

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

(Coef. ap-start.)

input	awvalid
	awaddr
	awdata
	awdata
output	awready
	wready

(Xin)

input	<del>ss-valid</del>
	<del>ss-trstid</del>
	ss-trstid
	ss-tdata
	ss-tlast
output	ss-tready

(Yout.)

input	sm-trstid
output	sm-trstid
	sm-tdata
	sm-tlast
	sm-tlast

sm-trstid-D

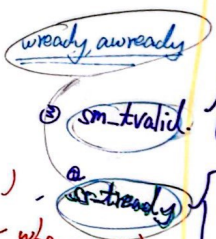
Read data I (Yout.)

- 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.)

- ① ss-trstid = (wbr-cyc-i & wbr-we-i)  
 ② wbr-adr-i  
 ③ ss-tdata = wbr-dat-i  
 ④ ss-tlast  
 ⑤ ss-tready

~~X ap-start~~



(sm-trstid-A)  
0x3800-0005

(sm-trstid-D)  
sm-trstid

0x3800-0005  
ss-trstid-A

(ss-trstid-D)  
ss-trstid

ss-trstid-D = ss-trstid

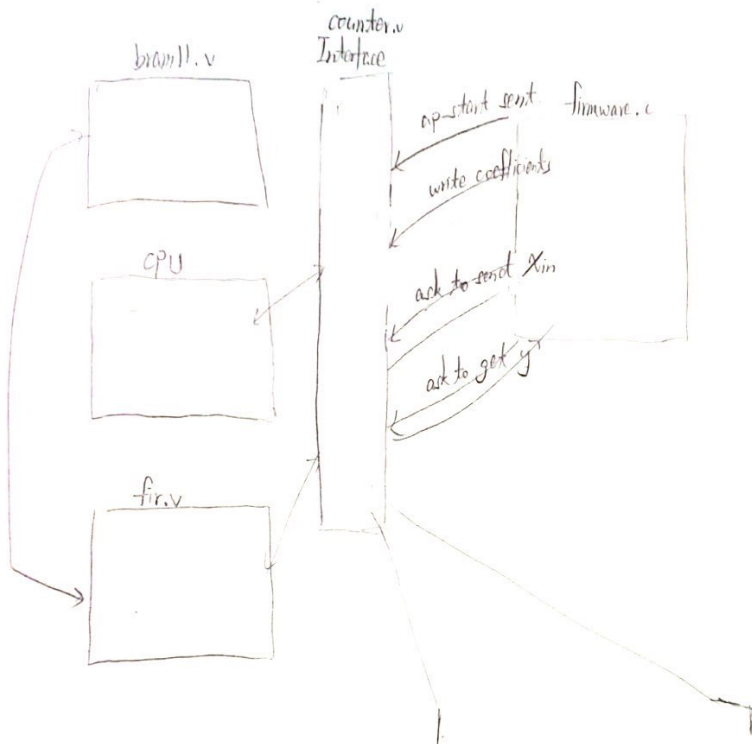
wready or

Write data II (Coef, ap-start)

wvalid = avalid<sup>①</sup> = (wbs-cyc-i & wbs-we-i)

awaddr<sup>②</sup> = wbs-adr-i

wdata<sup>③</sup> = wbs-dat



Coef

aw-xx

Xin

ss-xx

Yout

sm-xx

wbj-xx

(1) (wbs\_cyc-i & wbs\_we-i)  
 ↓  
 Write data from FW → HW.

(2) (wbs\_cyc-i & ~wbs\_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 0x3800-0080

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:

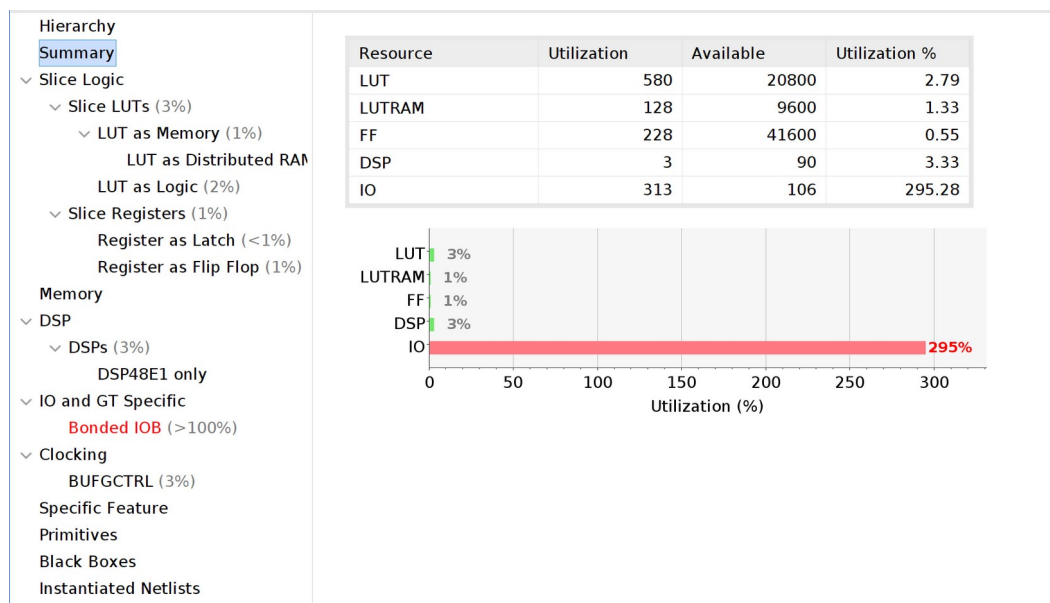
\* There are two brams in the current design. One is for coefficients and the other is for the X value and the ap control values.

