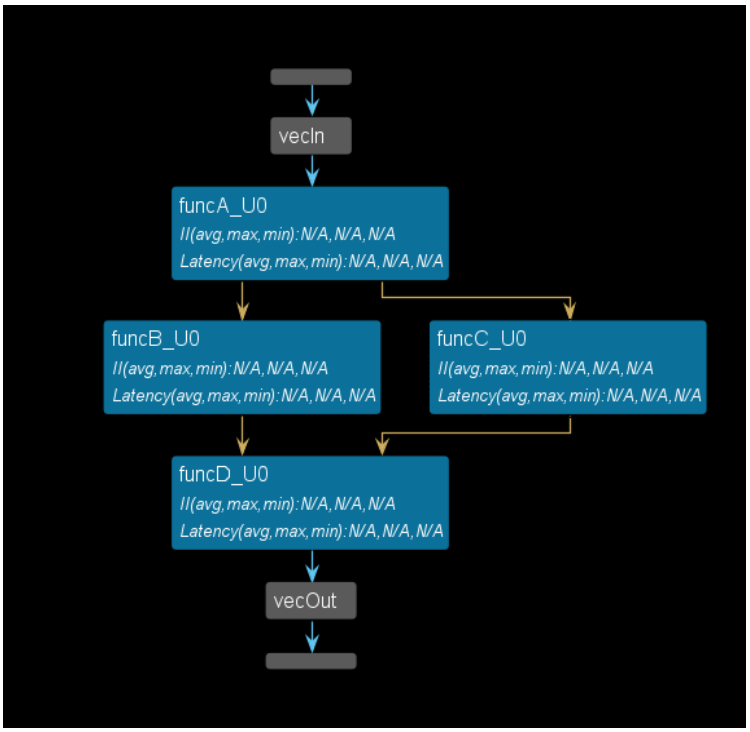
AAHLS Lab A Report R11943109 何國瑋

First Lab: Dataflow Viewer Basics

Introduction to the Overall System



The overall system consists of four arithmetic functions, and the datapath is shown in the above figure from the dataflow viewer.

Code Explanation

```
void diamond(data_t vecIn[N], data_t vecOut[N])
{
  data_t c1[N], c2[N], c3[N], c4[N];
#pragma HLS dataflow
  funcA(vecIn, c1, c2);
  funcB(c1, c3);
  funcC(c2, c4);
  funcD(c3, c4, vecOut);
}
```

Four arithmetic functions are just simple addition and multiplication.
The top function (diamond) defines the datapath of these functions.

```
// Executing the DUT thrice
for (int iter = 0; iter < 3; iter++)
{
  // Execute DUT
  diamond(test, outcome);
```

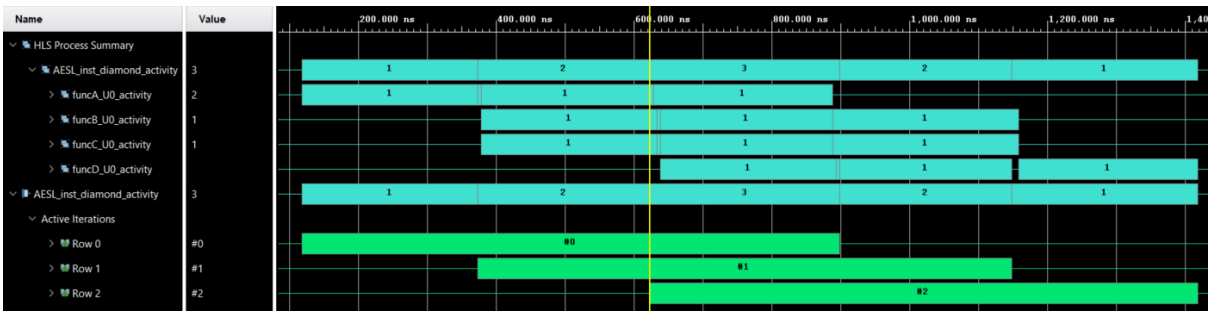
The testbench simply calls the top function three times.

