1 Introduction

In this lab, we will extend our IIR design done in the previous lab. We will implement unfolded and folded IIR filter designs and compare their differences.

For this lab, you are expected to complete the following tasks:

a) Implement the unfolded IIR filter design, verify the design by behavioral simulation. (Pre-lab)

b) Load the unfolded IIR design on the FPGA, run the testing program and verify the outputs.

c) Implement the folded IIR filter design, verify the design by behavioral simulation.

d) Infer the differences in the implementation when unfolding and folding are included in the design.

1.1 Introduction to Unfolding and Folding

Unfolding is a transformation technique that increases the throughput of a digital signal processing program by duplicating the functional blocks while maintaining functionality. On the other hand, **folding** converts multiple operations into a single operation block and minimizes the number of functional blocks needed in synthesizing the digital signal processing system.

Please refer to the lectures (or chapters 5 and 6 from the text authored by Keshab Parhi prescribed in the syllabus) to understand the unfolding and folding algorithms before the lab.

We need to design IIR for this lab based on the same IIR design. Figure 1, Figure 2, and Figure 3 show the original, unfolded, and folded design of the IIR filter, respectively.

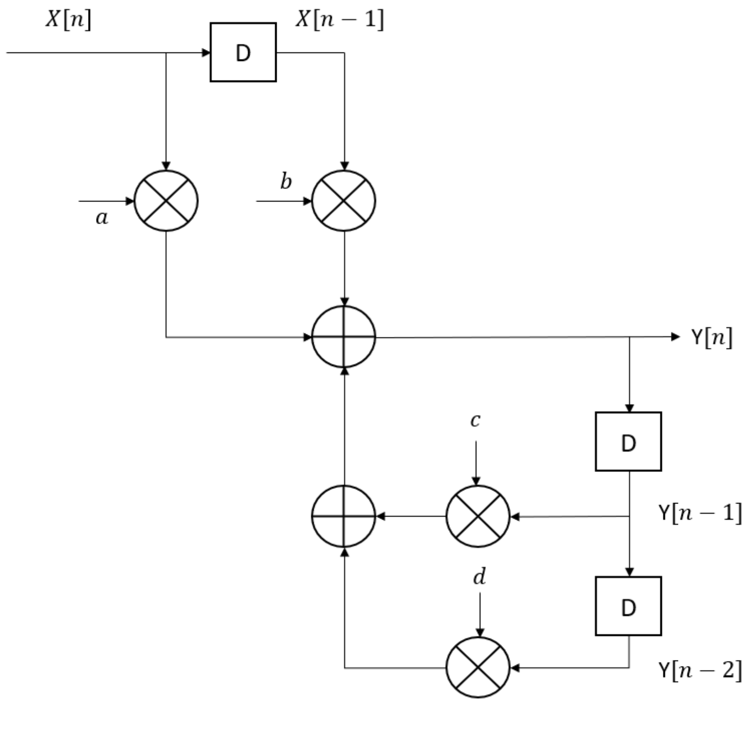


Figure 1: IIR Original Design

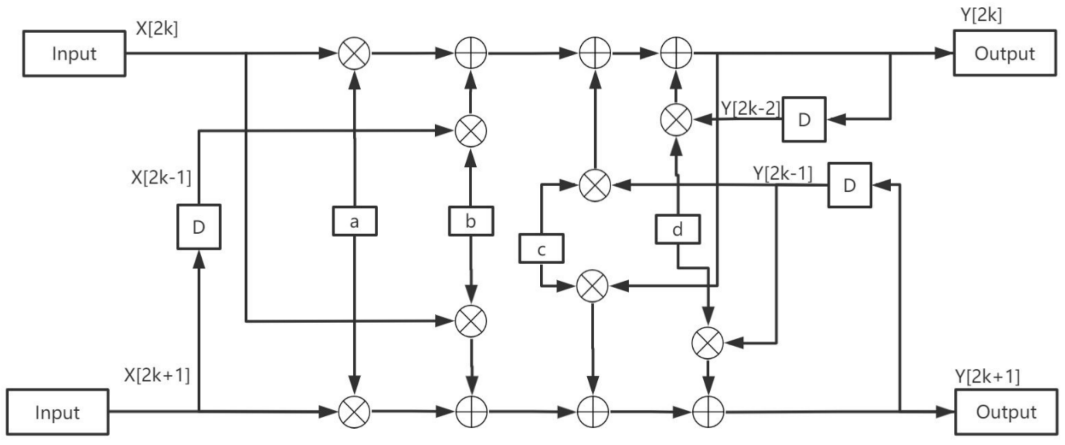


Figure 2: IIR Unfolded Design

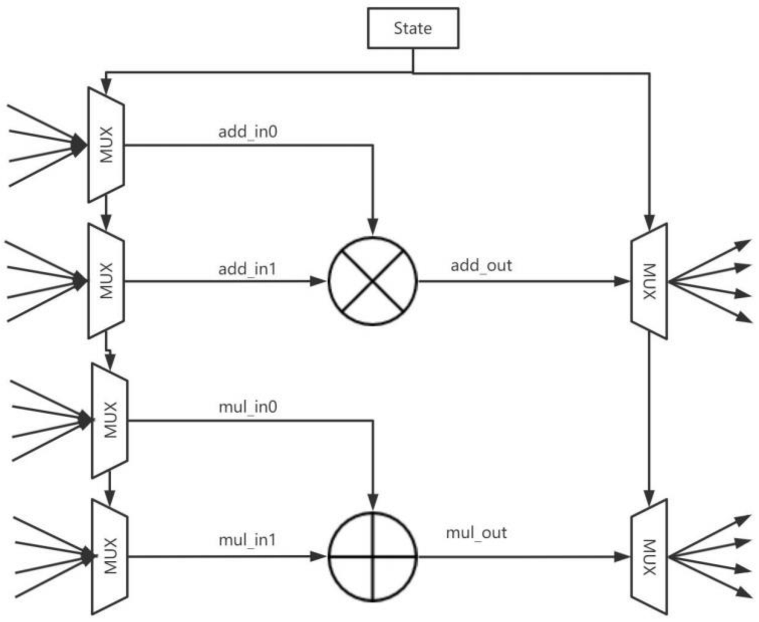


Figure 3: IIR Folded Design

2 Lab Design of the 2-Unfolded IIR

Please download "**base_vivado.zip**", and "lab3_code.zip" from Piazza and extract the files before you proceed. In the extracted folder, you will find the Vivado project file "base/base.xpr" and the source code in "/lab3_code". Please save a copy of this extracted folder, since you will need it in your future labs.
Now we start from adding the source files into the project.

a) Launch Vivado, open the project file **base.xpr**.

b) Click on "**File -> Add Sources..**" to add source files to the project as shown in Figure 4.

c) In the prompted window, select "**Add or create design sources**", click on "**Next**" as in Figure 5.

d) Click on "**Add Files**", select "**top_IIR_unfold.v**", "**IIR_unfold.v**" and "**multiply.v**", and click on "**Open**".

e) Click on “Finish” to add the previous files into the project as in Figure 6.

f) In the same manner, we add the test bench file “IIR_unfold_tb.v” to the project. Note: Please select “**Add or create simulation sources**” for the test bench file. Once finished, the sources panel should look like Figure 7.

