

	wbr-stb-i
input	wbr-cyc-i
	wbr-we-i
	wbr-sal-i
	wbr-dat-i
	wbr-adr-i
output	wbr-dat-o
	wbr-ack-o

(Coef. ap-start.)

input	awr-stb-i
	awr-cyc-i
	awr-we-i
	awr-sal-i
	awr-dat-i
output	awr-dat-o
	awr-ack-o

(Xin)

input	sr-stb-i
	sr-cyc-i
	sr-we-i
	sr-sal-i
	sr-dat-i
output	sr-dat-o
	sr-ack-o

(Yout.)

input	sm-stb-i
output	sm-tvalid
	sm-tdata
	sm-tlast

sm-tvalid-D

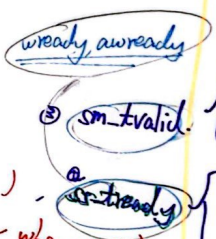
Read data I (Yout.)

- sm-tvalid
- ① wbr-cyc-i & ~ wbr-we-i
 - ② wbr-adr-i
 - ③ wbr-dat-o = sm-tdata
 - ④ wbr-ack-o = sm-tlast

Write data (Xin, Coef. ap-start.)

- sr-tvalid = (wbr-cyc-i & wbr-we-i)
- ① sr-tvalid
 - ② wbr-adr-i
 - ③ sr-tdata = wbr-dat-i
- sr-tlast
- sr-tvalid-D

~~ap-start~~



(sm-tvalid-A)
0x3800-0005

(sm-tvalid-D)
sm-tvalid
0x3800-0005
cs-tready-A

(sr-tready-D)
sr-tready

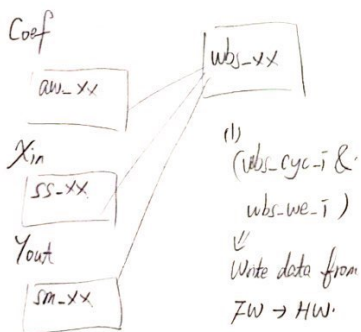
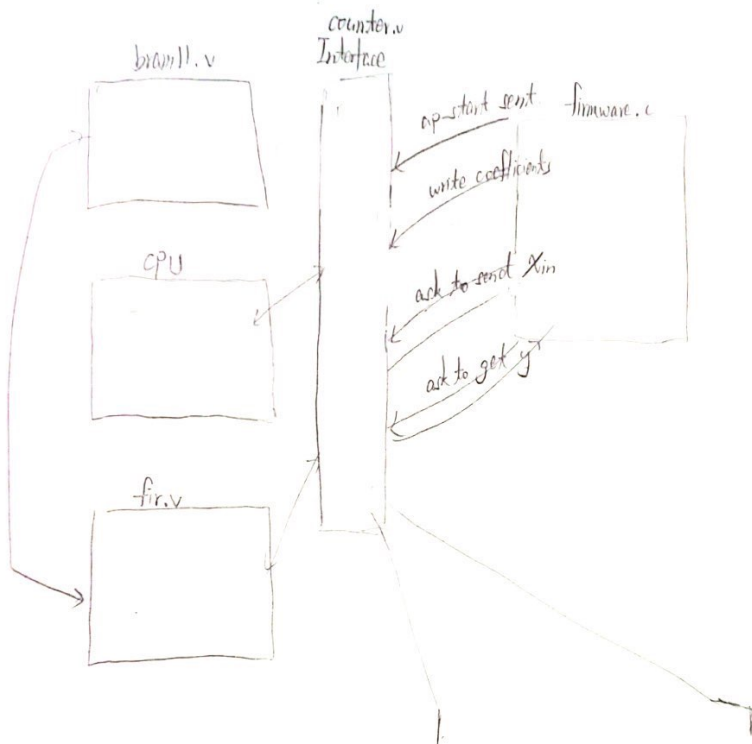
sr-tready-D = sr-tready
wready or

Write data II (Coef, ap-start)

wvalid = avalid^① = (wbs-cyc-i & wbs-we-i)

awaddr^② = wbs-adr-i

wdata^③ = wbs-dat



1) (wb3-cyc-i & ~wb3-we-i)
 ↓
 Write data from FW → HW.

2) (wb3-cyc-i & ~wb3-we-i)
 ↓
 Read data from HW to FW.

