen);

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

SC\_THREAD(thread\_ap\_var\_for\_const0);

ap\_CS\_fsm = "0001";

static int apTFileNum = 0;

stringstream apTFilenSS;

apTFilenSS << "simple\_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, "(port)a\_0");

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

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

sc\_trace(mVcdFile, a\_3, "(port)a\_3");

sc\_trace(mVcdFile, a\_4, "(port)a\_4");

sc\_trace(mVcdFile, a\_5, "(port)a\_5");

sc\_trace(mVcdFile, a\_6, "(port)a\_6");

sc\_trace(mVcdFile, a\_7, "(port)a\_7");

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

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

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

sc\_trace(mVcdFile, b\_3, "(port)b\_3");

sc\_trace(mVcdFile, b\_4, "(port)b\_4");

sc\_trace(mVcdFile, b\_5, "(port)b\_5");

sc\_trace(mVcdFile, b\_6, "(port)b\_6");

sc\_trace(mVcdFile, b\_7, "(port)b\_7");

sc\_trace(mVcdFile, c\_0, "(port)c\_0");

sc\_trace(mVcdFile, c\_0\_ap\_vld, "(port)c\_0\_ap\_vld");

sc\_trace(mVcdFile, c\_1, "(port)c\_1");

sc\_trace(mVcdFile, c\_1\_ap\_vld, "(port)c\_1\_ap\_vld");

sc\_trace(mVcdFile, c\_2, "(port)c\_2");

sc\_trace(mVcdFile, c\_2\_ap\_vld, "(port)c\_2\_ap\_vld");

sc\_trace(mVcdFile, c\_3, "(port)c\_3");

sc\_trace(mVcdFile, c\_3\_ap\_vld, "(port)c\_3\_ap\_vld");

sc\_trace(mVcdFile, c\_4, "(port)c\_4");

sc\_trace(mVcdFile, c\_4\_ap\_vld, "(port)c\_4\_ap\_vld");

sc\_trace(mVcdFile, c\_5, "(port)c\_5");

sc\_trace(mVcdFile, c\_5\_ap\_vld, "(port)c\_5\_ap\_vld");

sc\_trace(mVcdFile, c\_6, "(port)c\_6");

sc\_trace(mVcdFile, c\_6\_ap\_vld, "(port)c\_6\_ap\_vld");

sc\_trace(mVcdFile, c\_7, "(port)c\_7");

sc\_trace(mVcdFile, c\_7\_ap\_vld, "(port)c\_7\_ap\_vld");

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

#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, grp\_fu\_212\_p2, "grp\_fu\_212\_p2");

sc\_trace(mVcdFile, ap\_CS\_fsm\_state4, "ap\_CS\_fsm\_state4");

sc\_trace(mVcdFile, grp\_fu\_219\_p2, "grp\_fu\_219\_p2");

sc\_trace(mVcdFile, grp\_fu\_226\_p2, "grp\_fu\_226\_p2");

sc\_trace(mVcdFile, grp\_fu\_233\_p2, "grp\_fu\_233\_p2");

sc\_trace(mVcdFile, grp\_fu\_240\_p2, "grp\_fu\_240\_p2");

sc\_trace(mVcdFile, grp\_fu\_247\_p2, "grp\_fu\_247\_p2");

sc\_trace(mVcdFile, grp\_fu\_254\_p2, "grp\_fu\_254\_p2");

sc\_trace(mVcdFile, grp\_fu\_261\_p2, "grp\_fu\_261\_p2");

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

#endif

}

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

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

}

simple::~simple() {

if (mVcdFile)

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

mHdltvinHandle << "] " << endl;

mHdltvoutHandle << "] " << endl;

mHdltvinHandle.close();

mHdltvoutHandle.close();

delete simple\_fadd\_32ns\_bkb\_U1;

delete simple\_fadd\_32ns\_bkb\_U2;

delete simple\_fadd\_32ns\_bkb\_U3;

delete simple\_fadd\_32ns\_bkb\_U4;

delete simple\_fadd\_32ns\_bkb\_U5;

delete simple\_fadd\_32ns\_bkb\_U6;

delete simple\_fadd\_32ns\_bkb\_U7;

delete simple\_fadd\_32ns\_bkb\_U8;

}

void simple::thread\_ap\_var\_for\_const0() {

ap\_var\_for\_const0 = ap\_const\_logic\_1;

