

SOC Design

HW3 FIR Design

m111061547

蕭翔

Outline:

Overview-----

Block diagram-----

Operation-----

Resource usage-----

Report-----

GitHub link-----

Date:2023/10/22

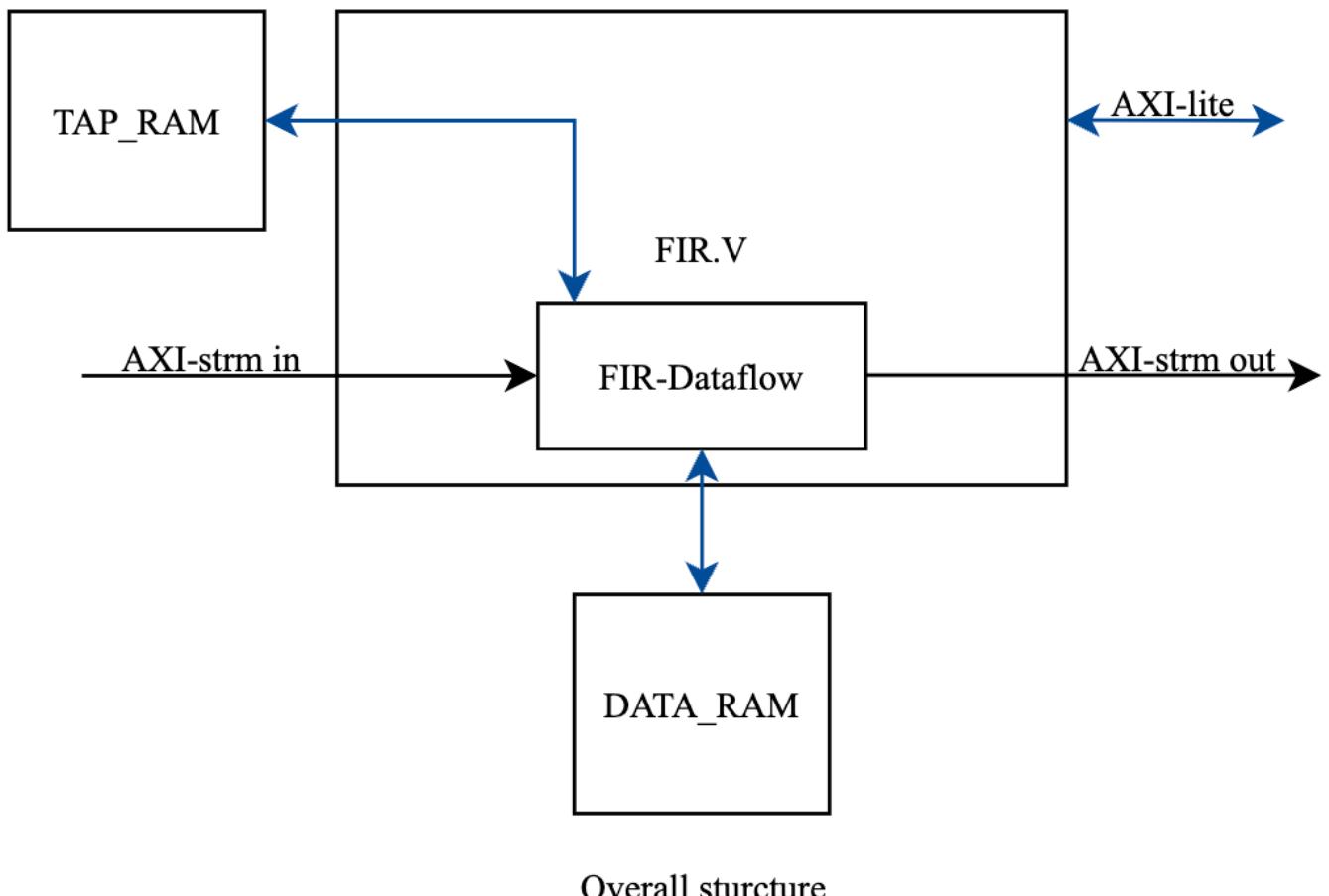
1. Overview

In this digital signal processing (DSP) finite impulse response (FIR) design, we have a total of 600 data points and 11 coefficients. However, we are constrained to use only one multiplier and one adder for our implementation.

The input data is received through the AXI-Stream protocol, which operates under the ARM architecture. It is essential to store this incoming data in a memory block known as Data Bram.

On the other hand, the FIR filter's coefficients, represented as "taps," are supplied through the AXI-Lite protocol.

The main objective of this design is to efficiently compute the convolution of the input data and the filter coefficients while adhering to the hardware constraints. The resulting output data is then transmitted using the AXI-Stream protocol.

