312510190 張家瑋

HW3 Report

Simulation results demonstrate the predicted output for the provided input data:

implementation approach:

at core.h, I divide module into two parts: handle_receive and handle transmit function.

```
22      Packet *check_packet;       // Packet for receiving data
23      Packet *packet;      // Packet for transmitting data
24
25      bool no_packet_flag;      // Indicates if there are no packets to packet
26      bool tx_busy;          // Indicates if the transmitter is busy
27      int tx_cnt;  // Transmission counter
28      vector<float> data_in;    // Input data
29      vector<float> data_out;   // Output data
30
31      int core_id;           // Core ID
32      int src_id;            // Source ID
33      int dest_id;           // Destination ID
34      bool input_finish;
35
36      SC_HAS_PROCESS(Core);
37
38      // Main process
39      void run() {
40          if (rst.read() == 1) {
41              initialize();
42          } else {
43              handle_receive();
44              handle_transmit();
45          }
46      }
47
```

For transmission protocol, I use valid and ready flag in AXI-4 protocol to be decide whether the signal can be deliver or not

```
12      sc_in<sc_lv<34>> flit_rx;   // Input channel
13      sc_in<bool> valid_rx;       // Input channel valid signal
14      sc_out<bool> ready_rx;      // Input channel ready signal
15
16      sc_out<sc_lv<34>> flit_tx; // Output channel
17      sc_out<bool> valid_tx;      // Output channel valid signal
18      sc_in<bool> ready_tx;       // Output channel ready signal
19
```

At core.h, there's a problem of translation between sc_lv<> type and float type, so a scfx_ieee_float parameter is necessary to tackle this issue

```
92      // Convert sc_lv to sc_dt::scfx_ieee_float
93      sc_dt::scfx_ieee_float convert_to_float( sc_lv<34> flit_rx_t) {
94          sc_dt::scfx_ieee_float rx_temp;
95          rx_temp.negative(int((sc_uint<1>)(sc_lv<1>)flit_rx_t[31]));
96          rx_temp.exponent(int((sc_uint<8>)(sc_lv<8>)flit_rx_t.range(30,23)));
97          rx_temp.mantissa(int((sc_uint<23>)(sc_lv<23>)flit_rx_t.range(22,0)));
98
99          /*float temp;
100         sc_dt::scfx_ieee_float id(temp); // convert to IEEE 754 bitfield
101         bool sgn = (flit_rx_t[31] == "1");
102         sc_uint<8> exp = flit_rx_t.range(30, 23).to_uint();
103         sc_uint<23> man = flit_rx_t.range(22, 0).to_uint();
104         id.negative(sgn);
105         id.exponent(exp);
106         id.mantissa(man);*/
107         return rx_temp;
108     }
```

At router.h, I create some buffer to store input flit_buffer[5][8]: Buffers to store flits for each input channel. Then, some flag for input ststus buffer_empty[5], buffer_full[5]: Status flags to indicate if a buffer is empty or full.

```
  // check input buffer status
  for(int i = 0; i < 5; i++){
      bool is_empty = (r_addr_buf[i] == w_addr_buf[i]);
      bool is_full = (r_addr_buf[i].range(2,0) == w_addr_buf[i].range(2,0) && r_addr_buf[i][3] != w_addr_buf[i][3]);
      //check whether the input buffer is empty or full
      buffer_empty[i] = (is_empty == 1)? 1 : 0;
      buffer_full[i] = (is_full == 1)? 1 : 0;
      if(buffer_full[i] == 1){

          //cout << "Router" << Router_ID << " In" << i << " is full.\n";
          in_ready[i].write(0);
      }
      else{
          in_ready[i].write(1);
      }

  }
```