Observation

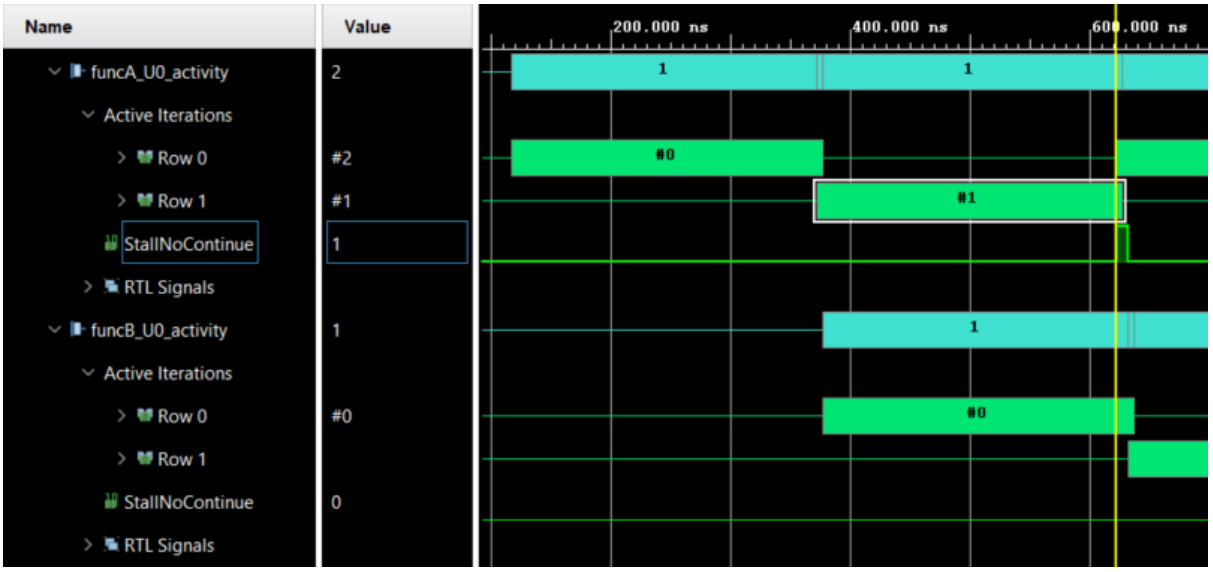
After the cosimulation, the latency and II (if there're two or more calls to the function) of each function are displayed on the datapath diagram as shown in the above figure.

Name	Cosim Category	Cosim Stalling Time	FIFO EMPTY	FIFO FULL	Cosim Stall No Start	Cosim Stall No Continue
funcA_U0	none	0.76%	0.00%	0.00%	0.00%	0.76%
funcB_U0	none	0.76%	0.00%	0.00%	0.38%	0.76%
funcC_U0	none	0.76%	0.00%	0.00%	0.38%	0.76%
funcD_U0	none	1.15%	0.00%	0.00%	1.15%	0.00%

More detailed statistics are shown in the dataflow process table (shown in the above figure). The columns "Cosim Stall No Start" and "Cosim Stall No Continue" indicates what percentage of the simulation time was spent stalling due to forward and back pressure respectively. This information helps the designer to locate the bottleneck of the system's computation, so the designer can adjust the datapath or change the implementation of some functions to achieve a better performance.



In the waveform viewer, we can see the detailed simulation timeline. The above timeline shows the iteration and the activity of each function. We can then check how the execution of functions overlap, and when the stalls actually happen.



