EEL - 5721 Reconfigurable Computing (Fall 2021)

Instructor for the course : **Dr. Greg Stitts**

Final Project Report

Topic : **1-D Time-Domain Convolution**

Date of Submission : 12/15/2021

Group 45 :
Student name : Mohit P Sathyaseelan
UFID : 39847026
Department of Electrical and Computer Engineering
mohit.palliyilsa@ufl.edu

Project status : Completed USER_APP, tested on hardware and test bench.

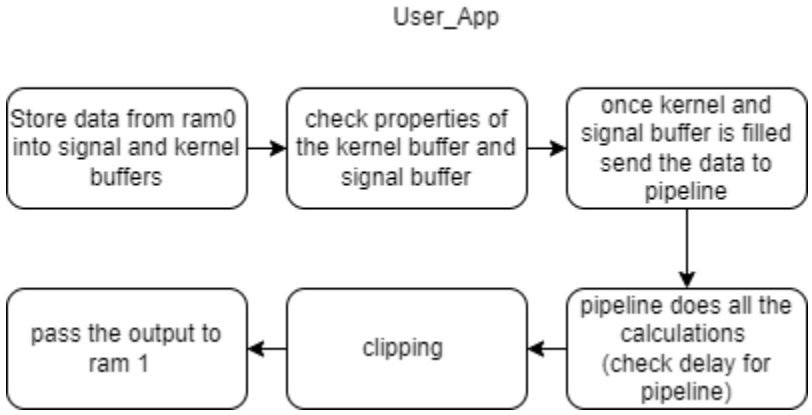
Project detail :

1. **Introduction :**

   This project is about how a common DSP operation “convolution” is carried out on Zedboard. We are using the Zync 7 ZC702 board for this project, also we will be improvinging the performance with parallelism; as compared to microprocessor. Here in Convolution we will be considering two different inputs, in our case signal and kernel array. We will take multiplication of each element from the x array and h array, for all the elements in the array. In the next step we have to add all the multiplication outputs and the addition pipeline continues until we get a single output.

2. **Project Flow chart:**

   We will only focus on the UserApp entity of the project as I’m working alone for the project. The UserApp has Signal buffer, kernel buffer, pipeline, controller, delays, and clipping logic as main entities. The Ram0 provides the signal buffer 16 bit input every cycle and similarly the kernel buffer attains the 16 bit data from memory map. Both the buffers work similarly although the Kernel Buffer, once filled with 128 elements it won't change until done signal is received. The outputs from these buffers are 128 elements of 16 bit width size, which goes to the pipeline. The pipeline takes these inputs and provides a 39 bit output. Further this 39 bit is clipped and sent to the dma interface for the ram1 (i.e output ram).



   a) Signal buffer:

Input for signal buffers is 16 bit data, once write enable is given to this buffer it starts storing these data into an array of registers. The array is of 128 depth and width of 16 bit, and acts like a fifo. Once the array is filled with 128 elements, the buffer stops taking in data and the full flag is asserted. The empty flag for the fifo is always high when there is space for any 16 bit element. Once the fifo is filled with shift operation. One important thing to note about Signal buffer is that it reverses the order of data, this is not the same with kernel buffer. The counters are the main logic of this buffer, so once it writes data the counter increases and alternatively reduces when read is enabled. The output is then passed onto the pipeline when it requests data.

b) Kernel Buffer:
Is similar to Signal buffer, although it takes its input from memory map. The full logic and empty logic remain the same as the signal buffer. Although here once the buffer is filled it won't take more inputs which means data won't shift when read enable is asserted. This is prevented by not decreasing the count when read enable is asserted. The buffer records data in an ascending order.

c) Pipeline :
Our pipeline here does most of the job, i.e. convolution calculation. If we go back to our assignment four, we figured out how to find if the data is valid or not using delay. Here in our pipeline once the data is received from our buffers the valid input bit is asserted. Calculating the delay matters here since, we only require the accurate output of convolution. The latency of the entity is clog2(128) = 7, this is added to the total delay of the mult_add pipeline. The first layer of the mult_add pipeline does multiplication and the rest layers is just addition, which sums to 128. So the valid out of the pipeline is 135(128+7). The pipeline provides us with an output of 39 bits.

d) Clipping :
Once our data is received from the pipeline we will clipp the data to 16 bits. If the output is greater than 65535 which is a decimal representation of 16 bits of one, we set the output as all ones. If not the output will be lower 16 bits.

