EE 3220 – System-on-Chip Design Assignment 3

Run C Simulation

ex.cpp | ex_tb.cpp | ALU_csim.log | Synthesis Summary(solution1) | Co-simulation Report(solution1)

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../src/ex_tb.cpp in debug mode
   Compiling ../../../src/ex.cpp in debug mode
   Generating csim.exe
mode: 0
mode: 1
mode: 2
mode: 3
mode: 4
mode: 5
mode: 6
mode: 7
mode: 8
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************

```

Run C Synthesis

ex.cpp | ex_tb.cpp | ALU_csim.log | Synthesis Summary(solution1) | Co-simulation Report(solution1)

Synthesis Summary Report of 'ALU'

General Information

Date: Fri Apr 15 09:19:15 2022
Version: 2021.2.1 (Build 3414424 on Sun Dec 19 10:57:22 MST 2021)
Project: ALU
Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z010i-clg225-1L

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	4.552 ns	2.70 ns

Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
ALU				-	0	0.0	-	1	-	no	0	0	157	573	0

HW Interfaces

S_AXILITE Interfaces

Interface	Data Width	Address Width	Offset	Register
s_axi_control	32	6	16	0

Run Co-simulation

Cosimulation Report for 'ALU'

General Information

Date: Fri Apr 15 09:25:13 2022
Version: 2021.2.1 (Build 3414424 on Sun Dec 19 10:57:22 MST 2021)
Project: ALU
Status: Pass
Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z010i-clg225-1L

Cosim Options

Tool: Vivado XSIM
RTL: Verilog
Optimizing Compile: True

Performance Estimates

Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
ALU	47	51	47	0	0	0

ALU description:

Mode 000: Add, $Y = A + B$

Mode 010: Subtract, $Y = A - B$

Mode 010: Decrement, $Y = A - 1$

Mode 011: Increment, $Y = A + 1$

Mode 100: 1's complement, $Y = not\ A$

Mode 101: Bitwise AND, $Y = A\ and\ B$

Mode 110: Bitwise OR, $Y = A\ or\ B$

Mode 111: Bitwise XOR, $Y = A\ xor\ B$

Mode default: Return 0, $Y = 0$

Question 4:

The model design was written in software with C++ and the software will test, verify the design of the IP or the FPGA and translate the code to Verilog or VHDL.

Advantage:

Productivity gain, the design period can be shortened with using HLS as the code are generated by the software instead writing the code from scratch. In the tutorials, it is much faster to develop a complicated design with HLS then using VHDL.

Optimization, the code will be more optimized by the software. From the tutorial, it is shown that the software will optimize the code instead of optimize by the programmer which requires a lot of experience on FPGA designs.

Appendix:

ex.cpp

```cpp
#include "ex.h"

void ALU(int a, int b, int &y, short mode)
{
#pragma HLS INTERFACE s_axilite port = return
#pragma HLS INTERFACE s_axilite port = a
#pragma HLS INTERFACE s_axilite port = b
#pragma HLS INTERFACE s_axilite port = y

    switch (mode)
    {
    case 0:
        y = a + b;
        break;
    case 1:
        y = a - b;
        break;
    case 2:
        y = a - 1;
        break;
    case 3:
        y = a + 1;
        break;
    case 4:
        y = ~a;
        break;
    case 5:
        y = a & b;
        break;
    case 6:
        y = a | b;
        break;
    case 7:
        y = a ^ b;
        break;
    default:
        y = 0;
        break;
    }
}
```

ex.h

```cpp
#ifndef __EX__
#define __EX__

void ALU(int a, int b, int &y, short mode);

#endif
```

ex_tb.cpp

```cpp
#include <iostream>
#include "ex.h"

using namespace std;

int main()
{
    int a;
    int b;
    int y;
    int gy;

    for (short mode = 0; mode <= 8; mode++)
    {
        cout << "mode: " << mode << endl;
        for (a = -100; a < 100; a++)
        {
            for (b = -100; b <= 100; b++)
            {
                switch (mode)
                {
                case 0:
                    gy = a + b;
                    break;
                case 1:
                    gy = a - b;
                    break;
                case 2:
                    gy = a - 1;
                    break;
                case 3:
                    gy = a + 1;
                    break;
                case 4:
                    gy = ~a;
                    break;
                case 5:
                    gy = a & b;
                    break;
                case 6:
                    gy = a | b;
                    break;
                case 7:
                    gy = a ^ b;
                    break;
                default:
                    gy = 0;
                    break;
                }

                ALU(a, b, y, mode);
                if (gy != y)
```

```
                {
                    cout << "Error when a=" << a << ", b=" << b;
                    return 1;
                }
            }
        }
    }

    return 0;
}
```