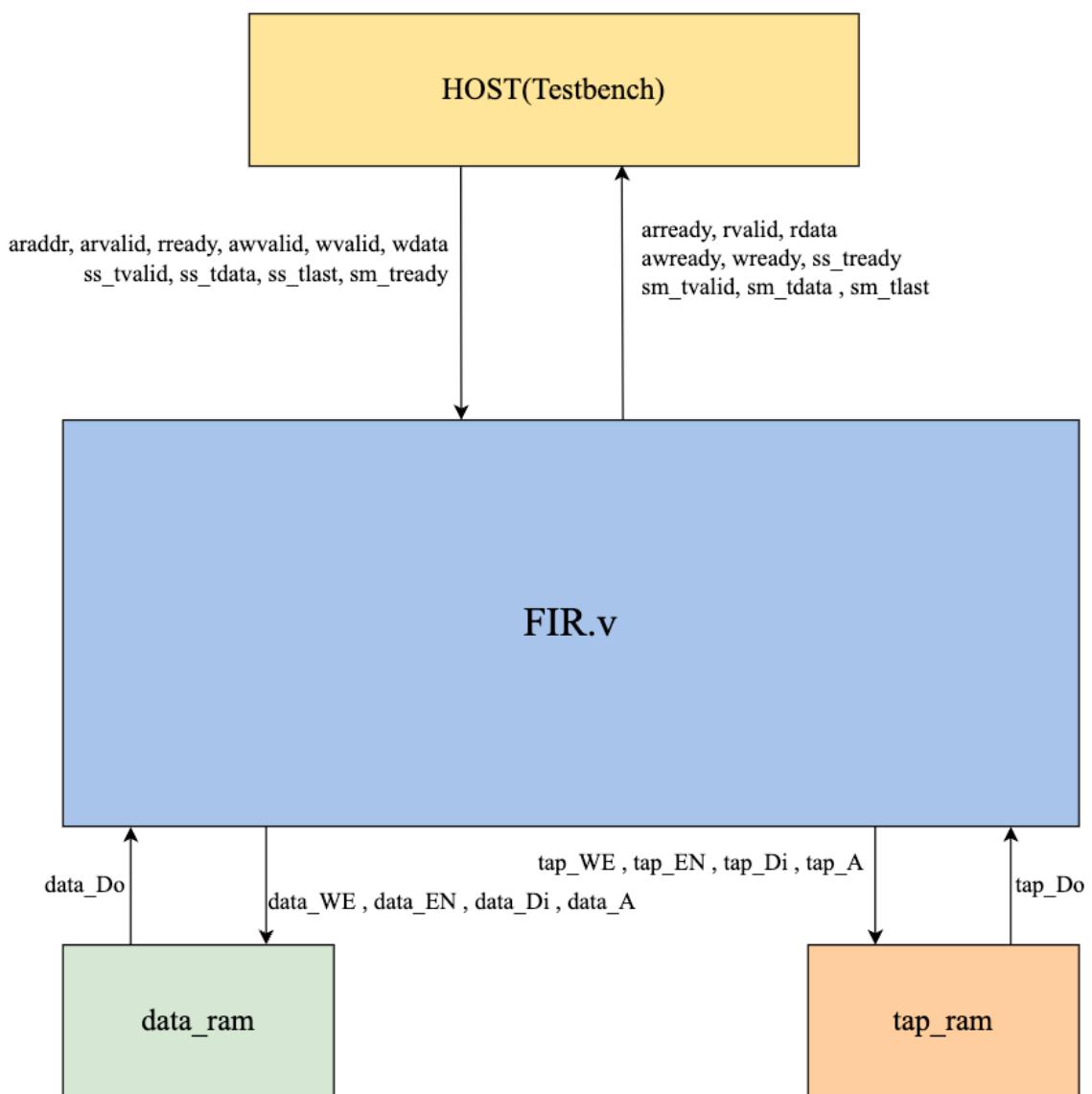


2. Block diagram

The block diagram depicts the key components of our system. This includes interfaces for data_ram, tap_ram, and the host-side (testbench).

Tap_ram serves as a straightforward storage unit for the 11 filter coefficients received via AXI-Lite.

Data_ram presents a more intricate challenge, handling 600 data points. It employs a shift_ram design to control data_A and facilitate data processing within the data_flow block.