For Routing and Channel Selection, channel_select[5]: Keeps track of the selected channel for each output. out_busy[5], out_valid_tmp[5]: Status flags to manage output channels.

```
111
112         // Output selection for the local output channel
113         if (out_busy[0] == 0) {
114             bool found_flit = false;
115             // Iterate over all input buffers
116             for (int i = 0; i < 5; i++) {
117                 // output selection
118                 if (buffer_empty[i] == 0) {
119                     sc_lv<34> flit = flit_buffer[i][r_addr_buf[i].range(2,0)];
120                     int flit_router_id = int((sc_uint<4>)(sc_lv<4>)flit.range(27,24));
121                     bool is_head_flit = flit[33] == 1;
122 
123                     if (flit_router_id == Router_ID && is_head_flit) {
124                         channel_select[0] = i;
125                         out_valid_tmp[0] = 1;
126                         found_flit = true;
127                         break;// found the flit
128                     }
129                 }
130             }
131             // If no valid flit found, set the output to invalid
132             if (!found_flit) {
133                 channel_select[0] = 5;
134                 out_valid_tmp[0] = 0;
135             }
136         } else {
137             // Output channel is busy, retain current buffer selection
138             channel_select[0] = channel_select[0];
139             out_valid_tmp[0] = 1;
140         }
```

And then, Implements the XY routing algorithm to determine the correct output channel for each flit. Checks buffer statuses (empty/full) and selects appropriate flits for each output channel. Updates out_valid_tmp and channel_select based on the routing logic. Writes valid outgoing flits to the output ports and sets the corresponding out_valid signals.

```
int Router_ID;
SC_HAS_PROCESS(Router);
void run(){
    if(rst.read() == 1){
        initialize_router();
    }
    else{
        handle_rx();
        habdle_tx();
        xy_routing();

    }
}
```

challenges faced:

while doing core.h, using type scfx_ieee_float is a trouble, it must be first change to sc_lv vector type, then switch to sc_uint type to satisfying the flits convert to float value

```
92      // convert sc_lv to sc_dt::scfx_ieee_float
93      sc_dt::scfx_ieee_float convert_to_float( sc_lv<34> flit_rx_t) {
94          sc_dt::scfx_ieee_float rx_temp;
95          rx_temp.negative(int((sc_uint<1>)(sc_lv<1>)flit_rx_t[31]));
96          rx_temp.exponent(int((sc_uint<8>)(sc_lv<8>)flit_rx_t.range(30,23))
97          rx_temp.mantissa(int((sc_uint<23>)(sc_lv<23>)flit_rx_t.range(22,0)
98
99          /*float temp;
100         sc_dt::scfx_ieee_float id(temp); // convert to IEEE 754 bitfield
101         bool sgn = (flit_rx_t[31] == "1");
102         sc_uint<8> exp = flit_rx_t.range(30, 23).to_uint();
103         sc_uint<23> man = flit_rx_t.range(22, 0).to_uint();
104         id.negative(sgn);
105         id.exponent(exp);
106         id.mantissa(man);*/
107         return rx_temp;
108     }
109
```

other observations or insights gained:

XY Routing is a widely used deterministic routing algorithm for mesh topology networks-on-chip (NoC). The algorithm routes packets first along the X-axis (horizontal direction) and then along the Y-axis (vertical direction) to reach their destination. This simplicity ensures deadlock-free operation and is easy to implement.

For the algorithm: X-Phase: If the destination is to the right (east), route the packet to the east neighbor. If the destination is to the left (west), route the packet to the west neighbor. Y-Phase: If the destination is below (south), route the packet to the south neighbor. If the destination is above (north), route the packet to the north neighbor.

```
//2: South output channel
if (out_busy[2] == 0) {
    bool found_flit = false;
    // Iterate over all input buffers
    for (int i = 0; i < 5; i++) {
        // output selection
        if (buffer_empty[i] == 0) {
            sc_lv<34> flit = flit_buffer[i][r_addr_buf[i].range(2,0)];
            int xy_route_direct = int((sc_uint<4>)(sc_lv<4>)flit.range(27,24)) / 4;
            bool is_head_flit = flit[33] == 1;
            bool select_correct = (channel_select[1] != i && channel_select[3] != i);
            if (xy_route_direct > Router_ID/4 && is_head_flit && select_correct) {
                channel_select[2] = i;
                out_valid_tmp[2] = 1;
                found_flit = true;
                break;// found the flit
            }
        }
    }
    // If no valid flit found, set the output to invalid
    if (!found_flit) {
        channel_select[2] = 5;
        out_valid_tmp[2] = 0;
    }
} else {
    // Output channel is busy, retain current buffer selection
    channel_select[2] = channel_select[2];
    out_valid_tmp[2] = 1;
}
```