Introduction

Document to test HLS optimizations on matrix convolution

Starting point

After experimenting, we can segregate our problem into two categories

1. I/O
2. Computation

Optimizing both individually and/or parallelizing them is the key to reducing latency.

Using wider data buses

Vitis HLS 2020.2 - Automatic Bus Widening for AXI Interfaces

Synthesis summary : Hardware interfaces

Max widen bit width = 512

Test 1

Description

In the loop to load the matrix into RAM

1. Unroll the loop by 16
2. Removed depth parameter from interface definition

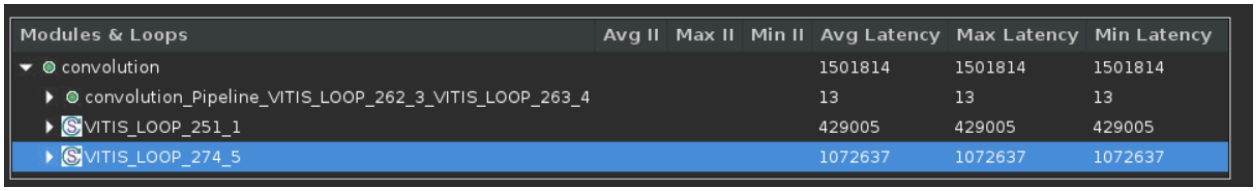
Results

1. In the loop loading the matrix from interface to temp matrix the latency reduced to **8ms** from **16ms** but it gives a II violation resource limitation. Unable to enforce a carried dependence constraint and unable to schedule a load operation.
2. The computation loop takes 84 ms to complete with a II violation resource limitation.
    a. Solution for the above problems is to ***unroll the convolution loop by 16*** as well partitioning the temp_matrix ***cyclically*** by ***16.***
3. Max widen bitwidth not achieved in synthesis.
4. II and depth.
    a. Loop
    b. Loop
    c. Loop
5. Latency = 13,79,302 cycles
6.

BRAM	DSP	FF	LUT

4096`	434	12K	18K

RTL-CoSimulation



Modules & Loops	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
convolution				1501814	1501814	1501814
convolution_Pipeline_VITIS_LOOP_262_3_VITIS_LOOP_263_4				13	13	13
VITIS_LOOP_251_1				429005	429005	429005
VITIS_LOOP_274_5				1072637	1072637	1072637

Clock period = 10ns
Execution time ~ 15ms

Issues

Unable to enforce a carried dependence constraint between bus read operation ('gmem0_addr_1_read_3', Convolution.cpp:268) on port 'gmem0' (Convolution.cpp:268) and bus read operation ('gmem0_addr_1_read', Convolution.cpp:268) on port 'gmem0' (Convolution.cpp:268).

Reason:
Multiple reads at the same time. Try partitioning the array. Block partition fails; cyclic partition works.

Inference and what to do next

1. Look at schedule viewer.
2. Change Unroll and cyclic partition factor
3. What is the logo with the S and the arrow.
4. Understand what is empty read req.
5. Methods to optimize computation :
    a. Spend a few clock cycles to load data into temp matrix and temp kernel and then compute and input data in parallel.
    b. Divide the matrix into two and compute in parallel.
        i. Need to account for edge pixel cases.
6. What is **interval** in the synthesis summary?

Test 2

Description

1. Change loop unroll and cyclic partition factor.
    a. Try with 20 and 32(Does making it a multiple of 16 change the result).

Results

1. **Unroll and partition = 20**
    a. Latency = 13,67,526 cycles
    b. Hardware usage

BRAM	DSP	FF	LUT
5120`	541	13K	18K

    c. Pipelining result :
        i. Loading data loop : II = 5 , Depth = 7
        ii. Loading kernel loop: II = 1, Depth = 3
        iii. Computation loop: II = 10, Depth = 16

2. **Unroll and partition = 32**
    a. Latency = 13,79,302 cycles
    b. Hardware usage

BRAM	DSP	FF	LUT
4096`	866	22K	29K

    c. Pipelining result :
        i. Loading data loop : II = 8 , Depth = 9
        ii. Loading kernel loop: II = 1, Depth = 3
        iii. Computation loop: II = 16, Depth = 22

3. Anything weird in console output?

Test 3

Description

1. Load a part of the data into RAM.
2. Begin computation and loading the rest of the data in parallel.

Results

1. Loaded 500 values. Simulation results in a lot of zeros in the output matrix. Possibly a race condition where computation is faster than loading data. Might be fixed by loading more data initially.

Issues

1. After temp matrix is partitioned : Non-shared array 'temp_matrix.5' failed dataflow checking: it can only have a single reader and a single writer.
    a. Temp_matrix is written to in the loop for loading partial matrix.
    b. Temp_matrix is written to in function for loading the remaining matrix.
    c. Is read in compute convolution function

2.

