Final Project

Zack Mendez and Michael Tung

Hey, Dr. Stitt and Wes, our project works entirely! It was a fun experience; we love digital design/FPGA development, so this whole course was really educational and fun. Here is a short summary followed by screenshots and stuff proving our design. If you need any more information, please let us know.

Summary:

Dram_Read (Zack)

Man, I am so glad I did not have any metastability issues when creating this design. I just used the handshake that Michael and I submitted for lab 5, and it works like a charm, apparently. The main issues I had were with timing, the insanely dramatic address generator, and the FIFO reset signals.

If you look at the schematic in more detail, it is easier to see what I did. In general, I created a sort of edge-triggered latch that takes the go signal from the software and cuts it off immediately after one cycle. Go being asserted for a long period of time because software is slow would have caused issues, so I added this component so that the dram_read entity has “go” asserted for only one clock cycle at a time. This "innovation" led to the easier implementation of the counter-go and reset signals, the FIFO "reset" logic, and the address generator. I also handed this component to my partner to help with his reset logic for the kernel buffer, and it worked really nicely!

The FIFO reset was annoying, but my solution seems sort of elegant. I "reset" the FIFO when “go” is asserted by draining the FIFO's last output, only if the previous computation used an odd signal size. If the previous computation uses an even signal size, then the FIFO never gets "reset" as the FIFO gets fully drained on each run. Thanks for the announcement as it helped me figure this out. The Vivado FIFO being both full and empty was really odd.

Convolution (Michael)

During the process of debugging why all the hardware outputs were 0, I tried numerous approaches, including hardcoding values into the buffers and modifying the testbench to better match how the design operates on the board. Initially, I suspected an issue with the RAM control signals, as 0 typically indicates that we were not writing to the RAM. However, despite extensive efforts, I could not locate the error.

After many hours of debugging, my teammate and I finally recreated the issue on board. By inspecting the waveform, we discovered that the problem originated from not resetting the buffers. This caused the buffers to remain stuck at the first written data, which were all 0s, ultimately leading to the entire design failing. Interestingly, the design worked when we tested one case at a time. I guess, in other words, we created a one-shot convolution design.

Once we identified the root cause, we rewrote the reset logic for the buffers, and the design now works back-to-back in simulation and on the server.

Overall design

We followed the videos closely and watched them around 20 times to see if we missed anything for the overall design. There are many things that videos did not mention, and we have to figure out. We tested the dram_read and convolution pipelines separately. In the end, we combine them together and create a successful design.

Part 1: dram_read

Here's the schematic of our dram_read, fully working with no metastability or timing issues:

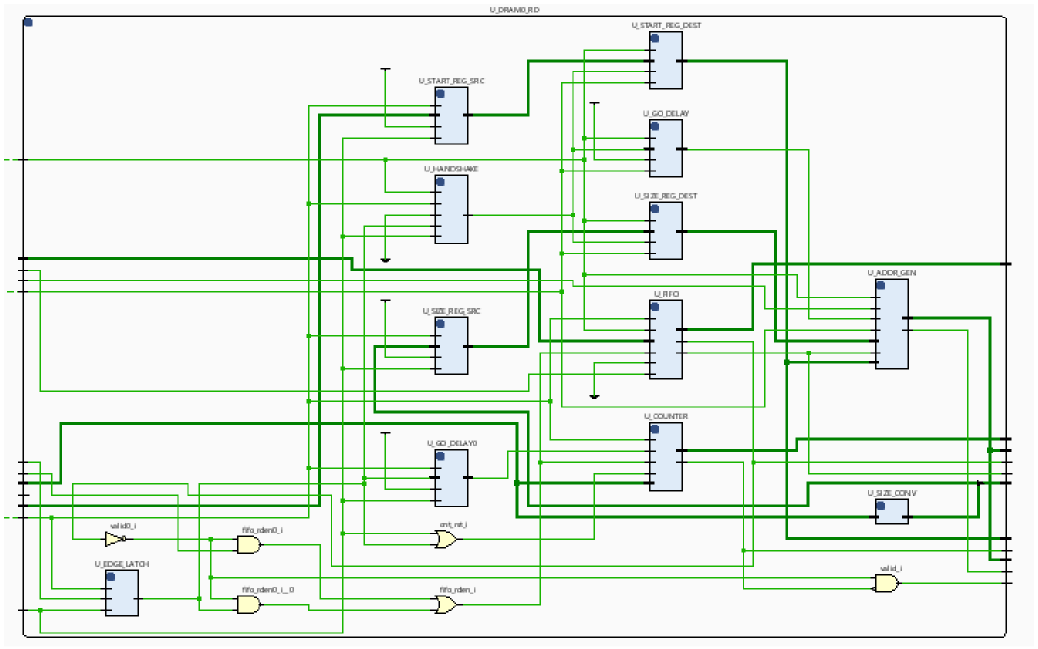
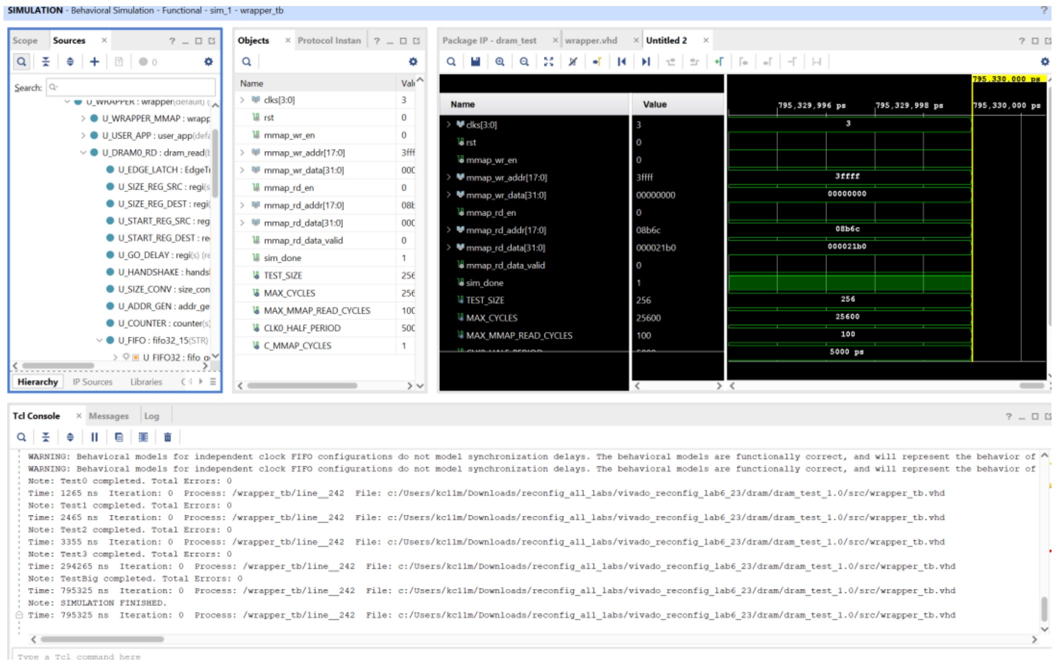


Figure 1: schematic of our dram_read

Next up, this image showcases the provided VHDL testbench running our dram_read code:



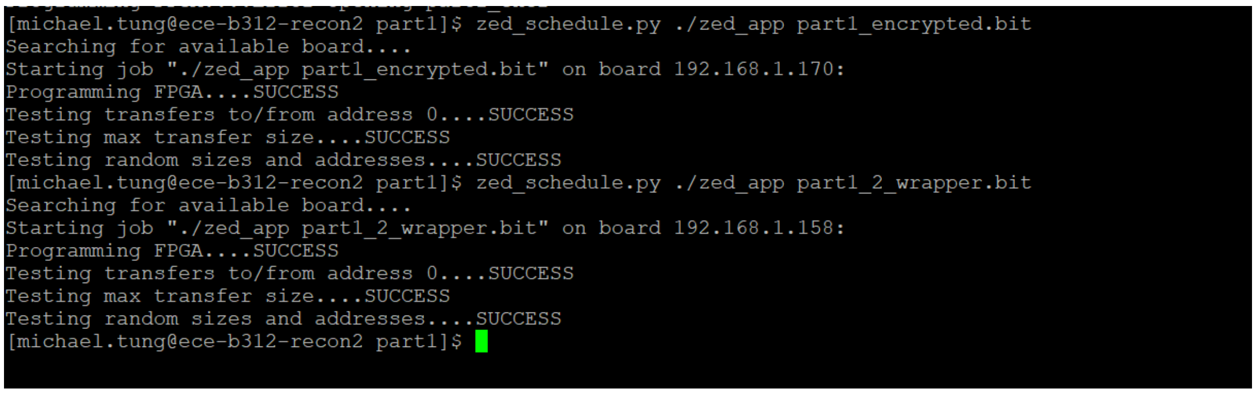
```
U_WRAPPER : wrapper(default)
  U_WRAPPER_MMAP : wrapp
  U_USER_APP : user_app(defa
  U_DRAM0_RD : dram_read(
    U_EDGE_LATCH : EdgeT
    U_SIZE_REG_SRC : reg(s
    U_SIZE_REG_DEST : regi
    U_START_REG_SRC : reg
    U_START_REG_DEST : re
    U_GO_DELAY : reg(s) (re
    U_HANDSHAKE : hands
    U_SIZE_CONV : size_con
    U_ADDR_GEN : addr_ge
    U_COUNTER : counter(s
    U_FIFO : fifo32_15(STR)
      U_FIFO32 : fifo_g
```

