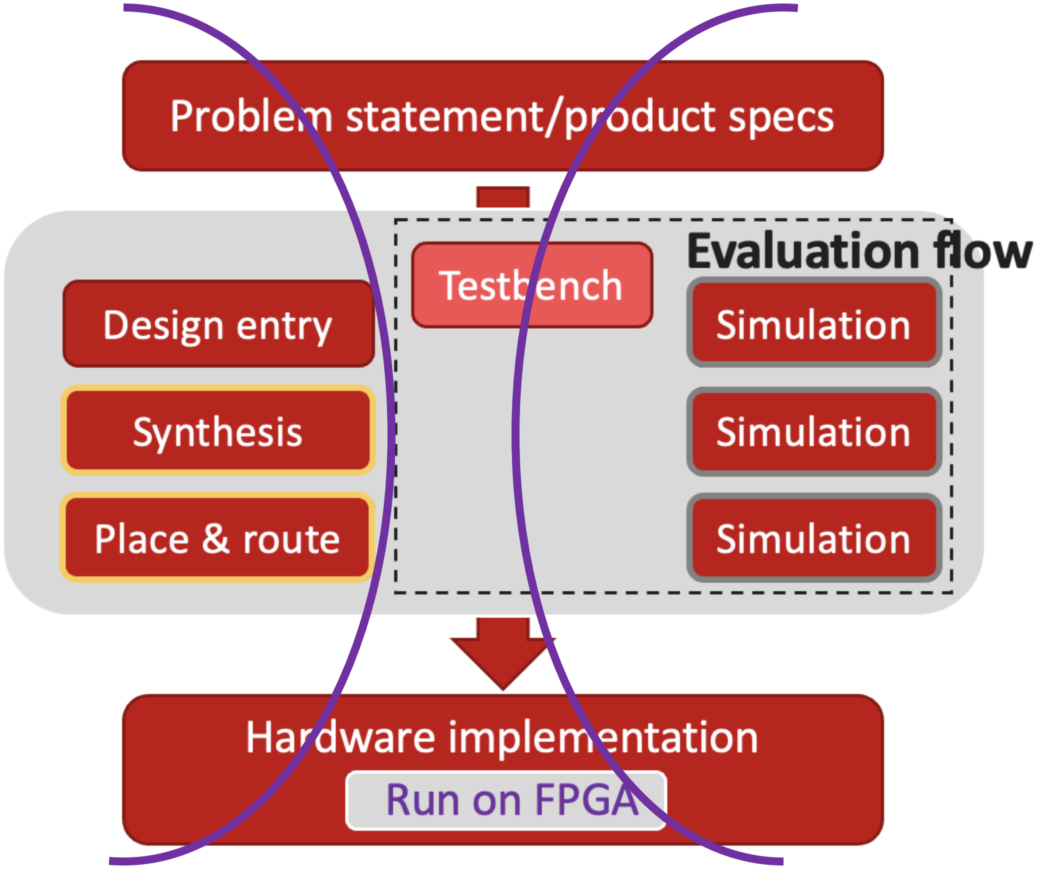
ECE 554: Minilab 1

Simulation & debugging

Minilab 0 - Minilab 1

Focus of Minilab 0



Focus of Minilab 0

Design validation

- Test and debug at several steps in the process
  - At architecture level - functional simulation of HDL.
  - At RTL level - functional simulation of RTL HDL.
  - At logic design or synthesis - functional simulation of gate-level circuit.
  - At implementation - timing simulation of schematic, netlist or HDL with implementation-based timing information (functional simulation can also be useful here).
  - At programmed FPGA level - in-circuit test of function and timing.

