L12: DSP Techniques

Cong (Callie) Hao

callie.hao@ece.gatech.edu

Assistant Professor
ECE, Georgia Institute of Technology

Sharc-lab @ Georgia Tech https://sharclab.ece.gatech.edu/
Outline

• C/RTL Co-Simulation

• Efficient DSP usage for low-bitwidth computation
void fooA(FIX_TYPE A[100][100])
{
    #pragma HLS inline off
    for(int j = 0; j < 100; j++) {
        for(int i = 0; i < 100; i++) {
            A[i][j] += 1;
        }
    }
}

void fooB(FIX_TYPE A[100][100])
{
    #pragma HLS inline off
    for(int j = 0; j < 100; j++) {
        for(int i = 0; i < 100; i+=2) {
            A[i][j] += 1;
        }
    }
}

void test(FIX_TYPE A[100][100],
           FIX_TYPE B[100][100],
           FIX_TYPE sum[100])
{
    #pragma HLS interface m_axi port=A offset=slave bundle=mem1
    #pragma HLS interface m_axi port=B offset=slave bundle=mem2
    #pragma HLS interface m_axi port=sum offset=slave bundle=mem3

    FIX_TYPE A_local[100][100];

    L1: for(int j = 0; j < 100; j++) {
        for(int i = 0; i < 100; i++) {
            A_local[i][j] = A[i][j];
        }
    }

    L2: for(int i = 0; i < 100; i++) {
        for(int j = 0; j < 100; j++) {
            sum[i] += A_local[i][j] + B[i][j];
        }
    }

    fooA(A);
    fooB(B);
}
An Example for C/RTL Co-simulation

```c
void fooA(FIX_TYPE A[100][100])
{
    #pragma HLS inline off
    for(int j = 0; j < 100; j++) {
        for(int i = 0; i < 100; i++) {
            A[i][j] += 1;
        }
    }
}

void fooB(FIX_TYPE A[100][100])
{
    #pragma HLS inline off
    for(int j = 0; j < 100; j++) {
        for(int i = 0; i < 100; i+=2) {
            A[i][j] += 1;
        }
    }
}
```

- 10000 cycles
- 5000 cycles

- Estimations!
- Use C/RTL Co-sim to get (relatively) accurate values
C/RTL Co-sim Steps

Click this

Then click this...
C/RTL Co-sim Steps

This tells you it’s successful
C/RTL Co-sim Steps

• Remember that...
  o You need a testbench in C++ (like what we have in main.cpp)
  o No dynamic arrays in testbench either!
  o Make sure the INTERFACE pragma is correctly set (to simulate the memory behavior)

• Co-sim Strongly recommended!
  o HLS synthesis report is not reliable!!!!
How to Check Co-sim Results

- Open Vivado and load the project.
- Go to the Simulation Results tab.
- Select the test.wcfg file.
- Use the Tcl Console to execute scripts for analysis.

Example command:
```tcl
open_wave_config [-home]/path/to/open_wave_config.xml
open_wave -config
```
How to Check Co-sim Results

![Diagram of co-simulation scope and sources]

**Scope**

- **Name**
  - apatb_test_top
  - AESI_inst_test
    - control_s_axi_U
    - mem1_m_axi_U
    - mem2_m_axi_U
    - mem3_m_axi_U
    - A_local_V_U
    - grp_fooA_fu_300
    - grp_fooB_fu_307
    - mul_7ns_10ns_16_1_l_U8
    - mac_muladd_7ns_8ns_7ns
    - AESI_AXI_MASTER_mem1
    - AESI_AXI_MASTER_mem2
    - AESI_AXI_MASTER_mem3
    - AESI_AXI_SLAVE_control
    - U_dataflow_monitor
    - glbl

**Sources**

- Design T... Block T...
How to Check Co-sim Results

Scope x Sources

<table>
<thead>
<tr>
<th>Name</th>
<th>Design...</th>
<th>Block Ti...</th>
</tr>
</thead>
<tbody>
<tr>
<td>apabt_test_top</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_inst_test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>control_s_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem1_m_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem2_m_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem3_m_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_local_V_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>grp_fooA_fu_300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>grp_fooB_fu_307</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mul7ns_10ns_111111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_muladd_7ns_8ns_7ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_MASTER_mem1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_MASTER_mem2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_MASTER_mem3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_SLAVE_control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U_dataflow_monitor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gbl</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scope x Sources

<table>
<thead>
<tr>
<th>Name</th>
<th>Design...</th>
<th>Block Ti...</th>
</tr>
</thead>
<tbody>
<tr>
<td>apabt_test_top</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_inst_test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>control_s_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem1_m_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem2_m_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem3_m_axi_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_local_V_U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>grp_fooA_fu_300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>grp_fooB_fu_307</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mul7ns_10ns_111111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_muladd_7ns_8ns_7ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_MASTER_mem1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_MASTER_mem2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_MASTER_mem3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AESL_AXI_SLAVE_control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U_dataflow_monitor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gbl</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Objects x Protocol Instances

