**WES237C: Project#1 Report**

**<https://github.com/aubaez/WES237C_Project1>**

1. **Variable Bitwidths:**

Performance for Baseline Code:

|  |  |
| --- | --- |
| Total CPU user time | 4.16 seconds |
| Total CPU system time | 1.02 seconds |
| Peak allocated memory | 252.090 MB |
| Throughput | Estimated Fmax 144.68 MHz |

I changed the bitwidth of the coefficients array data type by declaring coef\_t as ap\_int<10> in the fir.h file. After running C Synthesis I observed:

|  |  |
| --- | --- |
| Total CPU user time | 4.23 seconds |
| Total CPU system time | 1.05 seconds |
| Peak allocated memory | 266.055 MB |
| Throughput | Estimated Fmax 144.68 MHz |

With coef\_t declared as ap\_int<32>:

|  |  |
| --- | --- |
| Total CPU user time | 4.27 seconds |
| Total CPU system time | 1.12 seconds |
| Peak allocated memory | 261.121 MB |
| Throughput | Estimated Fmax 144.68 MHz |

With coef\_t declared as ap\_int<3>:

@E Simulation failed: Function 'main' returns nonzero value '1'.

ERROR: [SIM 211-100] 'csim\_design' failed: nonzero return value.

With coef\_t declared as ap\_int<5>:

|  |  |
| --- | --- |
| Total CPU user time | 4.39 seconds |
| Total CPU system time | 1.07 seconds |
| Peak allocated memory | 272.383 |
| Throughput | Estimated Fmax 144.68 MHz |

***Summary***: Changing the bitwidth of the variables in the fir function didn’t affect the

throughput but the CPU time and Memory allocation seemed to increase every time I ran my code. I captured all the performance data after running C Simulation and C Synthesize. If I only look at the CPU times after running C Simulation, then I see much better values. I observed that the smallest data size I could use was a 5 bitwidth without losing data accuracy.

1. **Pipelining:**

By inserting #pragma HLS pipeline II=1 after the for loop header in the fir function definition, I was able to optimize the for loop by pipelining. Keeping the typedef coef\_t as running C Simulation then C Synthesis, I got the following performance values:

|  |  |
| --- | --- |
| Total CPU user time | 4.35 seconds |
| Total CPU system time | 1.16 seconds |
| Peak allocated memory | 266.102 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 133 |

By inserting #pragma HLS pipeline II=2 after the for loop header I got the following performance values:

|  |  |
| --- | --- |
| Total CPU user time | 4.31 seconds |
| Total CPU system time | 1.08 seconds |
| Peak allocated memory | 266.086 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 260 |

By inserting #pragma HLS pipeline II=3 after the for loop header I got the following performance values:

|  |  |
| --- | --- |
| Total CPU user time | 4.29 seconds |
| Total CPU system time | 1.06 seconds |
| Peak allocated memory | 266.090 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 387 |

By inserting #pragma HLS pipeline II=4 after the for loop header I got the following performance values:

|  |  |
| --- | --- |
| Total CPU user time | 4.38 seconds |
| Total CPU system time | 1.09 seconds |
| Peak allocated memory | 266.051MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 514 |

By inserting #pragma HLS pipeline II=5 after the for loop header I got the following performance values:

|  |  |
| --- | --- |
| Total CPU user time | 4.21 seconds |
| Total CPU system time | 1.08seconds |
| Peak allocated memory | 265.977 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 642 |

***Summary*:** Increasing the II causes the loop latency to increase. Subsequently, this causes for the CPU user time and system time to decrease as the II increases. The peak memory allocated also seems to be decreasing. Setting the II to a larger value would not make sense once the loop latency equals or exceeds that of the non-pipelined version.

1. **Removing Conditional Statements:**

To remove the conditional, I moved the check for “if(I == 0)” in the for loop and execute this check as the 0th condition outside of the for loop. Within the for loop, I removed the check for the 0th condition and executed the code as normal.

The performance for the optimized code was as follows:

|  |  |
| --- | --- |
| Total CPU user time | 3.99 seconds |
| Total CPU system time | 1.11 seconds |
| Peak allocated memory | 265.898MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 132 |

As compared to the code without the conditional statement removed:

|  |  |
| --- | --- |
| Total CPU user time | 4.35 seconds |
| Total CPU system time | 1.16 seconds |
| Peak allocated memory | 266.102 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 133 |

***Summary***: Memory allocation and CPU user time/system time were both much better with the optimization. Also, the loop latency improved from 133 cycles to 132. The performance observed was at follows:

1. **Loop Partitioning:**

I loop partitioned the fir filter by using #pragma HLS unroll factor=2 in my for loop. I didn’t observe much better performance in CPU time, loop latency increased and throughput also decreased.

|  |  |
| --- | --- |
| Total CPU user time | 4.36 seconds |
| Total CPU system time | 1.15 seconds |
| Peak allocated memory | 269.770 MB |
| Throughput | Estimated Fmax 143.31 MHz |
| Loop Latency (cycles) | 147 |

1. **Memory Partitioning:**

When doing memory partitioning by parameter complete, my C synthesis was taking too long to run, so I had to force stop.

When doing the memory partitioning by parameter block, my performance was as follows:

|  |  |
| --- | --- |
| Total CPU user time | 4.3 seconds |
| Total CPU system time | 1.18 seconds |
| Peak allocated memory | 269.988 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 135 |
| BRAM | 6 |
| FF | 911 |

When doing the memory partitioning by parameter cyclic, my performance was as follows:

|  |  |
| --- | --- |
| Total CPU user time | 4.2 seconds |
| Total CPU system time | 1.08 seconds |
| Peak allocated memory | 266.758 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 72 |
| BRAM | 1 |
| FF | 879 |

***Summary***: The best performance was produced by the cyclic memory partitioning.

1. **Best Design:**

My best design would be the same as I got for question 5. I tried different combinations of the optimizations, but the one in question 5 had the best results. I included all the optimizations as follows:

1. bitwidth of the coefficients array data type: ap\_int<10>
2. pipelining with #pragma HLS pipeline II=1
3. removing conditional statement in for loop
4. loop unrolling with #pragma HLS unroll factor=2
5. cyclic memory partitioning.

|  |  |
| --- | --- |
| Total CPU user time | 4.2 seconds |
| Total CPU system time | 1.08 seconds |
| Peak allocated memory | 266.758 MB |
| Throughput | Estimated Fmax 144.68 MHz |
| Loop Latency (cycles) | 72 |
| BRAM | 1 |
| FF | 879 |

**Note**: I didn’t realize there was a formula for throughput, so I used the value given in the vitis\_hls Guidance tab.A screenshot of a computer

Description automatically generated