Name	Value	795,329,996 ps … 795,330,000 ps
clks[3:0]	3	3
rst	0	
mmap_wr_en	0	
mmap_wr_addr[17:0]	3ffff	3ffff
mmap_wr_data[31:0]	00000000	00000000
mmap_rd_en	0	
mmap_rd_addr[17:0]	08b6c	08b6c
mmap_rd_data[31:0]	000021b0	000021b0
mmap_rd_data_valid	0	
sim_done	1	
TEST_SIZE	256	256
MAX_CYCLES	25600	25600
MAX_MMAP_READ_CYCLES	100	100
CLK0_HALF_PERIOD		5000 ps
C_MMAP_CYCLES	1	

```
WARNING: Behavioral models for independent clock FIFO configurations do not model synchronization delays. The behavioral models are functionally correct, and will represent the behavior of
WARNING: Behavioral models for independent clock FIFO configurations do not model synchronization delays. The behavioral models are functionally correct, and will represent the behavior of
Note: Test0 completed. Total Errors: 0
Time: 1265 ns  Iteration: 0  Process: /wrapper_tb/line__242  File: c:/Users/kc11m/Downloads/reconfig_all_labs/vivado_reconfig_lab6_23/dram/dram_test_1.0/src/wrapper_tb.vhd
Note: Test1 completed. Total Errors: 0
Time: 2465 ns  Iteration: 0  Process: /wrapper_tb/line__242  File: c:/Users/kc11m/Downloads/reconfig_all_labs/vivado_reconfig_lab6_23/dram/dram_test_1.0/src/wrapper_tb.vhd
Note: Test2 completed. Total Errors: 0
Time: 3355 ns  Iteration: 0  Process: /wrapper_tb/line__242  File: c:/Users/kc11m/Downloads/reconfig_all_labs/vivado_reconfig_lab6_23/dram/dram_test_1.0/src/wrapper_tb.vhd
Note: Test3 completed. Total Errors: 0
Time: 294265 ns  Iteration: 0  Process: /wrapper_tb/line__242  File: c:/Users/kc11m/Downloads/reconfig_all_labs/vivado_reconfig_lab6_23/dram/dram_test_1.0/src/wrapper_tb.vhd
Note: TestBig completed. Total Errors: 0
Time: 795325 ns  Iteration: 0  Process: /wrapper_tb/line__242  File: c:/Users/kc11m/Downloads/reconfig_all_labs/vivado_reconfig_lab6_23/dram/dram_test_1.0/src/wrapper_tb.vhd
Note: SIMULATION FINISHED.
Time: 795325 ns  Iteration: 0  Process: /wrapper_tb/line__242  File: c:/Users/kc11m/Downloads/reconfig_all_labs/vivado_reconfig_lab6_23/dram/dram_test_1.0/src/wrapper_tb.vhd
```

Figure 2: passing the tb provided using our dram_read

After verification via testbench, we threw our dram_rd on the server (after a lot of annoying Vivado errors- in all honesty this project was more of a can you make Vivado work properly project than an RTL project).



```
[michael.tung@ece-b312-recon2 part1]$ zed_schedule.py ./zed_app part1_encrypted.bit
Searching for available board....
Starting job "./zed_app part1_encrypted.bit" on board 192.168.1.170:
Programming FPGA....SUCCESS
Testing transfers to/from address 0....SUCCESS
Testing max transfer size....SUCCESS
Testing random sizes and addresses....SUCCESS
[michael.tung@ece-b312-recon2 part1]$ zed_schedule.py ./zed_app part1_2_wrapper.bit
Searching for available board....
Starting job "./zed_app part1_2_wrapper.bit" on board 192.168.1.158:
Programming FPGA....SUCCESS
Testing transfers to/from address 0....SUCCESS
Testing max transfer size....SUCCESS
Testing random sizes and addresses....SUCCESS
[michael.tung@ece-b312-recon2 part1]$
```

Figure 3: our dram_read on the physical board

Part 2: Convolution Pipeline

Next up after dram_rd is convolution. Here are two screenshots of our convolution schematic, using a kernel size of 3 so it looks visually pleasing to look at.