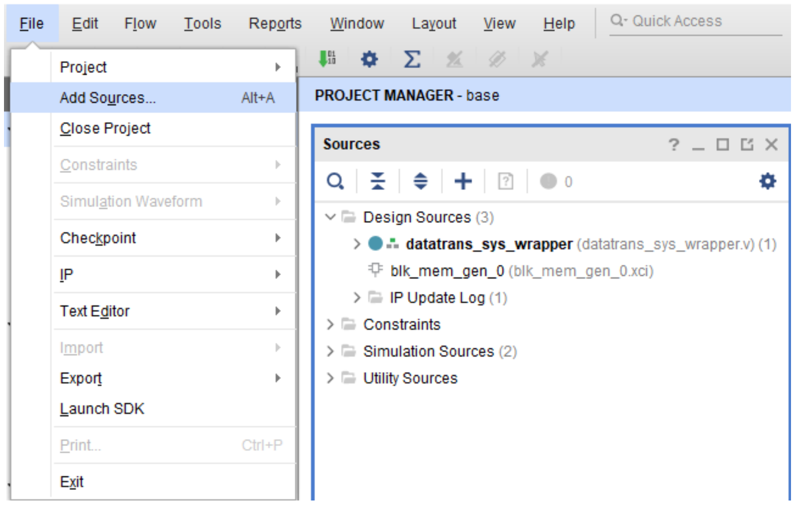
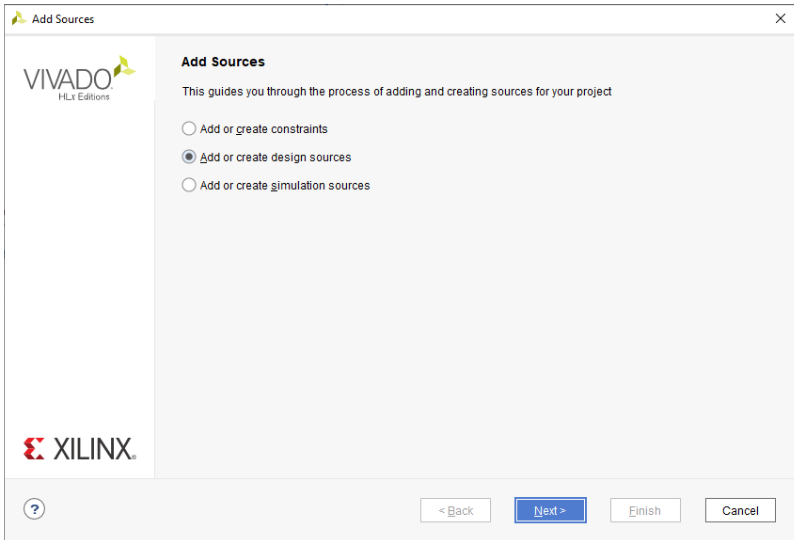


Figure 4: Adding Sources – 1

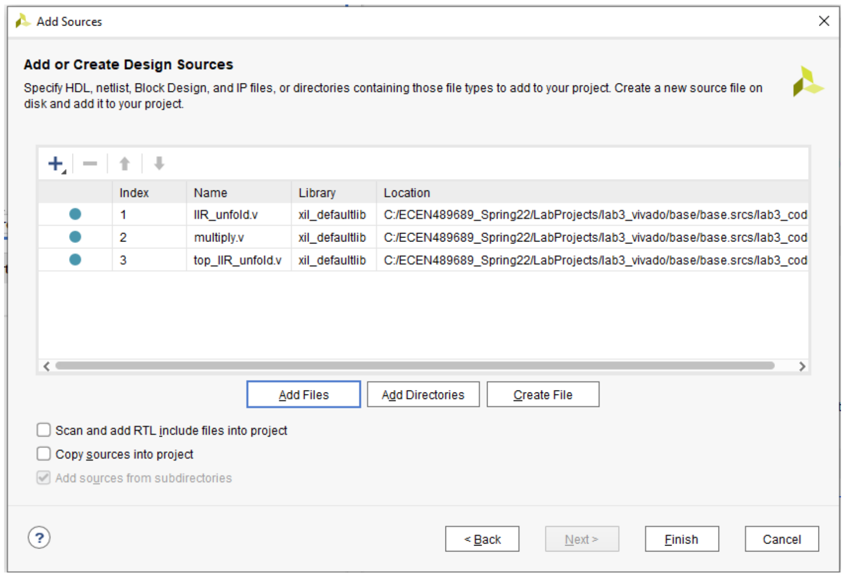


Add Sources

This guides you through the process of adding and creating sources for your project