3. **Project Steps:**

Identifying the enable and logic of different entities :

Signal buffer:	
Write enable	when ram0 sends valid
Read enable	when buffer is not empty and ram1wr is ready to take in input
Full	when count is = 128
Empty	when count is < 128
Kernel buffer	
Write enable	when kernel is ready
Read enable	when ram1wr is ready and kernel buffer is full
Full	when count = 128
Empty	when count < 128
Pipeline	
Enable	when ram1wr is ready
Pipeline valid	when data is valid from kernel buffer,signal buffer and when ram1wr is ready.
Input1	signal buffer output
Input2	kernel buffer output

The next step was creating the buffers, I was initially confused on how this fifo works and realized that it is not a normal fifo. Thanks to Prof. Greg Stitt's, I was able to clear

this out. Kernel buffer was comparatively easier since the only thing that changes is the count for read.

Initially I forgot its a sliding/shift buffer, this was conveyed from the tb i created for the buffers. Finally the buffers were working perfectly.

Next I added delay to the pipeline. I've used Prof. Greg Stitt's delay entity which I called inside my pipeline. The cycles for the delay were set to 135 and similarly the rest of the inputs were mapped.

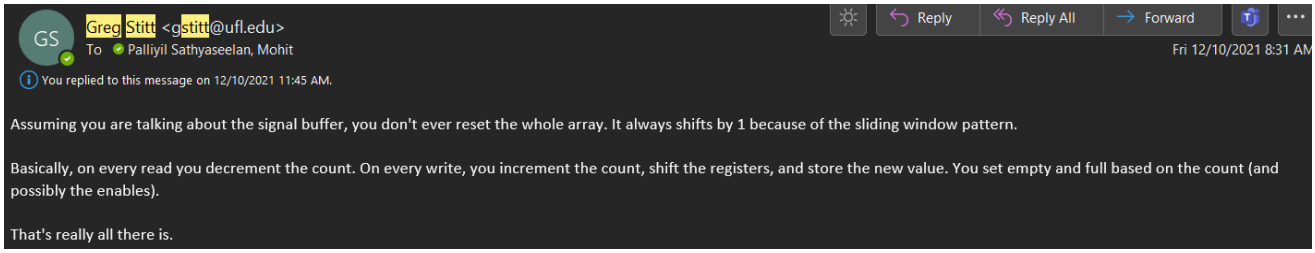
Finally I created the clipping logic in the user_app where it takes the pipeline output and checks for the condition. This output was mapped to ram1_wr_data.

For testing in the wrapper_tb, I had to first change some parts of the memory map as professor Stitt mentioned in class. The corresponding changes were made in the tb as well.

4. Analysis of code and problems faced:

Firstly i faced error for using vectorize function,
"Unable to open file '' because file does not have read permission. Would you like to open the directory instead? "
Although this was fixed, since I was mixing up the width and depth inputs for the function.

Then i faced issue with my buffers, for which i took a better understanding for Prof. Stits.



Re: RC final project

GS Greg Stitt <gstitt@ufl.edu>
To Palliyil Sathyaseelan, Mohit

Reply | Reply All | Forward

Fri 12/10/2021 11:51 AM

You replied to this message on 12/10/2021 2:37 PM.

The registers don't shift on a read, only on a write. The buffer should not replace the newest element. It will always shift out the data from the last element, which is also the oldest. The buffering slides illustrate this with registers.