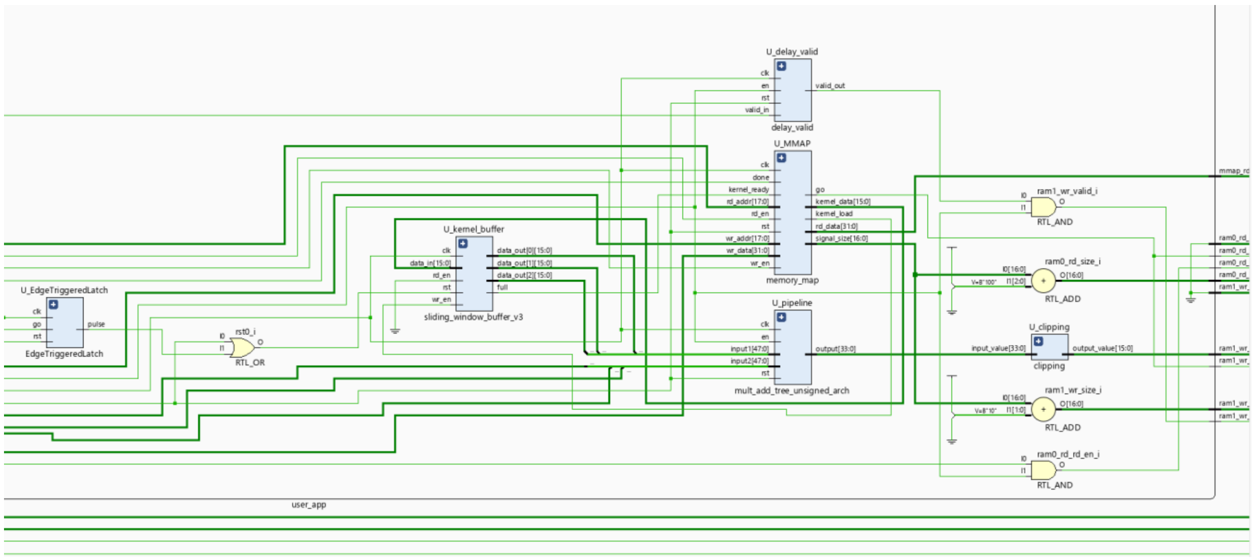


Figure 4: our schematic using 3 kernel size for better visual representation

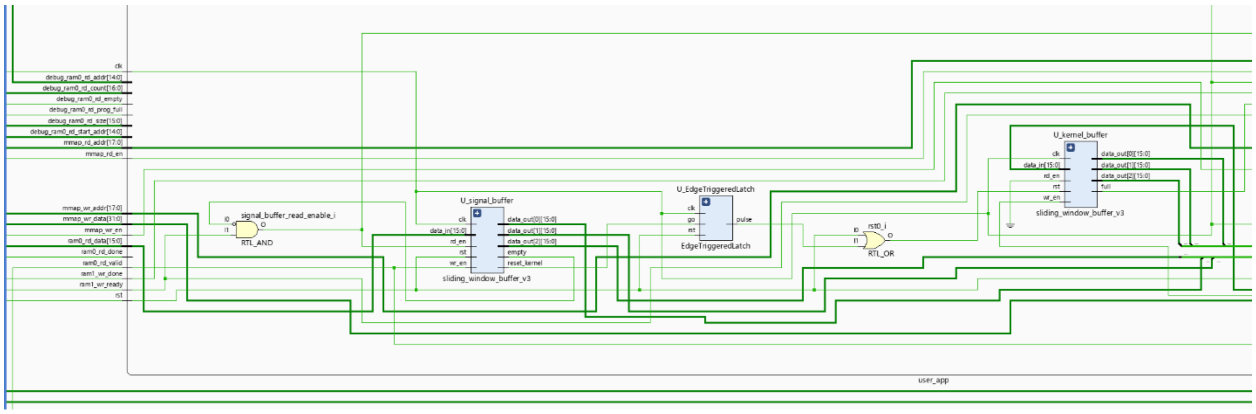
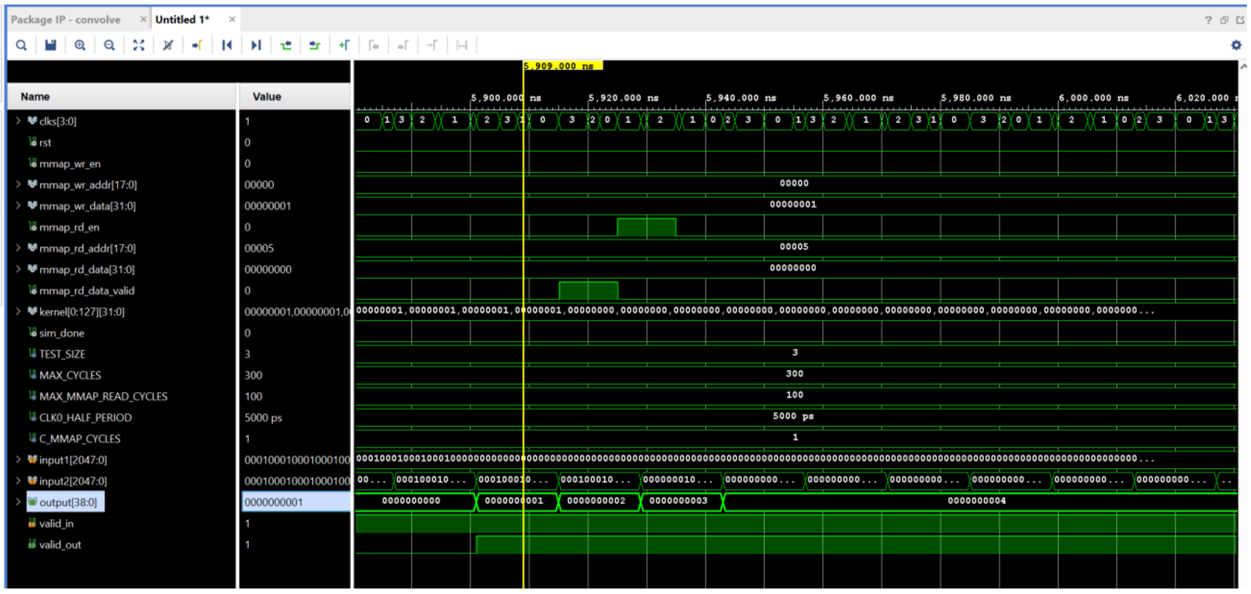