MMIO

ap_start 0x3000-0000
 fir-coef-reg 0x3800-0040
 fir-x-reg 0x3800-0080
 fir-y-reg 0x3800-00c0
 fir-begin-send-x 0x3000-0004
 fir-able-receive-y 0x3000-0008

Need { fir-begin-send-x
 fir-able-receive-y

to judge whether the data/accelerometer has been prepare to send/get the corresponding data

Report : SoC lab4-2

1. Introduction:

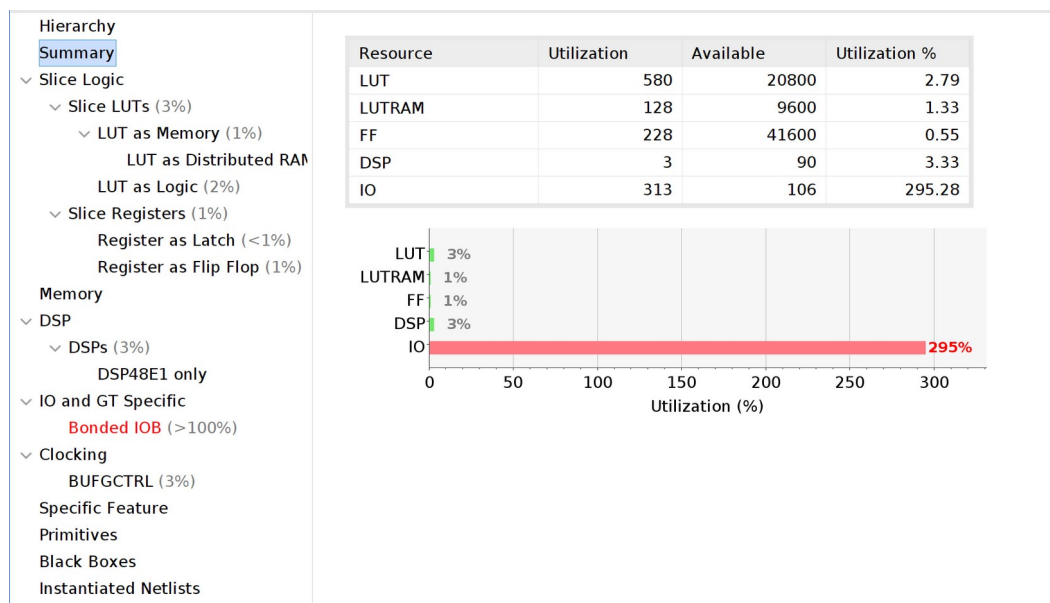
* There are two brams in the current design. One is for coefficients and the other is for the X value and the ap control values.

* The briefing data flow is as follows:

After reading one X, the Y could start be generated, and the X would be paused to generate till the Y has be computed.

Now let's turn to the firmware and accelerator interface. The data X and coefficients have been sent from firmware to hardware. Before sending the X value, firmware has to make sure the accelerator is able to get the data, and so does the Y output. Here, I create 2 registers to check the availability, fir_begin_send_x and fir_able_receive_y.