Add or create constraints
[✓ Add or create design sources]
Add or create simulation sources

Figure 5: Adding Sources - 2


Add Sources

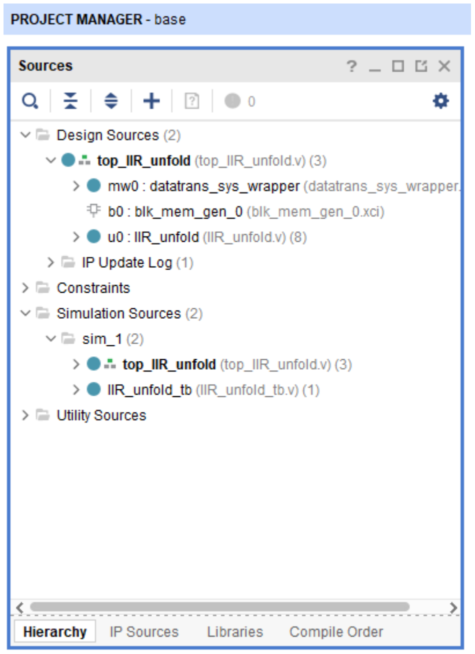
Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

	Index	Name	Library	Location
●	1	IIR_unfold.v	xil_defaultlib	C:/ECEN489689_Spring22/LabProjects/lab3_vivado/base/base.srcs/lab3_cod
●	2	multiply.v	xil_defaultlib	C:/ECEN489689_Spring22/LabProjects/lab3_vivado/base/base.srcs/lab3_cod
●	3	top_IIR_unfold.v	xil_defaultlib	C:/ECEN489689_Spring22/LabProjects/lab3_vivado/base/base.srcs/lab3_cod

Scan and add RTL include files into project
Copy sources into project
[✓] Add sources from subdirectories

Figure 6: Adding Sources – 3


PROJECT MANAGER - base

Sources

```
Design Sources (2)
  top_IIR_unfold (top_IIR_unfold.v) (3)
    mw0 : datatrans_sys_wrapper (datatrans_sys_wrapper.
    b0 : blk_mem_gen_0 (blk_mem_gen_0.xci)
    u0 : IIR_unfold (IIR_unfold.v) (8)
  IP Update Log (1)
Constraints
Simulation Sources (2)
  sim_1 (2)
    top_IIR_unfold (top_IIR_unfold.v) (3)
    IIR_unfold_tb (IIR_unfold_tb.v) (1)
Utility Sources
```

Figure 7: Sources View

g) Now it’s your job to write the 2-unfolded IIR module according to the unfolding algorithm taught in class and Figure 2. You will use 8-bit signed fixed-point numbers for calculation in the lab, with 4 bits for the fractional part.

After you finish coding “**IIR_unfold.v**”, you need to run the testbench for simulation. Please make sure to check the “**Simulation Settings**”, and set the simulation top module name as “**IIR_unfold_tb**”.

Now you can see the waveform of the simulation. Change the radix of the waveform to “Fixed point” with 4 fractional bits. If your design and testbench are correct, you should be able to see the filtered

waveform on the output ports of your IIR design.
Take a screenshot on the waveform and include it in your pre-lab report with your explanation on the results.

3 Implementation of Unfolded IIR on FPGA

Next, we need to implement the design on FPGA.

a) Add "**top_IIR_unfold.v**" into your project, right click this file in the "**Sources**" panel and click "*set as top*" (If this file is already in bold font, it is already the top module).

b) We need a block memory for the design to run on the FPGA. This is already included in the project. However, we need to do some configurations.