Figure 5: our schematic part 2

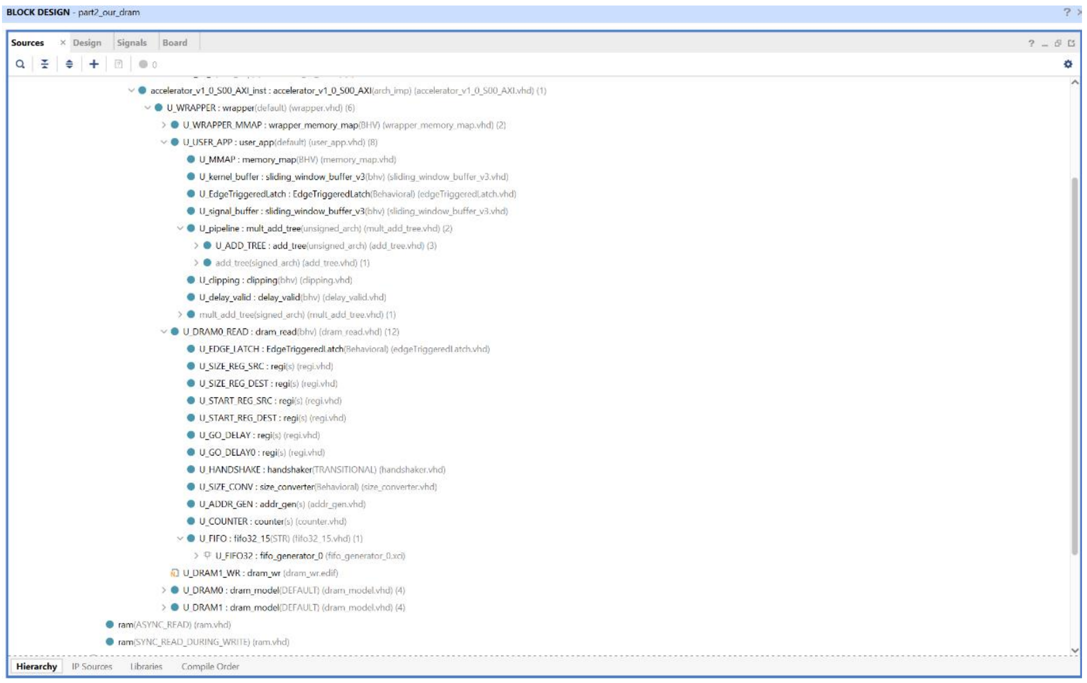
Here’s a snippet of the convolution pipeline working in simulation with a sample input.



Name	Value
clks[3:0]	1
rst	0
mmap_wr_en	0
mmap_wr_addr[17:0]	00000
mmap_wr_data[31:0]	00000001
mmap_rd_en	0
mmap_rd_addr[17:0]	00005
mmap_rd_data[31:0]	00000000
mmap_rd_data_valid	0
kernel[0:127][31:0]	00000001,00000001,0
sim_done	0
TEST_SIZE	3
MAX_CYCLES	300
MAX_MMAP_READ_CYCLES	100
CLK0_HALF_PERIOD	5000 ps
C_MMAP_CYCLES	1
input1[2047:0]	000100010001000100
input2[2047:0]	000100010001000100
output[38:0]	0000000001
valid_in	1
valid_out	1