2. Area and timing report:



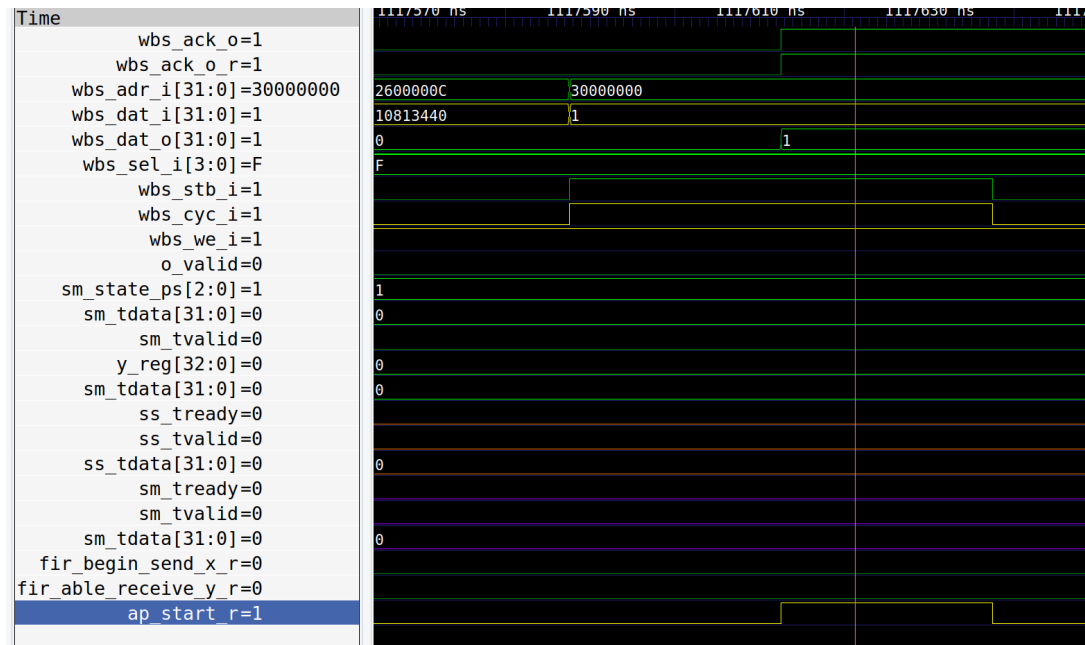
Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.938 ns	Worst Hold Slack (WHS): 0.139 ns	Worst Pulse Width Slack (WPWS): 0.250 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 1098	Total Number of Endpoints: 1098	Total Number of Endpoints: 345
All user specified timing constraints are met.		

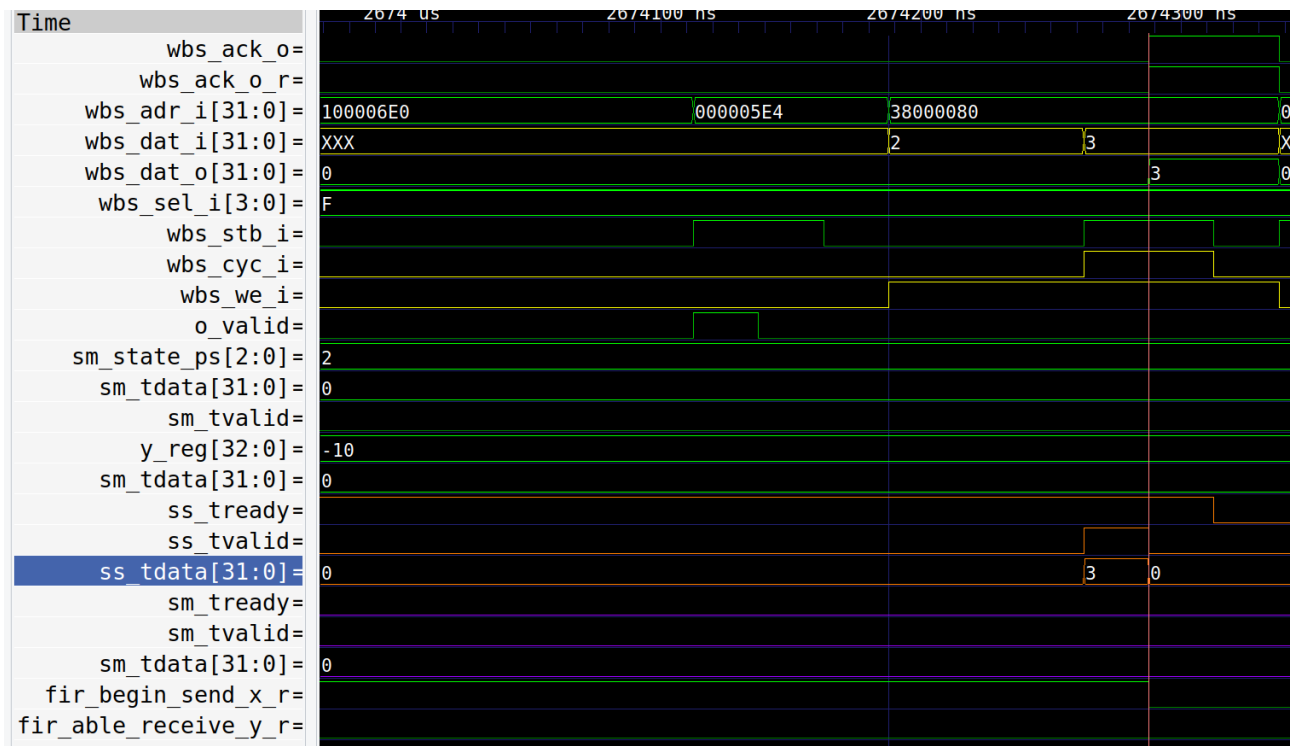
Slack is 0.938 ns under the cycle time = 15

