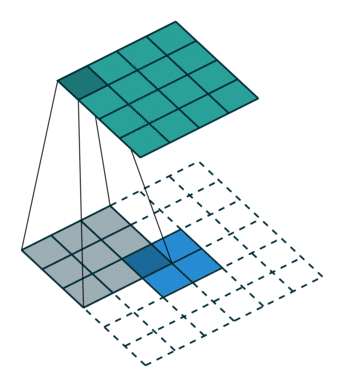
Convolution IP

Design Document

Revision 1 16 August 2019



ETRI AIRTL curriculum

Changhun Kim

E-mail : [means0416@gmail.com](mailto:means0416@gmail.com)

Tel : 82-10-9452-3534

INDEX

**1. Architecture3**

1.1. bramwrapper\_xxx module4

1.2. conv\_ctrlr module5

1.3. ETC module6

**2. Operation Spec6**

2.1. Setting value6

2.2. Memory map7

**3. Finite State Machine9**

3.1. bramwrapper\_xxx module9

3.2. conv\_ctrlr module10

**4. Pseudo code11**

1. Architecture

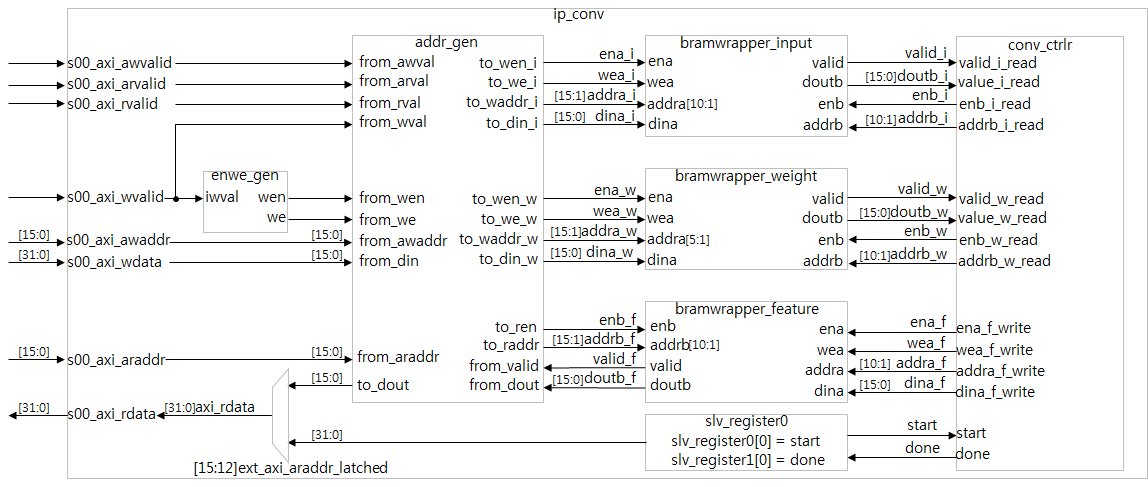


Figure 1.1 Convolution IP’s overall architecture

This page is to describe our Convolution IP’s overall architecture. In figure 1.1 ip\_conv is composed with 7 modules. The details of those modules, please see in below.

1. slv\_register0

This module is for getting information like address, ready, valid and so on from Master(ARM CORE : Zynq 7 processor).

2. addr\_gen

This module is to generate address to write information on each 4 Block ram wrapper.

3. bramwrapper\_input

This module is to store and send ‘input data’ to conv\_CRTL module.

4. bramwrapper\_weight

This module have same function with #3, the only different thing is that the data is ‘weight’ not ‘input data’.

5. bramwrapper\_feature

This module is to store the result data after convolution operation. Thus, the result data is input data from conv\_CTRL module and this data is going to send to Master through axi\_SLAVE, addr\_GEN modules.

6. conv\_ctrlr

Have same function with #3, the only different thing is data. Those data is the setting parameter such as STRIDE, PADDING, START, DONE.

1.1. bramwrapper\_xxx module

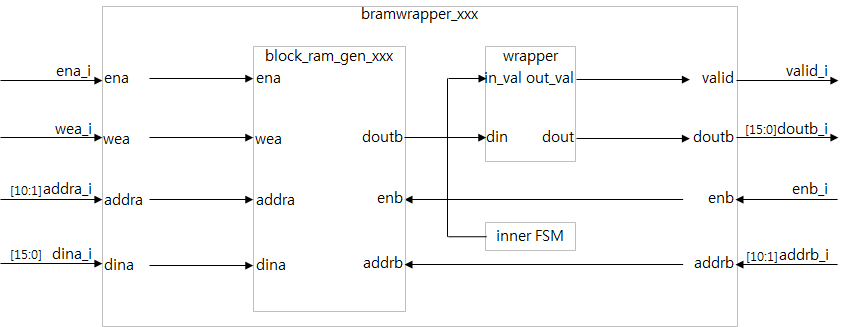


Figure 1.1.1 Convolution IP’s bramwrapper architecture

In blockram generator, we have to concern the clock latency to read our data that is saved in blockram generator. that’s why my ip have bramwrapper\_xxx(input or weight or feature) module.

This module is consisted with 3 sub modules like above Figure 1.1.1 block\_ram\_gen\_xxx(input or weight or feature), wrapper and inner FSM. The role of those 3 modules, please see the information in below.