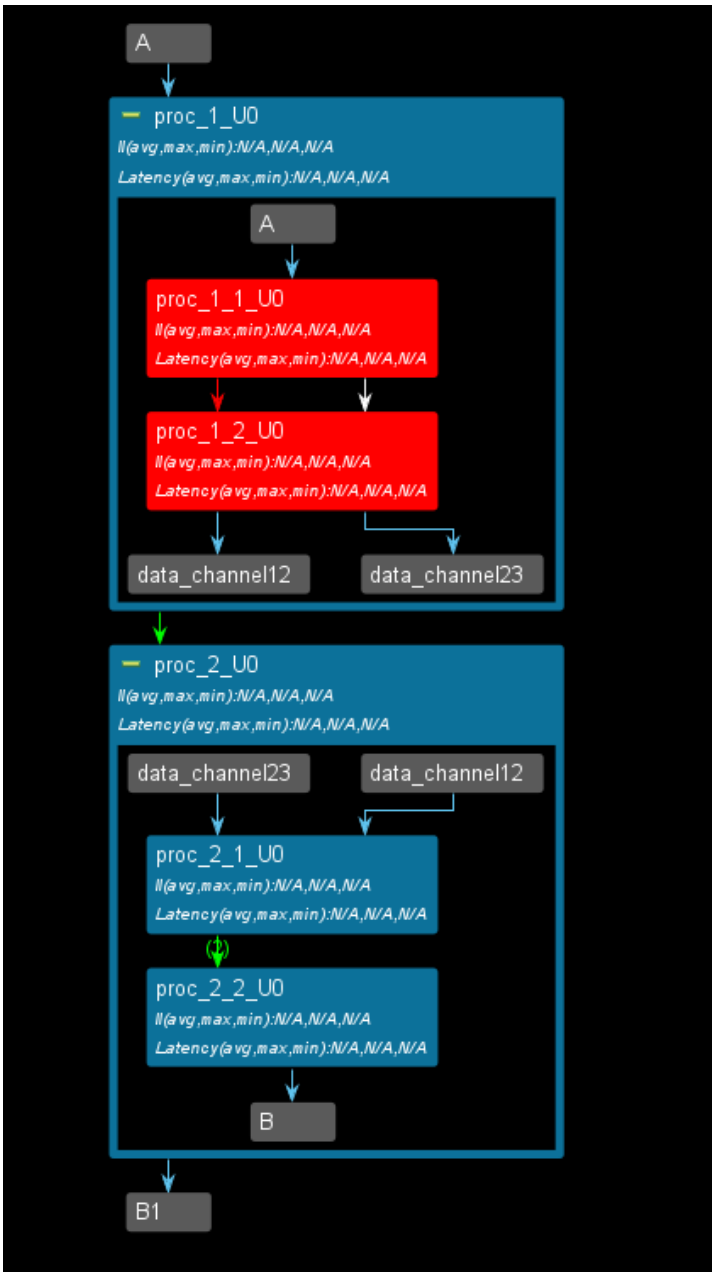
For example, from the above figure, we can see that when the iteration #1 of function A finished, function B is still in the computation iteration #0, causing a stall and increase the latency of iteration #1.

Problem Encountered

No significant problem encountered in the fist lab.

Second Lab: FIFO Sizing and Deadlock

Introduction to the Overall System



The datapath of the overall system is shown in the above figure. The system consists of four subfunctions. All the subfunctions read/write data from/to their corresponding i/o channels 10 times in an execution.

Code Explanation

The top level function simply defines the overall datapath in a hierarchical description.

```
void proc_1_2(hls::stream<int>& B, hls::stream<int>& C, hls::stream<int>& data_channel1, hls::stream<int>& data_channel2){
  int i;
  int tmp;

  for(i = 0; i < 10; i++){
    tmp = data_channel2.read() + data_channel1.read();
    B.write(tmp);
  }
  for(i = 0; i < 10; i++){
    C.write(tmp);
  }
}

void proc_2(hls::stream<int>& A, hls::stream<int>& B, hls::stream<int>& C){
#pragma HLS dataflow
    hls::stream<int> data_channel1;
    hls::stream<int> data_channel2;

    proc_2_1(A, B, data_channel1, data_channel2);
    proc_2_2(C, data_channel1, data_channel2);
}

void proc_2_1(hls::stream<int>& A, hls::stream<int>& B, hls::stream<int>& data_channel1, hls::stream<int>& data_channel2){
  int i;
  int tmp;
  for(i = 0; i < 10; i++){
    tmp = A.read() + B.read();
    data_channel1.write(tmp);
  }
  for(i = 0; i < 10; i++){
    data_channel2.write(tmp);
  }
}
```