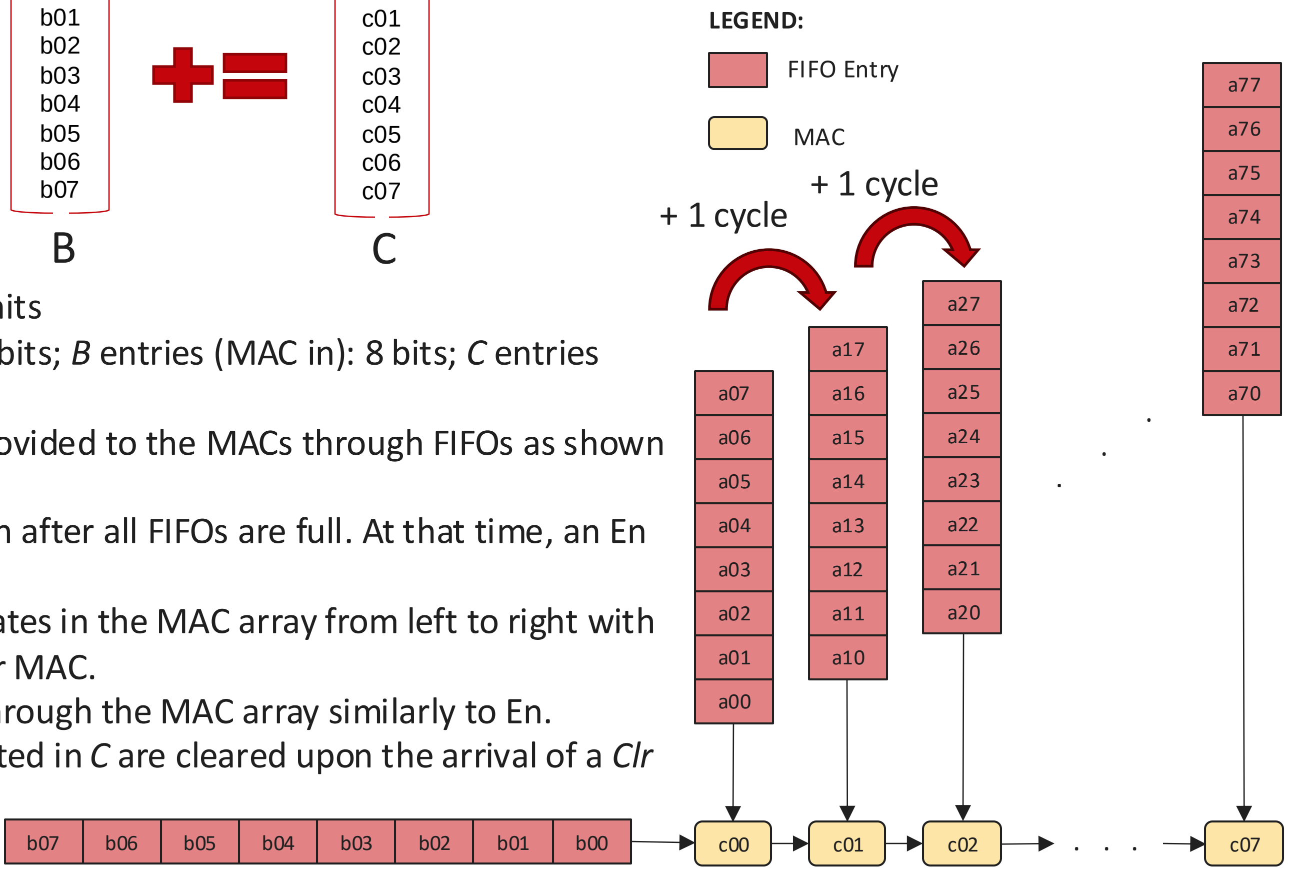
Minilab 1 Part 1 (RTL level, logic design or synthesis, implementation)

Minilab 1 Part 2 (programmed FPGA level)

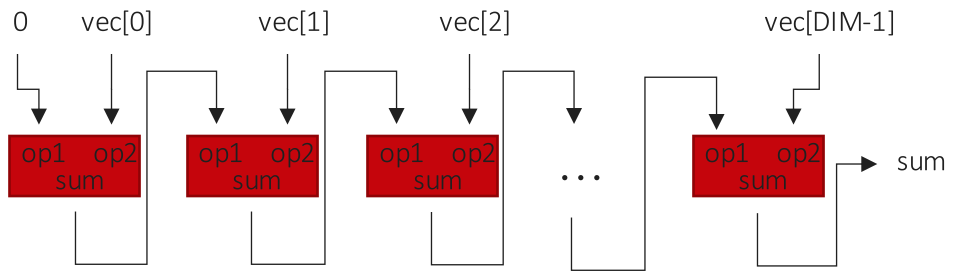
Example design: matrix-vector multiplication

$$\underset{A}{\begin{bmatrix} a00 & a01 & a02 & a03 & a04 & a05 & a06 & a07 \\ a10 & a11 & a12 & a13 & a14 & a15 & a16 & a17 \\ a20 & a21 & a22 & a23 & a24 & a25 & a26 & a27 \\ a30 & a31 & a32 & a33 & a34 & a35 & a36 & a37 \\ a40 & a41 & a42 & a43 & a44 & a45 & a46 & a47 \\ a50 & a51 & a52 & a53 & a54 & a55 & a56 & a57 \\ a60 & a61 & a62 & a63 & a64 & a65 & a66 & a67 \\ a70 & a71 & a72 & a73 & a74 & a75 & a76 & a77 \end{bmatrix}} \times \underset{B}{\begin{bmatrix} b00 \\ b01 \\ b02 \\ b03 \\ b04 \\ b05 \\ b06 \\ b07 \end{bmatrix}} += \underset{C}{\begin{bmatrix} c00 \\ c01 \\ c02 \\ c03 \\ c04 \\ c05 \\ c06 \\ c07 \end{bmatrix}}$$