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Data Typ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap_clk</td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td>ap_rst</td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td>ap_start</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add to Wave Window
- Log to Wave Database
- Show in Wave Window
- Go to Source Code
- Go to Actual
- Radix
- Show as Enumeration
- Report Drivers
- Force Constant...
- Forge Clock...
- Remove Force...
- Default Radix

For more information, visit https://sharclab.ece.gatech.edu/
How to Check Co-sim Results

- **Signal “ap_start”** describes a function’s start/stop behavior
  - “High” when the function is running
  - “Low” when it’s idle
How to Check Co-sim Results

- Observe data/array access?
- “ce” signal or “address” signal
How to Check Co-sim Results

• Observe the waveform and latency values here
HLS Report v.s. C/RTL Co-sim

HLS Report

<table>
<thead>
<tr>
<th>Modules &amp; Loops</th>
<th>Issue Type</th>
<th>Slack</th>
<th>Latency(cycles)</th>
<th>Latency(ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td>-</td>
<td>31236</td>
<td>3.120E5</td>
<td></td>
</tr>
<tr>
<td>fooA</td>
<td>-</td>
<td>10017</td>
<td>1.000E5</td>
<td></td>
</tr>
<tr>
<td>fooB</td>
<td>-</td>
<td>5017</td>
<td>5.017E4</td>
<td></td>
</tr>
<tr>
<td>L1_VITIS_LOOP_420_1</td>
<td>-</td>
<td>10009</td>
<td>1.000E5</td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>-</td>
<td>11200</td>
<td>1.120E5</td>
<td></td>
</tr>
</tbody>
</table>

50,170 ns
5,017 cycles

Co-sim

<table>
<thead>
<tr>
<th>Modules</th>
<th>Avg II</th>
<th>Max II</th>
<th>Min II</th>
<th>Avg Latency</th>
<th>Max Latency</th>
<th>Min Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td>55725</td>
<td>55725</td>
<td>55725</td>
<td>55725</td>
<td>55725</td>
<td>55725</td>
</tr>
<tr>
<td>fooA</td>
<td>25027</td>
<td>25027</td>
<td>25027</td>
<td>25027</td>
<td>25027</td>
<td>25027</td>
</tr>
<tr>
<td>fooB</td>
<td>12529</td>
<td>12529</td>
<td>12529</td>
<td>12529</td>
<td>12529</td>
<td>12529</td>
</tr>
</tbody>
</table>

125,300 ns
12,530 cycles
HLS Report v.s. C/RTL Co-sim

More than one cycle to perform one memory access

Strongly Recommend Co-Sim!!!
Outline

• C/RTL Co-Simulation
• Efficient DSP usage for low-bitwidth computation
• Deep Learning with **INT8** Optimization on Xilinx Devices

• Convolutional Neural Network with **INT4** Optimization on Xilinx Devices
(Xilinx) DSP Structure

18-bit * 27-bit

(A+D)\times B+C

But not quite common in ML applications
18-bit * 27-bit: a large waste for INT8
The higher 10-bit or 19-bit inputs are filled with zeros or ones
Idea: pack two multiplications into one DSP!
Idea: Pack Two Values into One

\[ A \times C + B \times C = (A + B) \times C \]
Idea: Pack Two Values into One

Are there sharing opportunities like this (A*C, B*C) in ML workloads?

C must be commonly used by two
Sharing Opportunities in MLP

Sum of product terms: \( a_1 w_{1,j} + \ldots + a_j w_{i,j} + \ldots + a_n w_{n,j} + w_0 \)
Sharing Opportunities in MLP

Input/Last Layer

Output/Next Layer
Sharing Opportunities in MLP

\[ a_1 \cdot w_1 \]

\[ a_1 \cdot w_2 \]
Sharing Opportunities in Convolution

- **Input Feature Maps (IFMs)**: 384 x 13 x 13
- **256 Kernel Weights**
- **Out Feature Maps (OFMs)**: 13 x 13 x 256

**a. Weight Sharing:** Compute two OFM samples in parallel
Sharing Opportunities in Convolution

b. Input Sharing: Compute two OFMs in parallel
(Xilinx) INT4 Optimization
Xilinx v.s. Intel

Figure 8: INT8 Deep Learning Power Efficiency Comparison: Xilinx vs. Intel
Summary

- C/RTL Co-simulation
- Advanced DSP packing for low bit-width computation
Summary

• That’s all!! You’ve learned everything from me!
• Now it’s fun time!! Go for the adventure!!!
**Summary**

- But be prepared...

<table>
<thead>
<tr>
<th>What they teach in class:</th>
<th>What they teach in class</th>
<th>What the professor covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>What they ask in exam:</td>
<td>What they ask in exam</td>
<td>What's on the exam</td>
</tr>
<tr>
<td></td>
<td></td>
<td>What you remember</td>
</tr>
</tbody>
</table>