\* 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

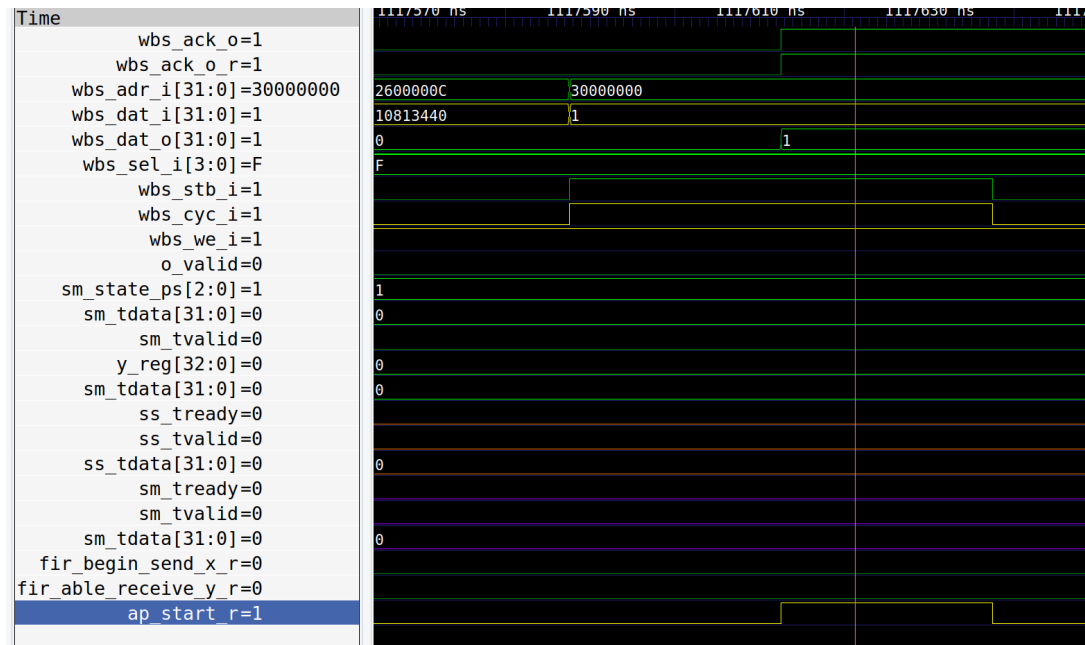
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.938 ns	Worst Hold Slack (WHS): 0.139 ns	Worst Pulse Width Slack (WPWS): 0.250 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing 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.