The count logic is just:

```
if (rd_en)
   count = count - 1;

if (wr_en)
   count = count + 1;
```

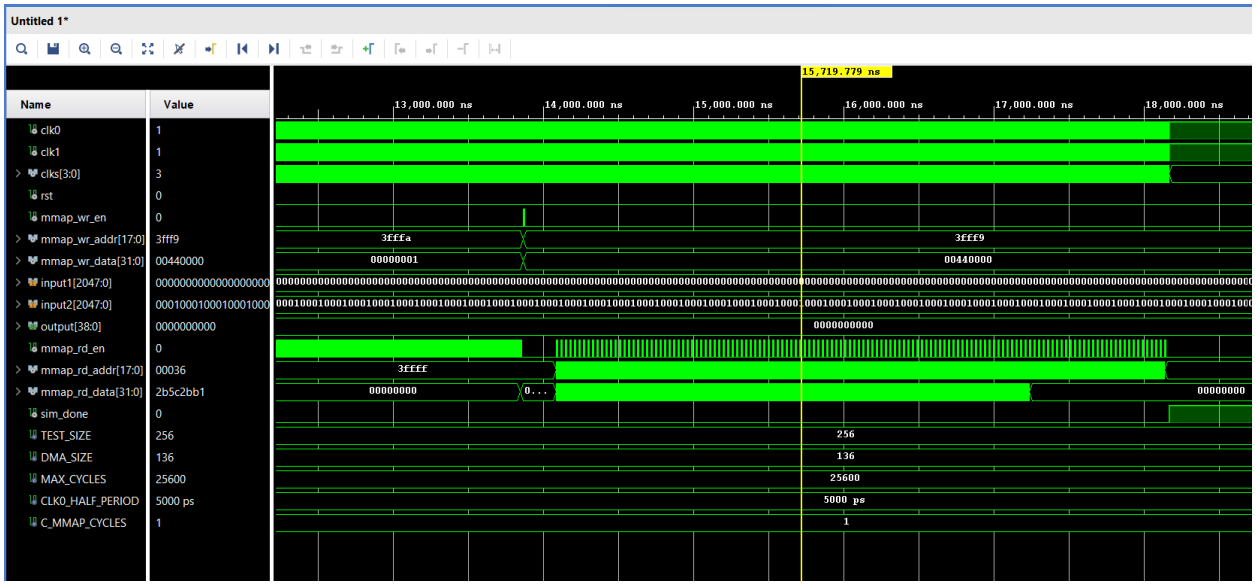
That has to be converted to VHDL, but it's that simple. I would also add some safety checks, so:

```
if (rd_en and not empty)
   count = count - 1

if (wr_en and not full)
   count = count + 1
```

After this my buffers were working perfectly fine on my TestBench. I put everything into the wrapper and tested with the wrapper tb. The add_tree had its own delay which i replaced with the delay used in the project after consultation with Prof. Stitts.

Then in the tb I was getting output from mmap_rd, for which I thought the code is working now.




```
Searching for available board....
Starting job "./zed_app design_1_wrapper.bit" on board 192.168.1.106:
Programming FPGA....SUCCESS
Testing transfers to/from address 0....(Size = 1)Error: Done was not asserted be
fore timeout.
terminate called after throwing an instance of 'TimeoutException'
  what():  Timeout Exception
sh: line 1:  1643 Aborted                 ./zed_app design_1_wrapper.bit
[mohit.palliyilsa@ece-b312-recon final]$
```

But this was the output from the hardware.

I consulted Prof. Stitts again regarding this, since I was lost.

Re: final project

GS

Greg Stitt <gstitt@ufl.edu>
To Palliyil Sathyaseelan, Mohit

Reply | Reply All | Forward

Mon 12/13/2021 7:36 PM

One possibility is that the individual test isn't the problem. The problem might arise from executing multiple tests in a row. If there is data left over in the FIFO, for example, that could mess up a subsequent test. I'm not sure how it would cause done to never be asserted, but it's a place to start.

