111064503 吳紹齊 https://github.com/cowboy35927/exam1-p1

1. (**p2 20pt**) Implementation of a decimation filter module with FIFO channels.
    - Please use SystemC FIFO channels to implement the communication interface of the decimation filter
    - Main.cpp
      定義 FIFO channel

```
//Create FIFO channels
sc_fifo<sc_ufixed_fast<4,1>> fifo_i_x;
sc_fifo<sc_ufixed_fast<4,1>> fifo_i_h;
sc_fifo<sc_ufixed_fast<4,1>> fifo_o_y;

//Connect FIFO channels with modules
testbench.i_clk(clk);
testbench.o_rst(rst);
dut.i_clk(clk);
dut.i_rst(rst);
testbench.o_x(fifo_i_x);
testbench.o_h(fifo_i_h);
testbench.i_y(fifo_o_y);
dut.i_x_port(fifo_i_x);
dut.i_h_port(fifo_i_h);
dut.o_y_port(fifo_o_y);
```

    - Adder.h

```
sc_fifo_in< sc_dt::sc_ufixed_fast<4,1>> i_x_port;
sc_fifo_in< sc_dt::sc_ufixed_fast<4,1>> i_h_port;

sc_fifo_out< sc_dt::sc_ufixed_fast<4,1>>  o_y_port;
```

```
void do_add() {
  while (true) {

      _i_x = i_x_port.read();
      _i_h = i_h_port.read();
      wait();
      _o_y =_o_y + _i_x * _i_h;
      if(count == 3){
        count=0;
        o_y_port.write(_o_y);
      }

      wait();
       count++;
  }
}
```

- decimation filter 累加
- Stim.h

```
sc_fifo_out<sc_ufixed_fast<4,1>> o_x;
sc_fifo_out<sc_ufixed_fast<4,1>> o_h;
sc_fifo_in<sc_ufixed_fast<4,1>> i_y;
//Store the previous inputs to FIFOs
sc_ufixed_fast<4,1> t_x;
sc_ufixed_fast<4,1> t_h;
sc_ufixed_fast<4,1> t_y;

void stim_gen() {
  cout << setw(12) << "time" << setw(12) << "a" << setw(12) << "b" << endl;
  for (int a = 0; a < 8*16; a++) {
    for (int b = 0; b < 3; b++) {
      o_x.write(x_input_signal[a*2-b+1]);
      o_h.write(h_k[b]);
      t_x=x_input_signal[a*2-b+1];
      t_h=h_k[b];
      //cout << setw(12) << sc_time_stamp();
      //cout << setw(12) << t_x.to_string(SC_BIN);
      //cout << setw(12) << t_h.to_string(SC_BIN) << endl;
      wait();
    }
  }
}
```

- decimation filter 的設計

```
1 :          7 ns    0b00.100
2 :         13 ns    0b00.110
3 :         19 ns    0b01.000
4 :         25 ns    0b01.010
5 :         31 ns    0b01.101
6 :         37 ns    0b00.000
7 :         43 ns    0b00.011
8 :         49 ns    0b00.110
9 :         55 ns    0b01.001
10 :         61 ns    0b01.100
11 :         67 ns    0b01.111
12 :         73 ns    0b00.010
13 :         79 ns    0b00.101
14 :         85 ns    0b01.000
15 :         91 ns    0b01.100
16 :         97 ns    0b01.111
17 :        103 ns    0b00.010
18 :        109 ns    0b00.101
19 :        115 ns    0b01.000
20 :        121 ns    0b01.011
21 :        127 ns    0b01.110
22 :        133 ns    0b00.001
```

result

1. (**p3 30pt**) Implementation of a decimation filter module with TLM2 interface.
    - (**20pt**) Please use TLM2 blocking transport to implement the communication interface of the decimation filter

```
int sc_main(int argc, char* argv[])
{
  Initiator initiator("initiator");
  Adder   adder("adder");

  // Bind initiator socket to target socket
  initiator.socket.bind( adder.socket );
  sc_start();
  return 0;
}
```