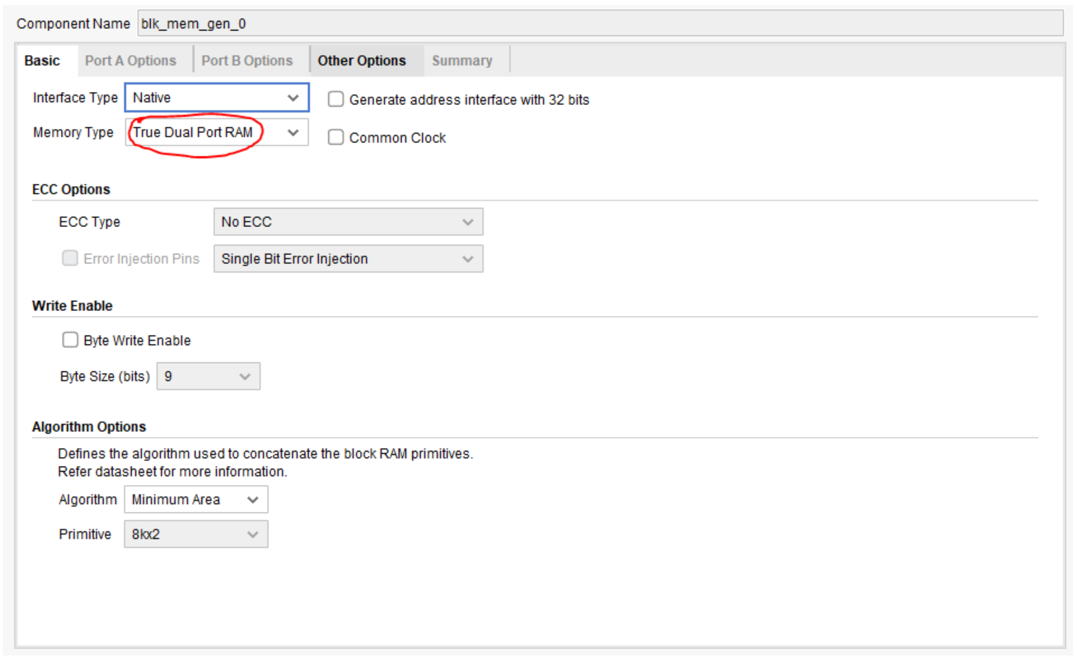
c) Double click on "blk_mem_gen_0" to customized the "block memory generator".

d) In this lab, we need a two ports ram for "**top_IIR_unfold.v**". For memory type, select "**True Dual Port Ram**" as in Figure 8

e) In both port A and port B options, change write width to 16, and write depth to 65536 as in Figure 9

f) Choose operation mode "**Read First**", enable port type **"Always Enabled"** as in Figure 9. Click "**OK**" to finish the customization.

g) In the pop-up window, select "**Global**" in the synthesis option, and click on "**Generate**".



Component Name blk_mem_gen_0

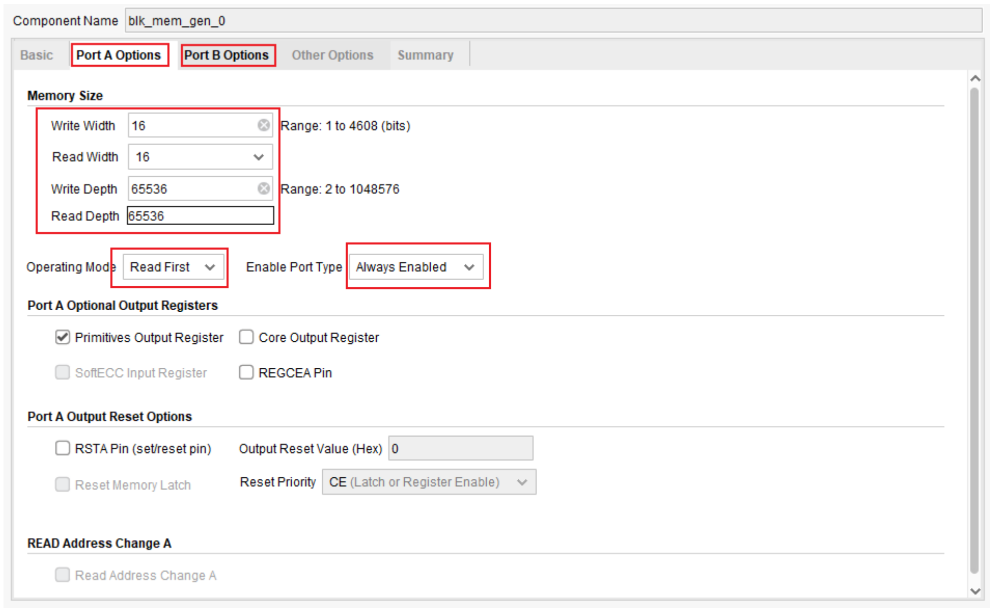
Interface Type: Native
Memory Type: True Dual Port RAM
☐ Generate address interface with 32 bits
☐ Common Clock