3. Operation

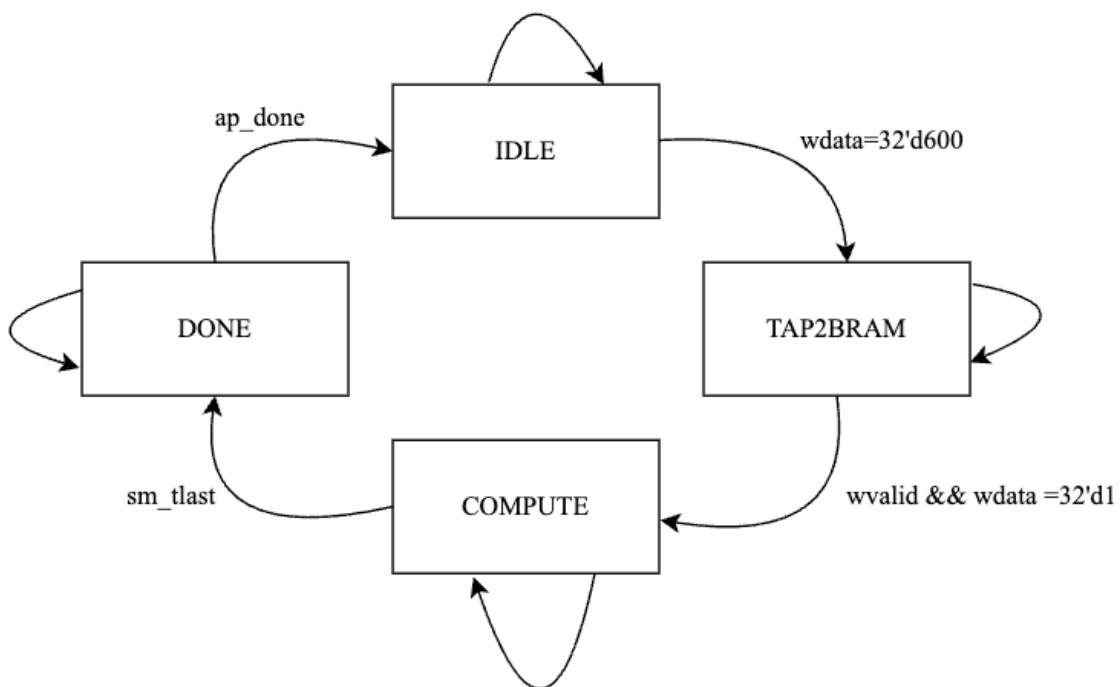
FSM:

IDLE State: Initially, the FSM is in the IDLE state, waiting for the first data to arrive. It remains in this state until it detects wdata with a value of $32'd600$, at which point it transitions to the TAP2BRAM state.

TAP2BRAM State: In the TAP2BRAM state, the FSM is responsible for storing the filter coefficients (taps). These coefficients are received via the AXI-Lite interface and are written into the tap_ram. Once the storage is complete, wvalid is asserted, and when rdata equals 0 (indicating ap_start), and wdata is set to 1, the FSM transitions to the Compute state.

Compute State: In the Compute state, the DSP FIR filtering operation commences. This is where the convolution between the input data and the filter taps is calculated.

Done State: When sm_tlast is asserted, indicating the end of the data stream, the FSM transitions to the done state. In this state, ap_done is asserted, indicating that the computation is finished, and the FSM returns to the IDLE state, awaiting the next data input.



Configuration Read:

ap_idle:

When rdata is 32'd4, signifying the ap_idle state, the FSM transitions to the TAP2BRAM state as soon as it reads a data_length of 600.



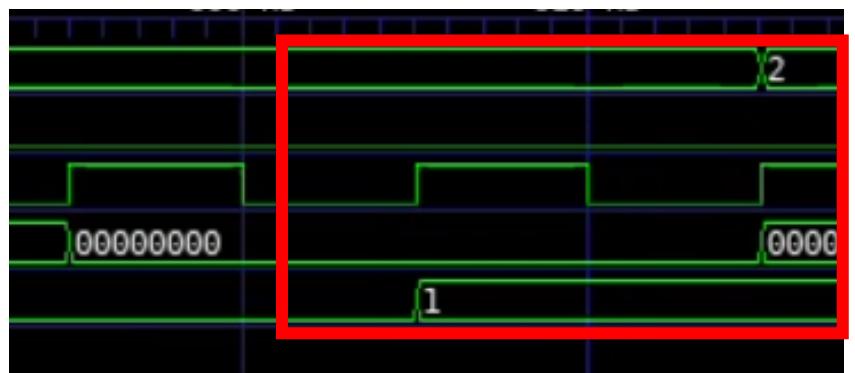
ap_start:

When transitioning into the Compute state, an rdata value of 32'b0 indicates that ap_start has been read, signaling the commencement of DSP computation.