Figure 6: output of the pipeline which input = 1, 1, 1, kernel = 1, 1, 1, 1 (128 of them), output = 1, 2, 3, 4, 4….. , 2, 1

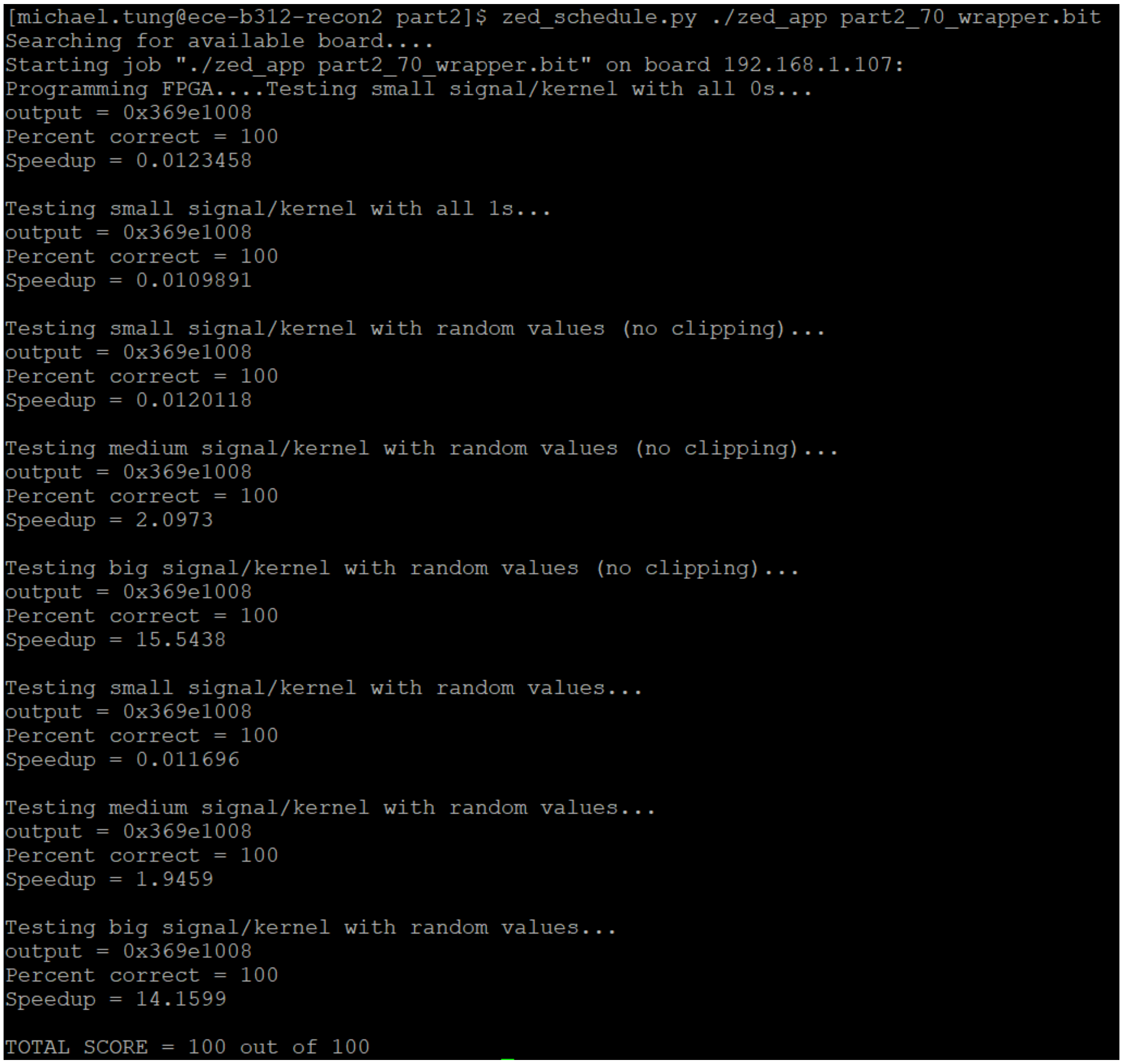
Next up is a really ugly screenshot of the project manager in Vivado holding our dram_read and convolution modules/entities.



```
BLOCK DESIGN - part2_our_dram
Sources  Design  Signals  Board

accelerator_v1_0_S00_AXI_inst : accelerator_v1_0_S00_AXI(arch_imp) (accelerator_v1_0_S00_AXI.vhd) (1)
  U_WRAPPER : wrapper(default) (wrapper.vhd) (6)
    U_WRAPPER_MMAP : wrapper_memory_map(BHV) (wrapper_memory_map.vhd) (2)
    U_USER_APP : user_app(default) (user_app.vhd) (8)
      U_MMAP : memory_map(BHV) (memory_map.vhd)
      U_kernel_buffer : sliding_window_buffer_v3(bhv) (sliding_window_buffer_v3.vhd)
      U_EdgeTriggeredLatch : EdgeTriggeredLatch(Behavioral) (edgeTriggeredLatch.vhd)
      U_signal_buffer : sliding_window_buffer_v3(bhv) (sliding_window_buffer_v3.vhd)
      U_pipeline : mult_add_tree(unsigned_arch) (mult_add_tree.vhd) (2)
        U_ADD_TREE : add_tree(unsigned_arch) (add_tree.vhd) (3)
        add_tree(signed_arch) (add_tree.vhd) (1)