ECC Options
ECC Type: No ECC
☐ Error Injection Pins: Single Bit Error Injection

Write Enable
☐ Byte Write Enable
Byte Size (bits): 9

Algorithm Options
Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.
Algorithm: Minimum Area
Primitive: 8kx2

Figure 8: Customizing Block Memory Generator - 1


Component Name: blk_mem_gen_0

Memory Size
- Write Width: 16 — Range: 1 to 4608 (bits)
- Read Width: 16
- Write Depth: 65536 — Range: 2 to 1048576
- Read Depth: 65536

Operating Mode: Read First — Enable Port Type: Always Enabled

Port A Optional Output Registers
- [✓] Primitives Output Register
- [ ] Core Output Register
- [ ] SoftECC Input Register
- [ ] REGCEA Pin

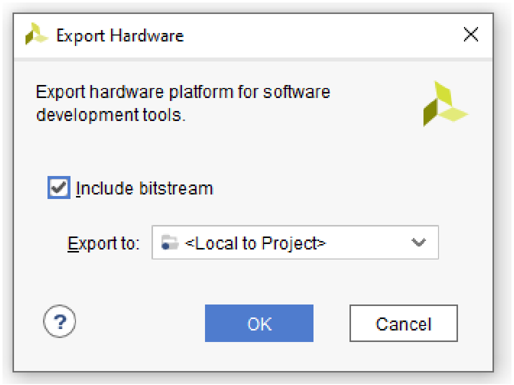
Port A Output Reset Options
- [ ] RSTA Pin (set/reset pin) — Output Reset Value (Hex): 0
- [ ] Reset Memory Latch — Reset Priority: CE (Latch or Register Enable)

READ Address Change A
- [ ] Read Address Change A

Figure 9: Customizing Block Memory Generator - 2

h) After the configuration is finished, we can run the implementation of the design. Click on "Run Implementation", and wait for several minutes until Vivado finishes.

i) If no error occurs, click on **generate bitstream**. After the bitstream is generated, click on "**File -> Export -> Export Hardware**". Check "**Include Bitstream"**, click on "**OK**" as in Figure 10.


Export Hardware

Export hardware platform for software development tools.

[✓] Include bitstream

Export to: <Local to Project>

Figure 10: Exporting Hardware

Now, we use can implement the design on the Zybo board.

Refer to "**HowToImplementDesignOnFPGA.pdf**" for the details.

Once you boot the Linux system on the Zybo board, run the testing program with the command:

./lab3_IIR_unfold_test

The results are written into **lab3_IIR_unfold_results**. (Check it by "ls")

Please screenshot the output in the terminal, and attach it in the post-lab report.