}

void simple::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();

}

}

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

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

}

void simple::thread\_ap\_CS\_fsm\_state4() {

ap\_CS\_fsm\_state4 = ap\_CS\_fsm.read()[3];

}

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

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

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

} else {

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

}

}

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

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

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

} else {

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

}

}

void simple::thread\_ap\_return() {

ap\_return = ap\_const\_lv32\_0;

}

void simple::thread\_c\_0() {

c\_0 = grp\_fu\_212\_p2.read();

}

void simple::thread\_c\_0\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_0\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_0\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::thread\_c\_1() {

c\_1 = grp\_fu\_219\_p2.read();

}

void simple::thread\_c\_1\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_1\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_1\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::thread\_c\_2() {

c\_2 = grp\_fu\_226\_p2.read();

}

void simple::thread\_c\_2\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_2\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_2\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::thread\_c\_3() {

c\_3 = grp\_fu\_233\_p2.read();

}

void simple::thread\_c\_3\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_3\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_3\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::thread\_c\_4() {

c\_4 = grp\_fu\_240\_p2.read();

}

void simple::thread\_c\_4\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_4\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_4\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::thread\_c\_5() {

c\_5 = grp\_fu\_247\_p2.read();

}

void simple::thread\_c\_5\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_5\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_5\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::thread\_c\_6() {

c\_6 = grp\_fu\_254\_p2.read();

}

void simple::thread\_c\_6\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_6\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_6\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::thread\_c\_7() {

c\_7 = grp\_fu\_261\_p2.read();

}

void simple::thread\_c\_7\_ap\_vld() {

if (esl\_seteq<1,1,1>(ap\_const\_logic\_1, ap\_CS\_fsm\_state4.read())) {

c\_7\_ap\_vld = ap\_const\_logic\_1;

} else {

c\_7\_ap\_vld = ap\_const\_logic\_0;

}

}

void simple::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 :

ap\_NS\_fsm = ap\_ST\_fsm\_state3;

break;

case 4 :

ap\_NS\_fsm = ap\_ST\_fsm\_state4;

break;

case 8 :

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

break;

default :

ap\_NS\_fsm = "XXXX";

break;

}

}

void simple::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() << "\" ";

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

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

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

mHdltvinHandle << " , " << " \"a\_3\" : \"" << a\_3.read() << "\" ";

mHdltvinHandle << " , " << " \"a\_4\" : \"" << a\_4.read() << "\" ";

mHdltvinHandle << " , " << " \"a\_5\" : \"" << a\_5.read() << "\" ";

mHdltvinHandle << " , " << " \"a\_6\" : \"" << a\_6.read() << "\" ";

mHdltvinHandle << " , " << " \"a\_7\" : \"" << a\_7.read() << "\" ";

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

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

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

mHdltvinHandle << " , " << " \"b\_3\" : \"" << b\_3.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_4\" : \"" << b\_4.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_5\" : \"" << b\_5.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_6\" : \"" << b\_6.read() << "\" ";

mHdltvinHandle << " , " << " \"b\_7\" : \"" << b\_7.read() << "\" ";

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

mHdltvoutHandle << " , " << " \"c\_0\_ap\_vld\" : \"" << c\_0\_ap\_vld.read() << "\" ";

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

mHdltvoutHandle << " , " << " \"c\_1\_ap\_vld\" : \"" << c\_1\_ap\_vld.read() << "\" ";

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

mHdltvoutHandle << " , " << " \"c\_2\_ap\_vld\" : \"" << c\_2\_ap\_vld.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_3\" : \"" << c\_3.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_3\_ap\_vld\" : \"" << c\_3\_ap\_vld.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_4\" : \"" << c\_4.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_4\_ap\_vld\" : \"" << c\_4\_ap\_vld.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_5\" : \"" << c\_5.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_5\_ap\_vld\" : \"" << c\_5\_ap\_vld.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_6\" : \"" << c\_6.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_6\_ap\_vld\" : \"" << c\_6\_ap\_vld.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_7\" : \"" << c\_7.read() << "\" ";

mHdltvoutHandle << " , " << " \"c\_7\_ap\_vld\" : \"" << c\_7\_ap\_vld.read() << "\" ";

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

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

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

ap\_cycleNo++;

}

}

}