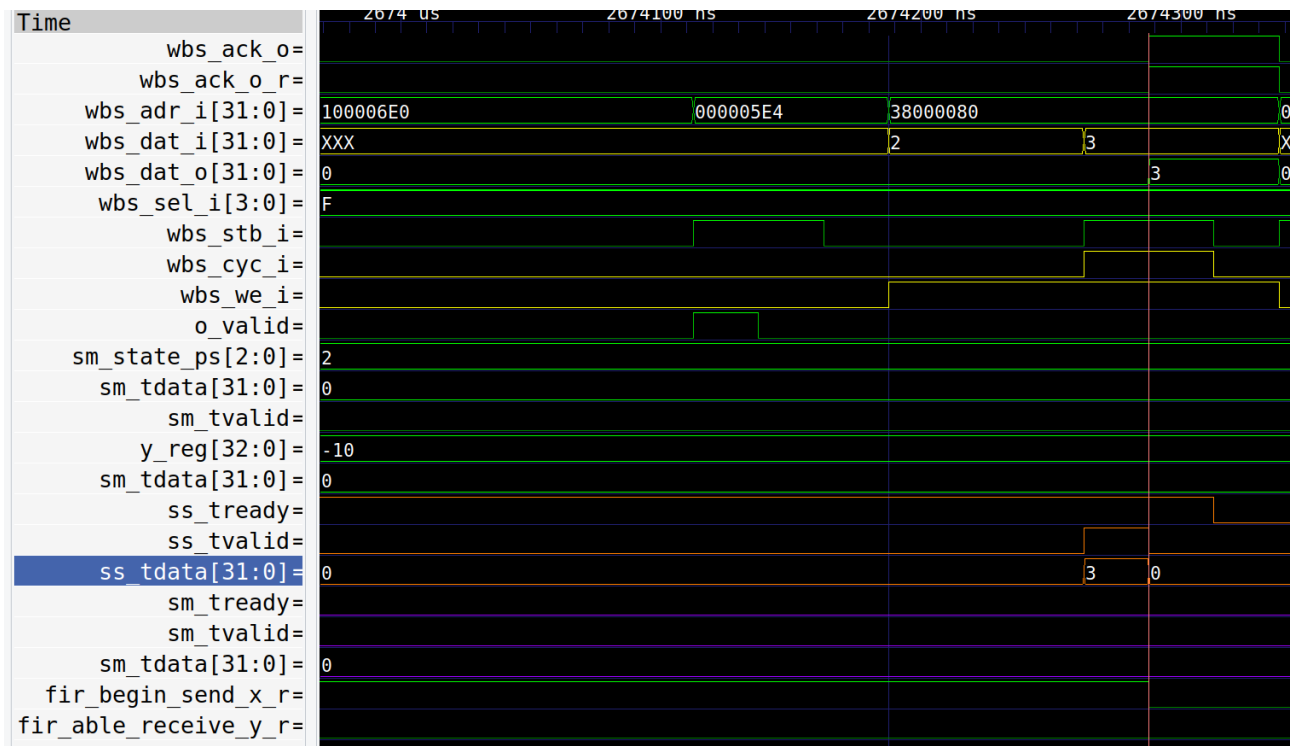
Slack is 0.938 ns under the cycle time = 15

### 3. Waveform and analysis:

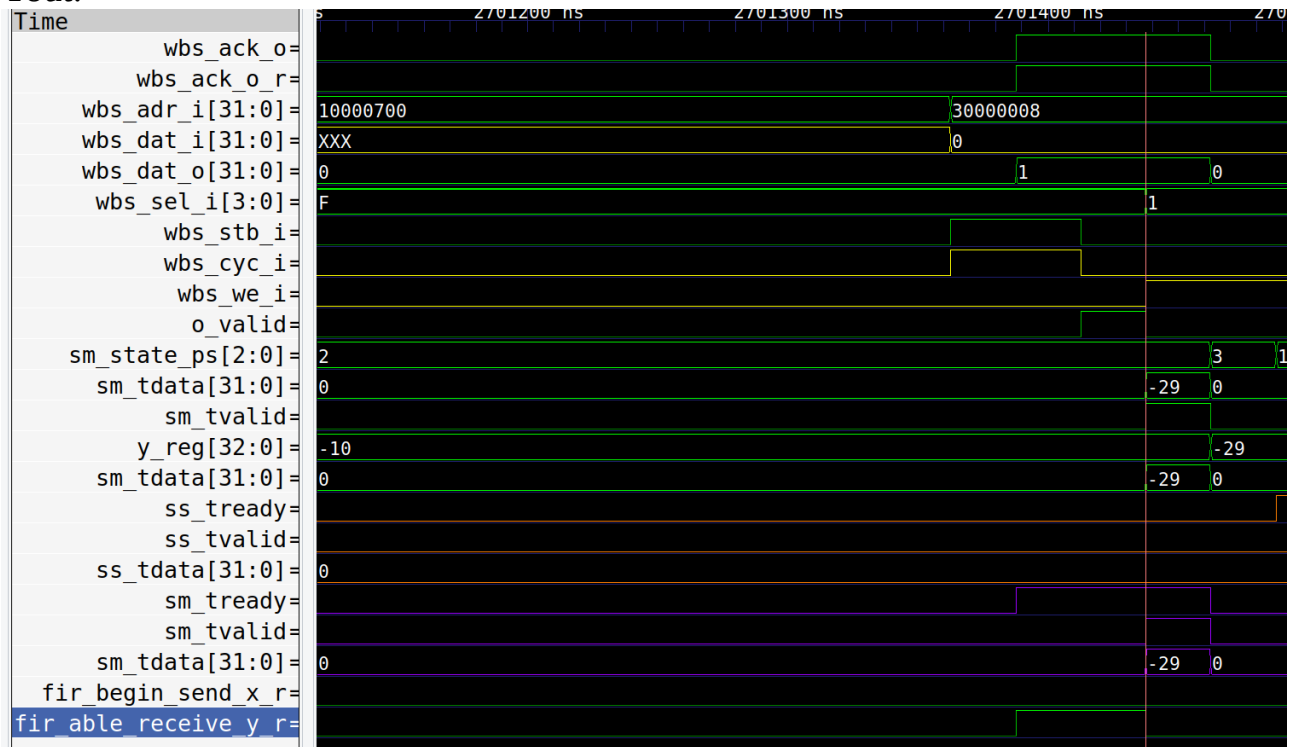
ap\_start:



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.