      U_clipping : clipping(bhv) (clipping.vhd)
      U_delay_valid : delay_valid(bhv) (delay_valid.vhd)
      mult_add_tree(signed_arch) (mult_add_tree.vhd) (1)
    U_DRAM0_READ : dram_read(bhv) (dram_read.vhd) (12)
      U_EDGE_LATCH : EdgeTriggeredLatch(Behavioral) (edgeTriggeredLatch.vhd)
      U_SIZE_REG_SRC : regi(s) (regi.vhd)
      U_SIZE_REG_DEST : regi(s) (regi.vhd)
      U_START_REG_SRC : regi(s) (regi.vhd)
      U_START_REG_DEST : regi(s) (regi.vhd)
      U_GO_DELAY : regi(s) (regi.vhd)
      U_GO_DELAY0 : regi(s) (regi.vhd)
      U_HANDSHAKE : handshaker(TRANSITIONAL) (handshaker.vhd)
      U_SIZE_CONV : size_converter(Behavioral) (size_converter.vhd)
      U_ADDR_GEN : addr_gen(s) (addr_gen.vhd)
      U_COUNTER : counter(s) (counter.vhd)
      U_FIFO : fifo32_15(STR) (fifo32_15.vhd) (1)
        U_FIFO32 : fifo_generator_0 (fifo_generator_0.xci)
    U_DRAM1_WR : dram_wr (dram_wr.edif)
    U_DRAM0 : dram_model(DEFAULT) (dram_model.vhd) (4)
    U_DRAM1 : dram_model(DEFAULT) (dram_model.vhd) (4)
ram(ASYNC_READ) (ram.vhd)
ram(SYNC_READ_DURING_WRITE) (ram.vhd)

Hierarchy  IP Sources  Libraries  Compile Order
```