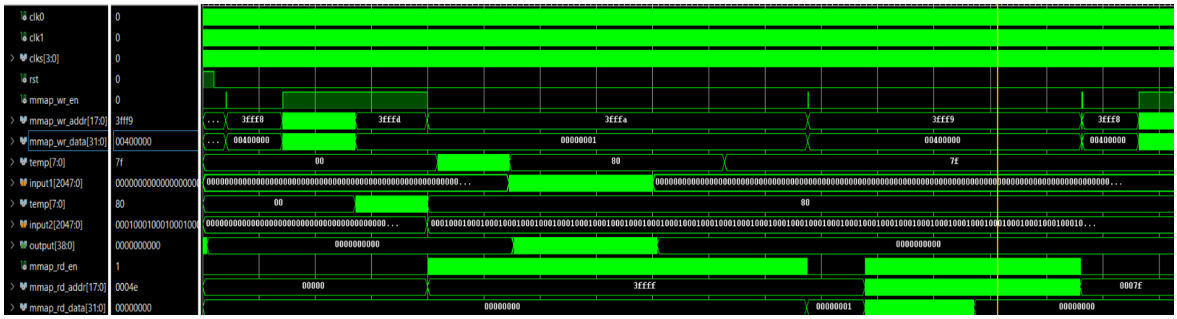
I would take the failing test and move it to the first test in the C++ code. If it passes there, then the issue is with repeated tests, which should be recreated in the testbench.

On 12/13/2021 6:27 PM, Palliyil Sathyaseelan, Mohit wrote:

> Hello prof. Stitts,
> I have tried making changes to the tb and I have tried for different test cases as well. The tb shows done being asserted but when I run the bit file on hardware I'm still getting run time exception. Could you give some tips on how to solve this ?
>
> Regards,
> Mohit P Sathyaseelan

So I tried analyzing the tb further down. I realized the issue has to be with my buffers because the rest of the program is working perfectly fine. I checked for latches etc but I was clueless.

From reference of this question :
https://stackoverflow.com/questions/27129747/trying-to-use-a-buffer-in-vhdl-not-working
I modified my Signal buffer and Kernel buffer using multiple processes. I cross checked all the parameters (i.e width and depth of the programs). Then I used the counter mentioned in the site and fixed this issue.



```
Error for output 49: HW = 0, SW = 65535
Error for output 50: HW = 0, SW = 65535
Error for output 51: HW = 0, SW = 65535
Error for output 52: HW = 0, SW = 65535
Error for output 53: HW = 0, SW = 65535
Error for output 54: HW = 0, SW = 65535
Error for output 55: HW = 0, SW = 65535
Error for output 56: HW = 0, SW = 65535
Error for output 57: HW = 0, SW = 65535
Error for output 58: HW = 0, SW = 65535
Error for output 59: HW = 0, SW = 65535
Error for output 60: HW = 0, SW = 65535
Error for output 61: HW = 0, SW = 65535
Error for output 62: HW = 0, SW = 65535
Error for output 63: HW = 0, SW = 65535
Error for output 64: HW = 0, SW = 65535
Error for output 65: HW = 0, SW = 65535
Error for output 66: HW = 0, SW = 65535
Error for output 67: HW = 0, SW = 65535
Error for output 68: HW = 0, SW = 65535
Error for output 69: HW = 0, SW = 65535
Error for output 70: HW = 0, SW = 65535
Error for output 71: HW = 0, SW = 65535
Error for output 72: HW = 0, SW = 65535
Error for output 73: HW = 0, SW = 65535
Error for output 74: HW = 0, SW = 65535
Error for output 75: HW = 0, SW = 65535
Error for output 76: HW = 0, SW = 65535
Error for output 77: HW = 0, SW = 65535
Error for output 78: HW = 0, SW = 65535
Error for output 79: HW = 0, SW = 65535
Error for output 80: HW = 0, SW = 65535
Error for output 81: HW = 0, SW = 65535
Error for output 82: HW = 0, SW = 65535
Error for output 83: HW = 0, SW = 65535
Error for output 84: HW = 0, SW = 65535
Error for output 85: HW = 0, SW = 65535
Error for output 86: HW = 0, SW = 65535
Error for output 87: HW = 0, SW = 65535
Percent correct = 91.5303
Speedup = 2.784

Testing big signal/kernel with random values...
Percent correct = 100
Speedup = 14.5061

TOTAL SCORE = 38.7622 out of 100
[mohit.palliyilsa@ece-b312-recon final_conv]$
```

Although now my output from hardware was as shown below.
I checked the comments in teams and realized that there were others facing the same issue. One of the comments said check the kernel and signal buffer logic. I actually copied the signal buffer code and forgot to change the method of asserting data to normal in the kernel buffer. Once I reversed back the logic, the whole model worked fine and gave me the output as predicted.

5. Final Result :