STAT:1→2

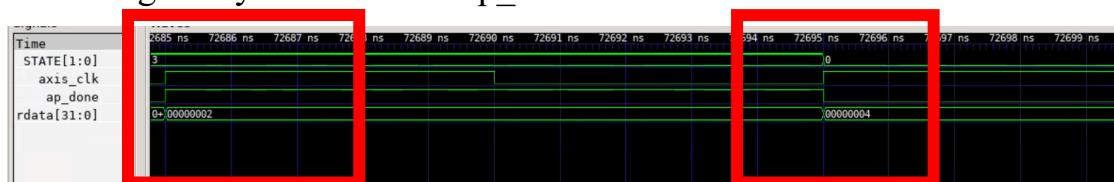
TAP2BRAM→Compute

rdata:



ap_done:

Upon entering the Done state, an rdata value of 2 is detected, signifying the completion of DSP computation. Subsequently, as the final sm_tdata is processed, the FSM returns to the IDLE state, with rdata reading 4, indicating the system is in the ap_idle state.



TAP_RAM:

AXI-lite write

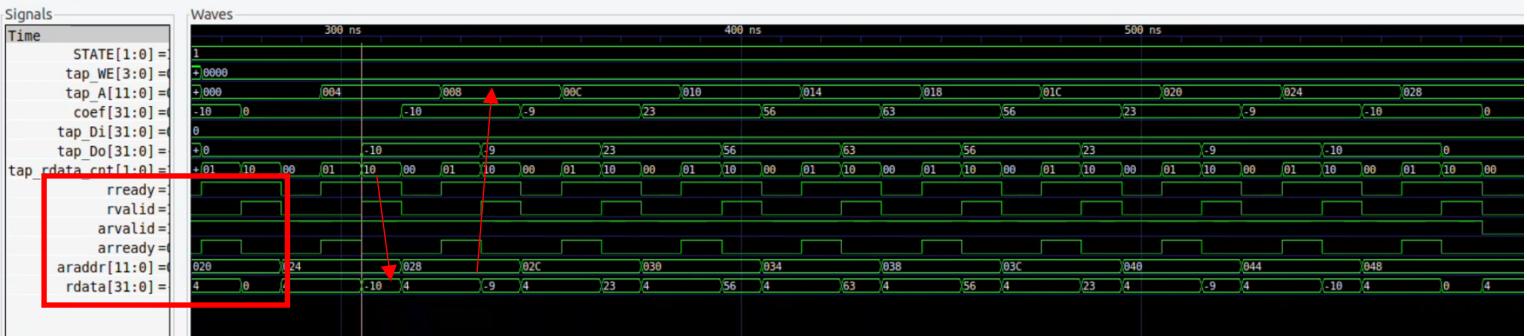


I've added a control signal, `tap_data_wen`, which synchronizes `awready` and `wready` during a handshake when `awvalid` is set to 1 from the host.

For `tap_WE`, it becomes 4'b1111 when both `wvalid` and `wready` are 1, allowing `wdata` to be written into `tap_Di` with a one-clock delay.

Additionally, I've implemented an address adjustment for `tap_A`. When `awaddr` is equal to or greater than 12'h020, (`awaddr - 12'h020`) is used as the address for `tap_A`. This ensures a clear address mapping for `tap_BRAM`, spanning from 12'h000 to 12'h028