The above code snippet shows that the subfunctions “proc_2_1” and “proc_1_2” both include a simple addition of two input data. The other two functions only perform read/write operations.

```
int main()
{
  int i;
  hls::stream<int> A;
  hls::stream<int> B;

  int time = 0;
  for   (time = 0 ; time < 4; time ++) {
      for(i=0; i < SIZE; i++){
        A << (i + time);
      }

      example(A,B);

  }
  return 0;
}
```

The above code snippet shows that the testbench simply executes the top function for times.

Observation

Name	Cosim Category	FIFO EMPTY	FIFO FULL	Cosim Max Depth	Depth
data_channel1	read_block	99.61%	0.00%	0	2
data_channel2	read_block	99.61%	0.00%	0	2
data_channel1	write_block	0.00%	99.51%	2	2
data_channel2	read_block	99.61%	0.00%	0	2
data_channel1	read_block	99.61%	0.00%	0	2
data_channel2	read_block	99.61%	0.00%	0	2

The above table shows the original FIFO depth with deadlock.

When manually performing the FIFO sizing, if one follows the tutorial, one will have to modify and run the cosimulation multiple times and try different FIFO depth. The resulting FIFO depth is 10 for all channels. This process is like debugging (trial and error). I think this method is better when the designer has certain knowledge about the design.

When using the global FIFO sizing method, one can resolve the deadlock in one modification. The resulting FIFO depth is 40 for all channels. This method is simple and requires no knowledge about the design, but could result in consuming unnecessary resources as in this example.

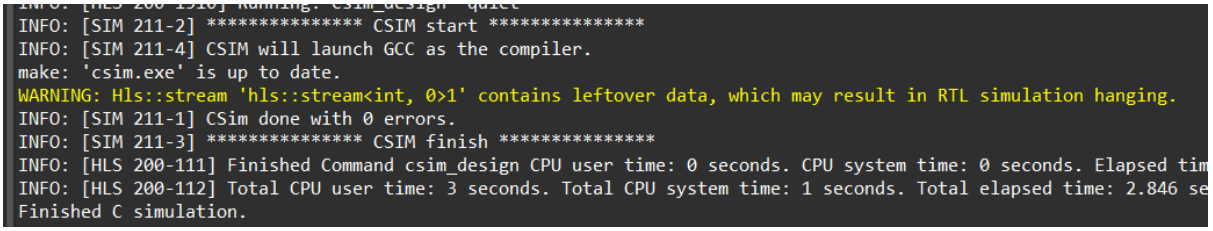
Name	Suggest Depth
data_channel1	11
data_channel2	2
data_channel1	11
data_channel2	2
data_channel1	11
data_channel2	2

When using the automatic FIFO sizing method, as shown in the above table, the result is pretty good in this example. However, as I said, if one has certain knowledge about the design, one can achieve a better solution (with fewer resources) than this heuristic method. For example, the following settings with smaller FIFO depths can also resolve the deadlock.

Name	Cosim Category	FIFO EMPTY	FIFO FULL	Cosim Max Depth	Depth
data_channel1	read_block	10.45%	0.00%	10	10
data_channel2	read_block	22.39%	0.00%	1	2
data_channel1	none	0.00%	0.00%	10	10
data_channel2	read_block	11.94%	0.00%	1	2
data_channel1	read_block	25.37%	0.00%	10	10
data_channel2	read_block	64.18%	0.00%	1	2

Problem Encountered

When I tried the “global FIFO sizing” method, I didn’t see the maximum depth printed by the console.



After the C simulation, instead of the maximum depth, the console showed this warning of leftover data.

Another small problem encountered is that, when using OnlineFPGA, I couldn’t open vitis since vitis encountered an error when executing script.tcl (the command set_part virtex7).

GitHub Link

https://github.com/allen1236/AAHLS_LabA