decimation filter 的設計

```
for (int i = 0; i < 16*8; i++)
  for(int j=0;j<3;j++){
  {

    tlm::tlm_command cmd = tlm::TLM_WRITE_COMMAND;
    //prepare 4 bytes (uint8_t)
    data[0]=x_input_signal[i*2-j+1];
    data[1]=h_k[j];
    data[2]=0;
    data[3]=0;

    // Prepare payload
    trans->set_command( cmd );
    trans->set_address( BASE_TARGET_INPUT_ADDR ); //P2P TLM write to target's address 0
    trans->set_data_ptr( reinterpret_cast<unsigned char*>(&data) );
    trans->set_data_length( 4 );
    trans->set_streaming_width( 4 ); // = data_length to indicate no streaming
    trans->set_byte_enable_ptr( 0 ); // 0 indicates unused
    trans->set_dmi_allowed( false ); // Mandatory initial value
    trans->set_response_status( tlm::TLM_INCOMPLETE_RESPONSE ); // Mandatory initial value

    socket->b_transport( *trans, delay );  // Blocking transport call

    // Initiator obliged to check response status and delay
    if ( trans->is_response_error() )
      SC_REPORT_ERROR("TLM-2", "Response error from b_transport");
```

```
// Internal data buffer used by initiator with generic payload
sc_ufixed_fast<4,1> data[4];
sc_ufixed_fast<4,1> result;
```

```
// Check address range and check for unsupported features
if (byt != 0 || len > 4 || wid < len)
  SC_REPORT_ERROR("TLM-2", "Target does not support given generic payload transaction");

// Obliged to implement read and write commands
if ( cmd == tlm::TLM_READ_COMMAND ){
  if(adr==BASE_TARGET_OUTPUT_ADDR){
    //Copy 4 bytes to ptr
    for (unsigned int i = 0; i < len; i++) ptr[i] = sum[i];
    delay=sc_time(5, SC_NS);
  }
  else{
    SC_REPORT_ERROR("TLM-2", "Address not supported for read operation.");
  }
}
else if ( cmd == tlm::TLM_WRITE_COMMAND ){
  if(adr==BASE_TARGET_INPUT_ADDR){
    //Copy 4 bytes from ptr
    for (unsigned int i = 0; i < len; i++) i_data[i] = ptr[i];
    //Compute summation with lower two uint8_t integers
    sum=i_data[0]+i_data[1];
    delay=sc_time(10, SC_NS);
  }
  else{
    SC_REPORT_ERROR("TLM-2", "Address not supported for write operation.");
  }
}
```

```
sc_ufixed_fast<4,1> i_data[SIZE];
sc_ufixed_fast<4,1> sum;
```

Target.h 的設計

- (**10pt**) Please use quantum keeper for timing annotation. The timing parameters of the filter should be derived from the FIFO channel version above.
- ```
  #include "tlm_utils/tlm_quantumkeeper.h"
  ```
- 加上這 include
- ```
  SC_CTOR(Initiator)
  : socket("socket")  // Construct and name socket
  {
    SC_THREAD(thread_process);
    m_qk.set_global_quantum( sc_time(10, SC_NS) );
    m_qk.reset();
  }
  ```
- 初始化 m_qk
- ```
  for (int i = 0; i < 16*8; i++)
    for(int j=0;j<3;j++){
    {
      delay = m_qk.get_local_time();
      tlm::tlm_command cmd = tlm::TLM_WRITE_COMMAND;
  ```
- 得到 local time
- ```
      // Realize the delay annotated onto the transport call
      m_qk.inc( delay );
    // Check if synchronize is necessary
      if (m_qk.need_sync()) m_qk.sync();
  ```
- 將 wait 改成上述兩式，其意義是累加 local time 若超過設置的 quantum time 就進行同步。
- 其他地方都和 TLM 一樣。