3. Waveform and analysis:

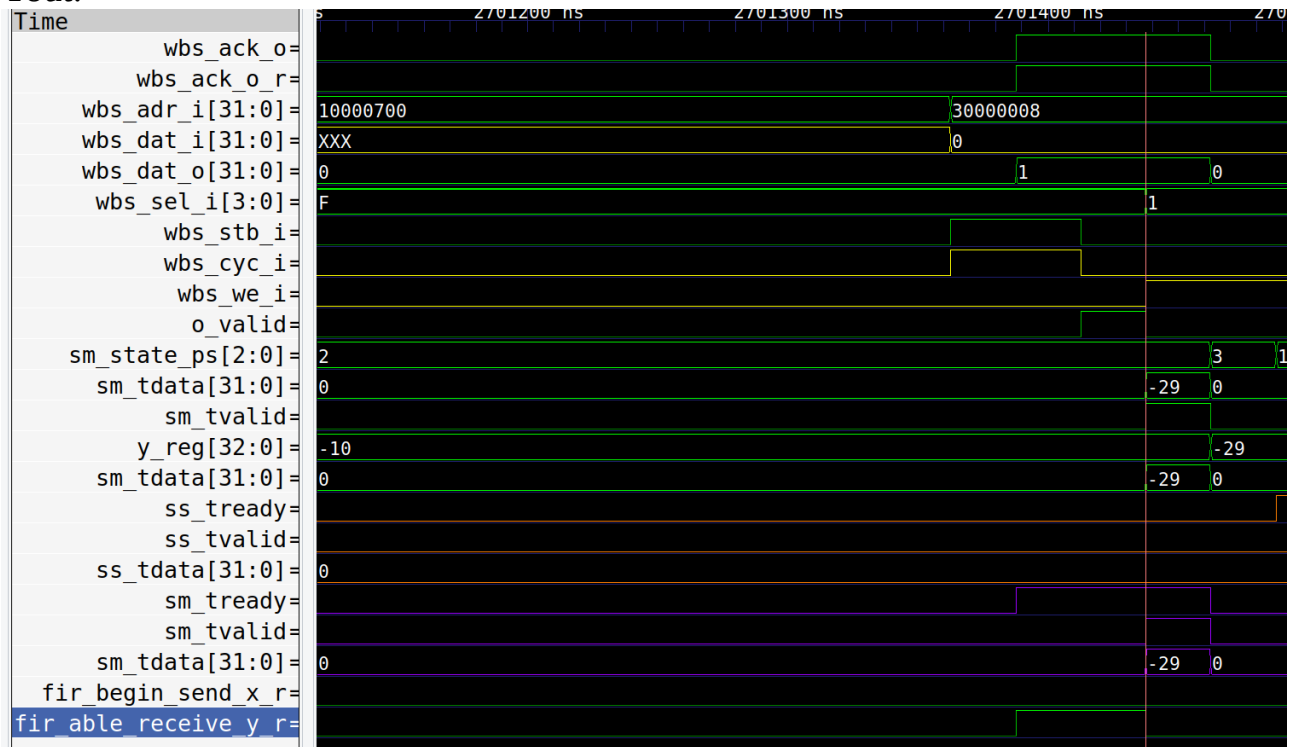
ap_start:



Xin:



Yout:



4. Let's say the number of Yout we want to produce is N. The total Xin is approximate to 11N. The design trick used in this design is that the bram we use to store the 11 necessary X for computing a Y is 10(one of the bram block we use to store the ap signals), therefore, we have to stall there while the current Y has not been produced. The latency would be a bit longer compared to the one used the bram12. The throughput is approximately 1 OP per cycle.

The another way to improve the performance is to add some buffers to prefetch the X so that we could decrease the stall time.

One thing want to mentioned is that the design of firmware sometimes would also important for performance while integrated to the hardware accelerator. The handshake mechanism could also be improved in order to enhance the performance.

<https://github.com/Raymonna/fir-caravel.git>