AXI-lite read

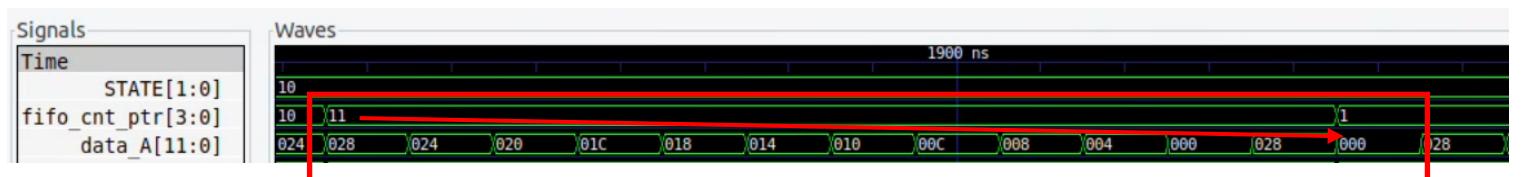
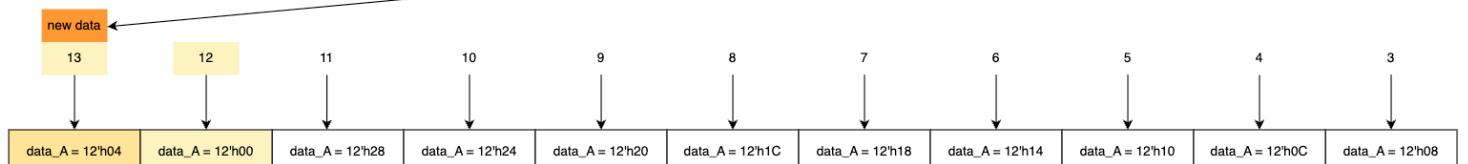
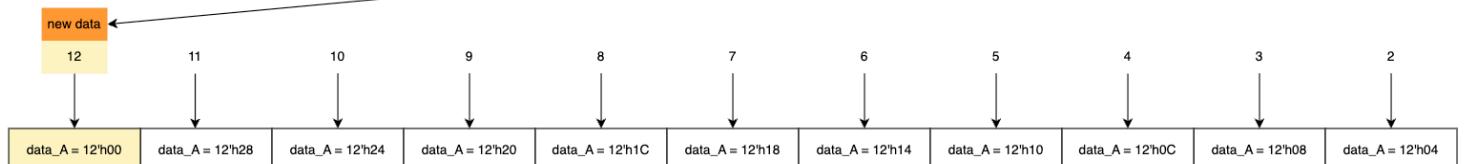
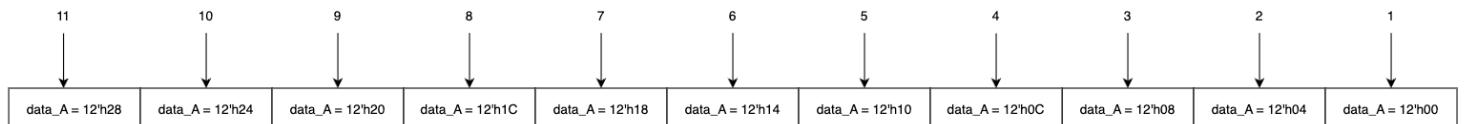
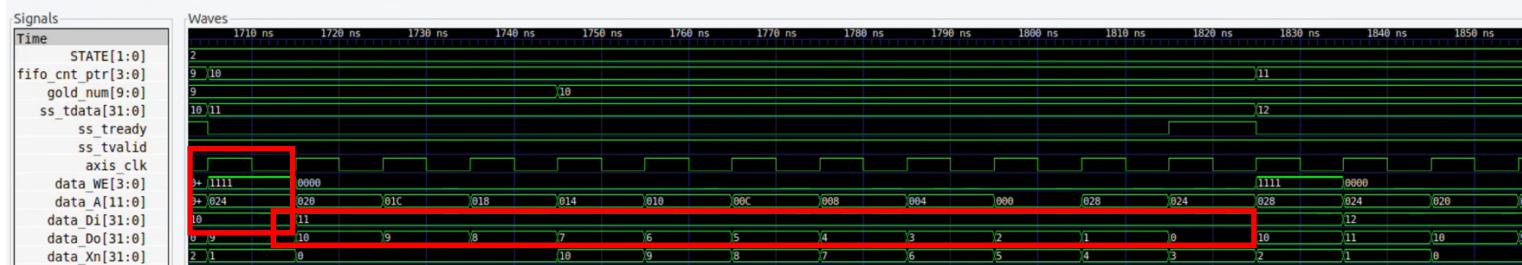


I've set up a `tap_rdata_cnt` to cycle through values 0-2. When it reaches 1, `already` is asserted, and `rvalid` is delayed by one clock cycle to align their timing. With `rvalid` and `rready` both high, you can read the `rdata` corresponding to the `tap_address`.

Additionally, when `already` and `arvalid` are both 1, `araddr` is received. To correctly access `tap_ram`, we subtract 12'h020 from `araddr`, aligning with the address configuration we've established.

These adjustments ensure synchronized data transfer and address calculations in our system.