Test 4

Description

In the convolution code, change 1298 to 1300 ,synthesize and check for changes in latency. After seeing the result, change output size to 32 bit instead of 64 bit and see if anything changes. Look for ways to make the HW interface size 512 bits.

Results

1. Latency drops to 9.58 ms - 8.7ms
    a. 4.74/4.31 ms for input.
    b. 4.83/4.39 ms for computation and output.
2. HW interface synth summary for 1298 in convolution loop (64 bit output)

Interface	Data Width (SW->HW)
m_axi_gmem0	32 -> 128
m_axi_gmem1	32 -> 32
m_axi_gmem2	64 -> 128

3. HW interface synth summary for 1300 in convolution loop (64 bit output).

Interface	Data Width (SW->HW)
m_axi_gmem0	32 -> 128
m_axi_gmem1	32 -> 32
m_axi_gmem2	64 -> 256

4. When the output is set to 32 bit the latency does not change but the data width of m_axi_gmem2 changes from 32 => 128.

Inference

1. The interface size increases because of **automatic port widening** (Refer to synthesis summary document: Reference 4).
2. When you adjusted the loop bounds from 1298 to 1300, you aligned the loop bounds to a closer power-of-2 boundary. This can allow the HLS tool to apply optimizations more effectively, such as pipelining and loop unrolling.
3. By moving to 1300, you might have eliminated padding or partial bursts that would have been required with 1298, allowing for fully efficient burst transfers across the interfaces.
4. Computation and writing to output takes 4.8 ms. Can we store the intermediate output on BRAM during computation and write to output later???

https://github.com/Xilinx/Vitis-HLS-Introductory-Examples/tree/master/Interface/Streaming/axi_stream_to_master

Streams can be used in top-level functions.

Test 5

Description

1. Store output on BRAM using a temp variable.
2. Write the output to the interface after computation / in parallel.
3. Partition the output temporary array ***cyclically*** with a ***factor of 16.***

Results

1. Latency = 10ms
2. The computation only takes 1.3ms
3. Hardware usage

BRAM	DSP	FF	LUT
24K`	434	13K	17K

4. Pipelining results

Inference

Test 6

Description

1. Use **hls::vector or ap_(u)int<512>** as the data type of the port to infer **maximum burst lengths.** Check hw interface synth summary.
2. Loop merge pragma
3. LOOP_FLATTEN pragma

Code changes tried

1. Ignore loading the matrix into temp_matrix and compute directly

Results

Interface	Data Width (SW->HW)
m_axi_gmem0	512 -> 512
m_axi_gmem1	32 -> 32
m_axi_gmem2	512 -> 512

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	L
convolution				-	632044	
convolution_Pipeline_VITIS_LOOP_81_1_VITIS_LOOP_82_2				-	12	
convolution_Pipeline_VITIS_LOOP_88_3_VITIS_LOOP_89_4	II Violation			-	631885	

BRAM	DSP	FF	LUT
0	291	46307	14314
0	0	46	137
0	291	41485	6764

II = 1 and depth = 7

Inference

Resources

1. Deprecated and Unsupported Features - 2024.1 English
2. Array-Specific Dataflow Caches for High-Level Synthesis of Memory-Intensive Algorithms on FPGAs | IEEE Journals & Magazine
3. Synthesis Summary - 2024.1 English

4.

VIVADO HLS 2D Convolution on hardware - part 1
A tutorial on non-separable 2D convolutions in Vivado HLS – Basile Van Hoorick

Can streams have multiple readers and writers???

Blog content planning

1. Introduction
    a. Introduction to HLS
    b. Problem description
    c. Applications of convolution?
    d. Purpose of the project
        i. Thought process oriented.
        ii. Research ways to optimize throughput and latency for this application
        iii. The goal is not to achieve a certain latency.
        iv. The goal is to understand the problem and think of ways that you can keep reducing the latency.
        v. Learn about optimizations on an FPGA.
        vi. Constraints (Time, skill etc.)?
2. Baseline performance
    a. The II was very high (introduce II and put ss).
3. Introduce interfaces and move to the next optimization
4. Store data in temporary memory
    a. This is where we get a better picture of what exactly we are dealing with and what problem we're gonna solve.
    b. Designing hardware is not the same as writing code (HLS makes it tricky to think differently). Coding is sequential thinking HW is more parallelism oriented.
    c. In hardware, we bring data in, compute and write data out.
    d. Show synth summary.
5. Loop unroll and partition by 16
    a. Explain why we thought of this.
    b. (Did it change the HW interface section of the synth summary. 4 times more improvement).
6. Separating writing to output from computation (Might not be necessary)
7. HLS vector code.
    a. Explain in detail and final result
8. Changing from 1298 to 1300.
9. Improvements
    a. One way to optimize this further is exploiting more parallelism.
        i. We bring a certain amount of data

ii. Begin computation and bring in next data and write to output in parallel

b. Minimize memory accesses.