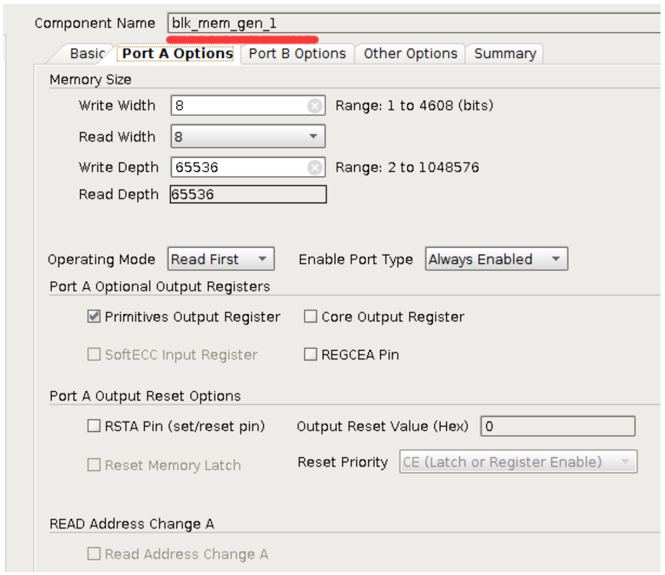
In your computer, plot the data in **lab3_IIR_unfold_results** using your favorate tool, and **attach it in the post-lab report**.

4 Lab Design on Folded IIR (For graduate students)

a) Repeat Section 2 and 3 to design a 4-folded IIR, with files: "**IIR_fold.v**", "**IIR_fold_tb.v**". Run the behavior simulation of the folded IIR design, and include it in the post-lab report.

b) Bonus: If you are interested, you can try implementing the folded IIR design on the FPGA with top_IIR_fold.v and lab3_IIR_fold_test. The output from the implementation is lab3_IIR_fold_results.

c) For the implementation on the board, please add the RAM IP with write width 8, and name **blk_mem_gen_1**, as shown in Figure 11: RAM IP.



Component Name: blk_mem_gen_1

Memory Size
Write Width: 8 Range: 1 to 4608 (bits)
Read Width: 8
Write Depth: 65536 Range: 2 to 1048576
Read Depth: 65536

Operating Mode: Read First Enable Port Type: Always Enabled

Port A Optional Output Registers
☑ Primitives Output Register ☐ Core Output Register
☐ SoftECC Input Register ☐ REGCEA Pin

Port A Output Reset Options
☐ RSTA Pin (set/reset pin) Output Reset Value (Hex): 0
☐ Reset Memory Latch Reset Priority: CE (Latch or Register Enable)

READ Address Change A
☐ Read Address Change A

Figure 11: RAM IP

d) Note that the 3-input adder in Figure 1 should be replaced by two 2-input adders, as in Figure 2.

e) You are required to have the following folding set for the multipliers: {m1, m2, m4, m3}, and for the adders: {a3, a2, a1,Ø}. The folding orders of input and output are both 0 (you can realize this by regarding the input and output ports as components and adding two folding sets when performing folding: {input,Ø ,Ø ,Ø}, {output,Ø ,Ø ,Ø}).

f) You may need to do retiming for these folding sets. Please refer to the lectures for the detail of retiming.

5 Pre-lab Submission

1. Please only submit 1 PDF file, containing the following items:
    a. Copy your code in "unfolded_IIR.v"
    b. Screenshots of the behavioral simulation of unfolded IIR design.
    c. A few workds explaining the results in the screenshots.
    d. Answers to the following question:
       What are the pros and cons of unfolding and folding, compared with the original design?
2. Please name the PDF file as "Lab#_PreLab_Section#_LastName_FirstName.pdf"
3. Please submit the PDF file on Canvas before February 14 (Monday) 11:59 pm.

6 Post-lab Submission

1. Please only submit 1 zip file, containing all the code files you used in this lab, and 1 PDF report. Name it as "Lab#_Post_Lab_Section#_LastName_FirstName.zip"
2. In this PDF report, please include the following items:
   a. Screenshots of the terminal outputs of unfolded IIR design.
   b. Plot of the data in the unfolded IIR output file.
   c. A few workds explaining the results in the screenshots.
   d. Screenshots of the behavioral simulation of the folded IIR design. (Graduate students only)
3. Please submit the zip file on Canvas before February 18 (Friday) 11:59 pm.