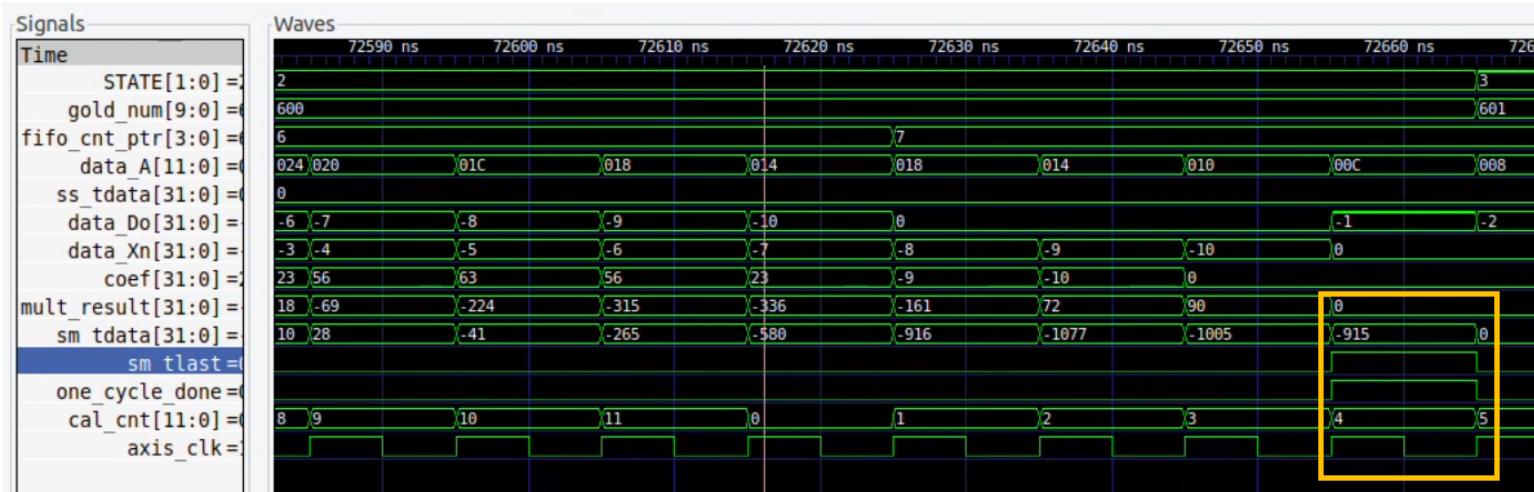
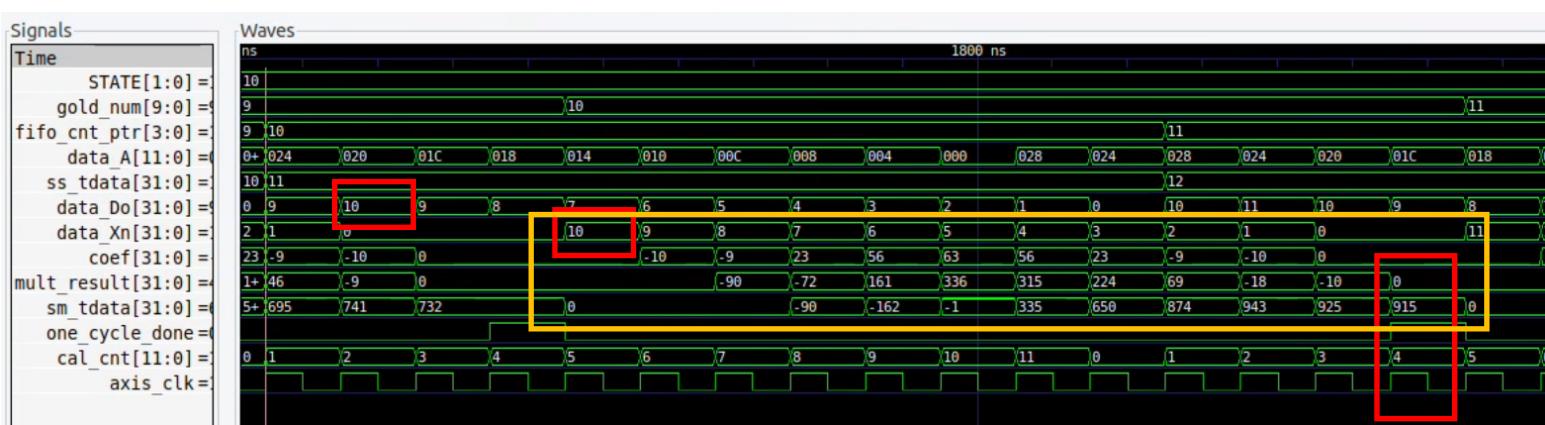
Shift_RAM:



I've implemented a data storage approach for data_ram, resembling a FIFO design. It includes a fifo_ptr_cnt to control data_A access. When ss_tready is set to 1, it enables the writing of ss_tdata into data_ram when data_we is 4'b1111.

When fifo_ptr_cnt reaches 11, data_A is reset to zero. This signifies that the multiplication operation has consumed the initial 11 data points, and we require new data for further DSP computation. If fifo_ptr_cnt t is not equal to 11, we multiply its value by 4 and assign it to data_A, facilitating the shift_ram functionality as depicted in the diagram.