Figure 7: convolution pipeline using our dram_read

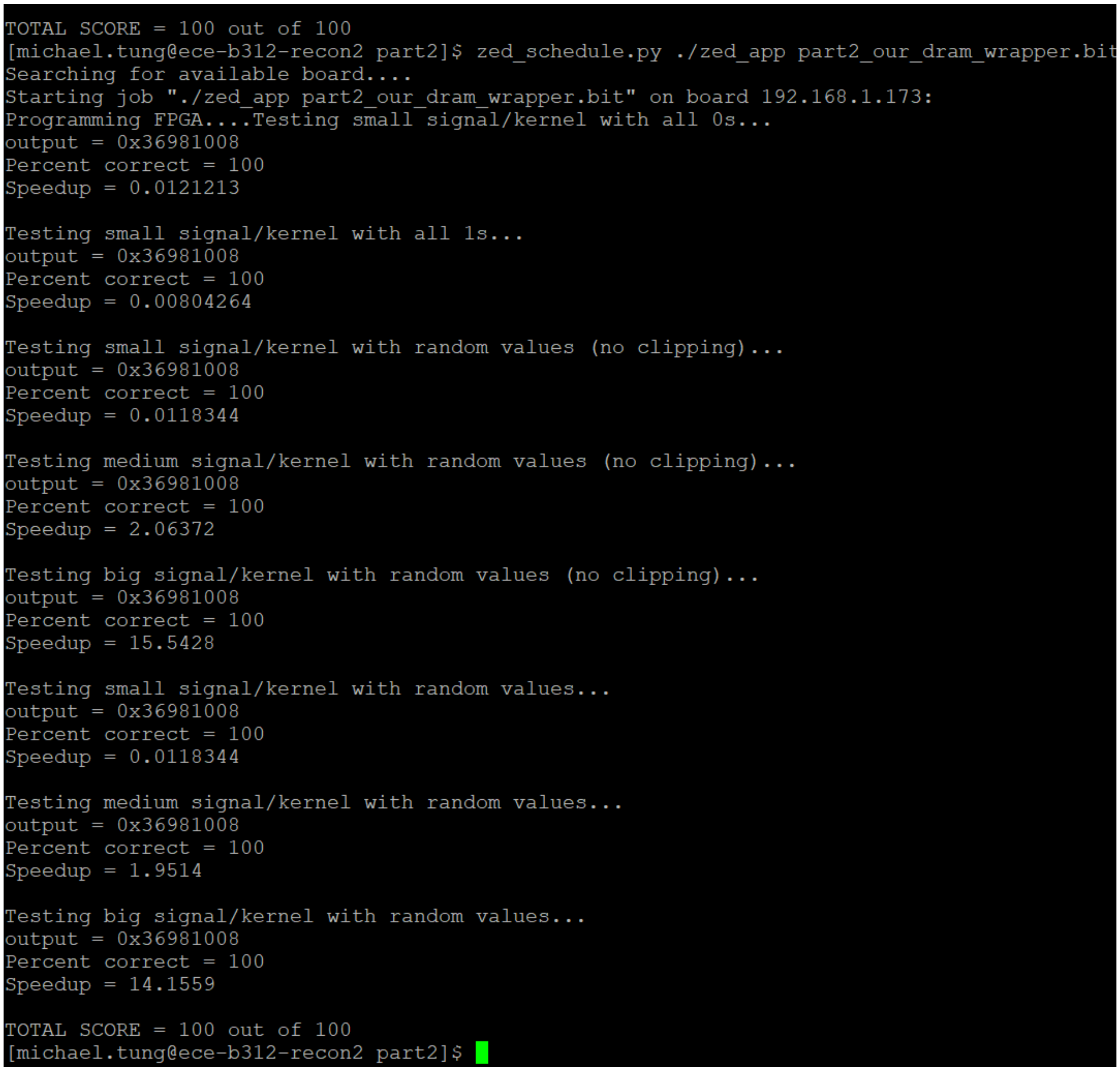
Here is a screenshot of the server running the code with the ****GIVEN** dram_rd** and convolution pipeline. Note the speedup, we think it looks good!



```
[michael.tung@ece-b312-recon2 part2]$ zed_schedule.py ./zed_app part2_70_wrapper.bit
Searching for available board....
Starting job "./zed_app part2_70_wrapper.bit" on board 192.168.1.107:
Programming FPGA....Testing small signal/kernel with all 0s...
output = 0x369e1008
Percent correct = 100
Speedup = 0.0123458

Testing small signal/kernel with all 1s...
output = 0x369e1008
Percent correct = 100
Speedup = 0.0109891

Testing small signal/kernel with random values (no clipping)...
output = 0x369e1008
Percent correct = 100
Speedup = 0.0120118

Testing medium signal/kernel with random values (no clipping)...
output = 0x369e1008
Percent correct = 100
Speedup = 2.0973

Testing big signal/kernel with random values (no clipping)...
output = 0x369e1008
Percent correct = 100
Speedup = 15.5438

Testing small signal/kernel with random values...
output = 0x369e1008
Percent correct = 100
Speedup = 0.011696

Testing medium signal/kernel with random values...
output = 0x369e1008
Percent correct = 100
Speedup = 1.9459

Testing big signal/kernel with random values...
output = 0x369e1008
Percent correct = 100
Speedup = 14.1599

TOTAL SCORE = 100 out of 100
```

Figure 8: given dram_rd and our convolution code

Here is a screenshot of the server running the code with the ****OUR** dram_read** and convolution pipeline (aka the whole project). Again, note the speedup.



```
TOTAL SCORE = 100 out of 100
[michael.tung@ece-b312-recon2 part2]$ zed_schedule.py ./zed_app part2_our_dram_wrapper.bit
Searching for available board....
Starting job "./zed_app part2_our_dram_wrapper.bit" on board 192.168.1.173:
Programming FPGA....Testing small signal/kernel with all 0s...
output = 0x36981008
Percent correct = 100
Speedup = 0.0121213

Testing small signal/kernel with all 1s...
output = 0x36981008
Percent correct = 100
Speedup = 0.00804264

Testing small signal/kernel with random values (no clipping)...
output = 0x36981008
Percent correct = 100
Speedup = 0.0118344

Testing medium signal/kernel with random values (no clipping)...
output = 0x36981008
Percent correct = 100
Speedup = 2.06372

Testing big signal/kernel with random values (no clipping)...
output = 0x36981008
Percent correct = 100
Speedup = 15.5428

Testing small signal/kernel with random values...
output = 0x36981008
Percent correct = 100
Speedup = 0.0118344

Testing medium signal/kernel with random values...
output = 0x36981008
Percent correct = 100
Speedup = 1.9514

Testing big signal/kernel with random values...
output = 0x36981008
Percent correct = 100
Speedup = 14.1559

TOTAL SCORE = 100 out of 100
[michael.tung@ece-b312-recon2 part2]$
```

Figure 9: our dram_read and convolution pipeline

Thanks!