- ❑ Create an array of 8 MAC units
  - ➢ *A* entries (MAC in): 8 bits; *B* entries (MAC in): 8 bits; *C* entries (MAC out): 24 bits.
  - ➢ Inputs A and B are provided to the MACs through FIFOs as shown in the diagram.
  - ➢ MAC operations begin after all FIFOs are full. At that time, an En signal gets triggered.
  - ➢ The En signal propagates in the MAC array from left to right with one cycle of delay per MAC.
  - ➢ B inputs propagate through the MAC array similarly to En.
  - ➢ The results accumulated in *C* are cleared upon the arrival of a *Clr* signal.


LEGEND:
🟥 FIFO Entry
🟨 MAC
+ 1 cycle
+ 1 cycle
a07 a06 a05 a04 a03 a02 a01 a00
a17 a16 a15 a14 a13 a12 a11 a10
a27 a26 a25 a24 a23 a22 a21 a20
a77 a76 a75 a74 a73 a72 a71 a70
b07 b06 b05 b04 b03 b02 b01 b00
c00
c01
c02
c07

Instantiating Regular Structures



```
module vector_reduce(input [15:0] vec [7:0],
                     output [15:0] sum);
logic [15:0] sums [8:0]; // note extra wiring
always_comb begin
  sums[0] = 0;
  for(int i=0; i<8; ++i) begin
    sums[i+1] = sums[i] + vec[i];
  end // for
end // always_comb
endmodule // vector_reduce
```

```
module vector_reduce(input [15:0] vec [7:0],
                     output [15:0] sum);
logic [15:0] sums [8:0]; // note extra wiring
assign sums[0] = 0;
assign sum = sums[8];
genvar i;
generate
  for (i=0; i < 8; ++i) begin
    myadder (.op1(vec[i]),
             .op2(sums[i]),
             .sum(sums[i+1]));
  end // for(i
endgenerate
endmodule // vector_reduce
```

Part 1

- ❑ Part 1 consists of 2 subparts.
  - ❑ Part 1a: Complete the design and simulate it with a testbench.
    - ❑ A memory module with a **.mif** file is provided along with the required IP files. The memory module has a partial implementation of INTEL Avalon MM slave interface. Apart from the matrix-vector multiplication module, design a module that fetches data from the provided memory module to fill the FIFOs. Look at Tables 1 & 2 and Figure 7 here for more details on Avalon MM Interface.
    - ❑ The testbench should print all interface signals, states, and output signals.
    - ❑ Use this command to simulate in QuestaSim.

    ```
    vsim work.<your_testbench_name>_tb -L
    C:/intelFPGA_lite/21.1/questa_fse/intel/verilog/altera_mf -voptargs="+acc"
    ```
  - ❑ Part 1b: Fix design to meet timing constraints.
    - ❑ Change the clock in the Synopsys Design Constraint (.sdc) file to 200 MHz by changing the clock period (in nanoseconds).
    - ❑ Synthesize the design again and see if it meets timing by looking at the reports from the **Timing Analyzer**. Here is a helpful video that shows how to do it.
    - ❑ Fix the design to meet the timing requirements of the 200 MHz clock.

Part 2

- Part 2 also consists of 2 subparts.
  - Part 2a:
    - Use LEDs to display and track the states of your top-level state-machine.
    - Use 7-segment display the outputs of the MACs. (Please)

    Note 1: Use switches to control the displays. The board has 10 switches; so, there are 2^10 different ways of control.

    Note 2: Minilab 0 codes may be handy.

  - Part 2b: Use SignalTap to ensure that you have implemented the Avalon MM interface correctly.

Demo

- ❑ Simulation of working design
- ❑ Timing analyzer report
- ❑ On-board testing with LEDs, 7-segment display, and switches.
- ❑ On-board testing with SignalTap

Deliverables

- ❑ Create a new public GitHub repository for your minilab submission (1 per team.
- ❑ Push all relevant files to the repository, including your codes, testbenches, and simulation logs. If for the implementation IPs are used, also add all IP-related files.

NOTE: Simulation log with proper print statements from the testbench that explains whether a test passed or failed. If the design failed a test, it should print where and why it failed.

- ❑ Minilab 1 report is due on 2/4 before the start of Minilab 2.
- ❑ In your report (1 per team), explain:
  - ➢ How you tested the design via simulation (screenshot of waveform).
  - ➢ How you fixed the design to meet the timing constraint.
  - ➢ How you tested the design on board using an example and snapshot of the board that shows it.
  - ➢ How you tested the design using SignalTap (screenshot of waveform around the trigger condition).
  - ➢ Explain any difficulties that you faced during the whole process and how you overcame it.

Questions?