DSP calculation:



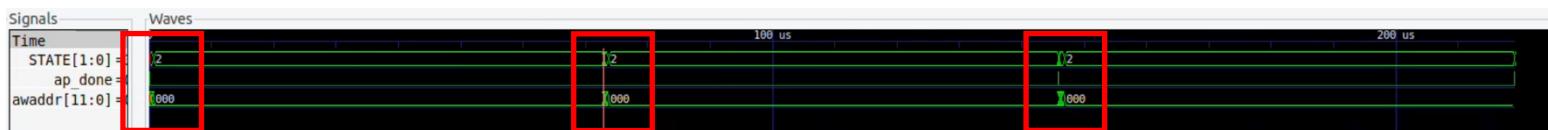
Our DSP computation follows the structure illustrated in the diagram. It involves a delay of 3 clock cycles from data_RAM to feed data_Xn. Tap_A provides the corresponding address to read tap_Do, and we employ a "coef" register to store these 11 taps. Subsequently, we perform pairwise multiplications and accumulate the results. The final accumulated value is sent as sm_tdata for comparison with the golden data.

Additionally, I've implemented a "cal_cnt" to count from 1 to 11 to ensure that data_ram receives all 11 data points. When "cal_cnt" reaches 4, an "enable_one_cycle_done" signal is set. At this point, sm_tdata's value is sent for comparison. Finally, when "gold_cnt" reaches 600, indicating the completion of 600 calculations, "sm_last" is asserted, signifying the conclusion of DSP computation and the transition to the "done_state."

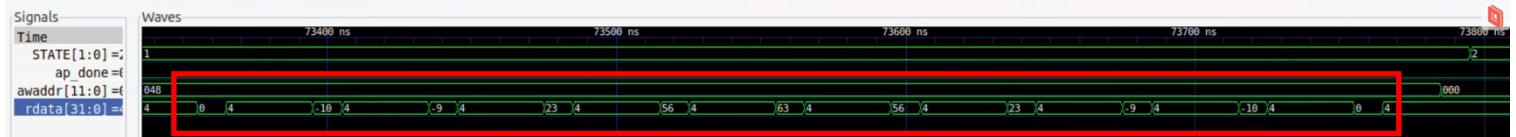
These steps outline the core of our DSP computation process, providing an efficient and structured approach to achieving our desired results.

TESTBENCH CYCLE:

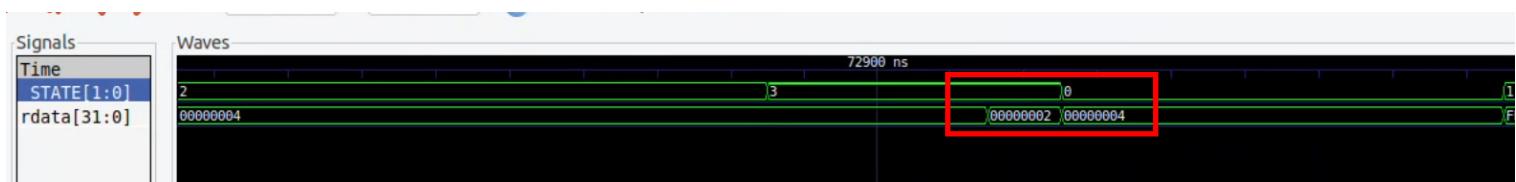
Due to the requirements of the topic, the testbench is set to test DSP (Digital Signal Processing) calculations three times. Accompanying waveforms are also provided to demonstrate that the state has been reset and that the AWADDR (Address Write Address) has been re-entered. This serves as evidence for the correctness of the configuration.



The above figure shows the waveform representing three cycles of calculation.



The figure above shows the waveform representing the tap values read out for the second time.



The figure above shows the waveform indicating that after confirming the state as "Done," it signifies the completion of the first DSP calculation. The rdata reads "ap_done" (rdata =2) and then returns to "ap_idle" (rdata =4) with the state set to "Idle," preparing for the second round of computation.

The testbench workflow is completed after the third and final calculation.

4. Resource usage

FF and LUT:

1. Slice Logic						
Site Type	Used	Fixed	Prohibited	Available	Util%	
Slice LUTs*	266	0	0	53200	0.50	
LUT as Logic	266	0	0	53200	0.50	
LUT as Memory	0	0	0	17400	0.00	
Slice Registers	301	0	0	106400	0.28	
Register as Flip Flop	301	0	0	106400	0.28	
Register as Latch	0	0	0	106400	0.00	
F7 Muxes	0	0	0	26600	0.00	
F8 Muxes	0	0	0	13300	0.00	

Bram:

2. Memory						
Site Type	Used	Fixed	Prohibited	Available	Util%	
Block RAM Tile	0	0	0	140	0.00	
RAMB36/FIFO*	0	0	0	140	0.00	
RAMB18	0	0	0	280	0.00	

DSP:

3. DSP						
Site Type	Used	Fixed	Prohibited	Available	Util%	
DSPs	3	0	0	220	1.36	
DSP48E1 only	3	0	0	0	0.00	

5. Report

RTL simulation

First:

```
-----End the data input(AXI-Stream)----- ||  
[PASS] [Pattern      598] Golden answer:      -1098, Your answer:      -1098  
[PASS] [Pattern      599] Golden answer:      -915, Your answer:      -915  
OK: exp =      2, rdata =      2  
OK: exp =      4, rdata =      4  
Latency:    7206  
=====TASK: axilite_write=====  
---Start the coefficient input(AXI-lite)---  
=====TASK: streaming_write_task=====  
-----Start simulation-----  
---Start the data input(AXI-Stream)---  
=====TASK: streaming_read_task=====  
---End the coefficient input(AXI-lite)---  
---End the coefficient input(AXI-lite)---  
  
=====TASK: axilite_read_task=====  
Check Coefficient ...  
OK: exp =      0, rdata =      0  
OK: exp =     -10, rdata =     -10  
OK: exp =      -9, rdata =      -9  
OK: exp =      23, rdata =      23  
OK: exp =      56, rdata =      56  
OK: exp =      63, rdata =      63  
OK: exp =      56, rdata =      56  
OK: exp =      23, rdata =      23  
OK: exp =      -9, rdata =      -9  
OK: exp =     -10, rdata =     -10  
OK: exp =      0, rdata =      0  
Tape programming done ...  
Start FIR
```

Second:

```
-----End the data input(AXI-Stream)----- ||  
[PASS] [Pattern      598] Golden answer:      -1098, Your answer:      -1098  
[PASS] [Pattern      599] Golden answer:      -915, Your answer:      -915  
OK: exp =      2, rdata =      2  
OK: exp =      4, rdata =      4  
Latency:    7213  
=====TASK: axilite_write=====  
---Start the coefficient input(AXI-lite)---  
=====TASK: streaming_write_task=====  
-----Start simulation-----  
---Start the data input(AXI-Stream)---  
=====TASK: streaming_read_task=====  
---End the coefficient input(AXI-lite)---  
---End the coefficient input(AXI-lite)---  
  
=====TASK: axilite_read_task=====  
Check Coefficient ...  
OK: exp =      0, rdata =      0  
OK: exp =     -10, rdata =     -10  
OK: exp =      -9, rdata =      -9  
OK: exp =      23, rdata =      23  
OK: exp =      56, rdata =      56  
OK: exp =      63, rdata =      63  
OK: exp =      56, rdata =      56  
OK: exp =      23, rdata =      23  
OK: exp =      -9, rdata =      -9  
OK: exp =     -10, rdata =     -10  
OK: exp =      0, rdata =      0  
Tape programming done ...  
Start FIR
```

Third:

```
-----End the data input(AXI-Stream)----- ||  
[PASS] [Pattern      598] Golden answer:      -1098, Your answer:      -1098  
[PASS] [Pattern      599] Golden answer:      -915, Your answer:      -915  
OK: exp =          2, rdata =          2  
OK: exp =          4, rdata =          4  
Latency:    7213  
-----  
-----Congratulations! Pass-----
```

Timing

clock period:7.5ns

Critical path:

```
Max Delay Paths  
-----  
Slack (MET) :      0.376ns (required time - arrival time)  
Source:           data_Xn_reg[16]/C  
                  (rising edge-triggered cell FDCE clocked by Sysclk {rise@0.000ns fall@3.750ns period=7.500ns})  
Destination:       mult_result0_1/PCIN[0]  
                  (rising edge-triggered cell DSP48E1 clocked by Sysclk {rise@0.000ns fall@3.750ns period=7.500ns})  
Path Group:        Sysclk  
Path Type:         Setup (Max at Slow Process Corner)  
Requirement:       7.500ns (Sysclk rise@7.500ns - Sysclk rise@0.000ns)  
Data Path Delay:   5.544ns (logic 4.689ns (84.582%) route 0.855ns (15.418%))  
Logic Levels:      1 (DSP48E1=1)  
Clock Path Skew:   -0.145ns (DCD - SCD + CPR)  
Destination Clock Delay (DCD): 2.128ns = ( 9.628 - 7.500 )  
Source Clock Delay (SCD): 2.456ns  
Clock Pessimism Removal (CPR): 0.184ns  
Clock Uncertainty:  0.035ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE  
Total System Jitter (TSJ): 0.071ns  
Total Input Jitter (TIJ): 0.000ns  
Discrete Jitter (DJ): 0.000ns  
Phase Error (PE): 0.000ns
```

6. GitHub link

https://github.com/sonnyshiau/lab_fir