1. block\_ram\_gen\_xxx

As a memory, the data width is 16bits and address width is depends on the ‘xxx’ naming. Thus, if ‘xxx’ is input or feature, address width is 11bits and else ‘xxx’ is weight, address width is 6bits because the condition of our ip is fixed like that the data size is 32x32, stride 1, non-padding and the filter size is 5x5 so, the feature size is going to be 28x28.

2. wrapper

As a filter to get collect data what we want. for example, if the clock latency is ‘2’ the wrapper keep that enable signal for 2clocks to get collect data.

3. inner FSM

AS a controller to operate top module. the composition is with IDLE, WAIT, READ 3states. if you want to see the more details please find it in the index #3 FInite state machine.

1.2. conv\_ctrlr module

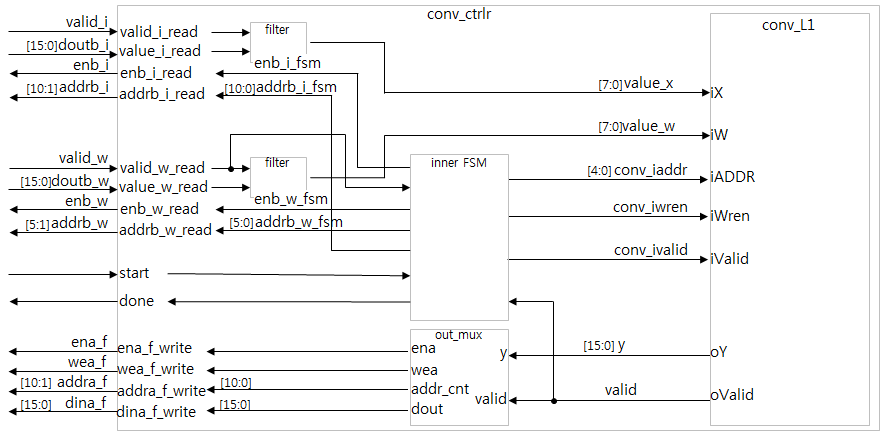


Figure 1.2.1 Convolution IP’s conv\_ctrlr architecture

This module is to control the convolution operation. like Figure 1.2.1, conv\_ctrlr module is consists of 4modules filter, inner FSM, out\_mux, conv\_L1. For the more details, please see the below informaiotn.

1. filter

The filter module is acting like data path when the valid signal is ‘high’. for example, when the valid\_iread signal is ‘low’, the value\_x is ‘0’. But if the valid\_iread signal is ‘high’ the value\_x follow the value\_iread without above 8bits.

2. inner FSM

My Conv\_ctrlr state is consisted with 2states; IDLE and CONV. The transition condition is that the ‘start’ signal is ‘high’, the IDLE state is going to be CONV state. in this state, the conv\_L1 module is operated by CONV state signal. for the more details, please see the index #3 Finite State Machine.

3. out\_mux

To write the feature data in block ram generator. when the valid signal is ‘high’, the ena and wea are going to be ‘high’, addr\_cnt will be counted up and the dout is going to follow ‘y’ value.

4. conv\_L1

To Convolution operation, iX and iW value are from the each filters and iADDR, iWren, iValid are from the FSM and oY, oValid is to out\_mux module. For the more details, please see my the other design document ‘conv\_L1’.

1.3. ETC module

1. enwe\_gen

To generate en and we signal to act bram wrapper.

2. muxing for the s00\_axi\_rdata

Void the multi connection on s00\_axi\_rdata.

3. addr\_gen

For address mux, to define the block ram memory address that is based on Memory map Figure 2.2.1.

4. slv\_register 0

To send start signal from ARM core to conv\_ctrlr module. also the done signal have a same operation with start signal.

2. Operation Spec

2.1. Setting value

① Stride : 1 or 3

② Padding : 0

③ Filter size : 5

④ Input size : 32

⑤ Feature size : 28

※ Feature size is depend on #1~#3 specs. For example, the Stride ‘1’, Padding ‘OFF’, Filter size ‘5’ the Feature size is going to be ‘28’. The feature size formula like Figure 2.1.1

Figure 2.1.1 Feature size formula

2.2. Memory map

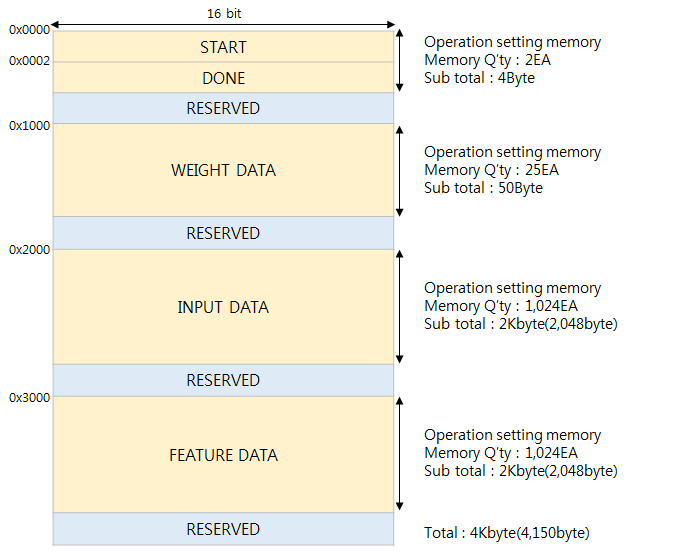


Figure 2.2.1 Memory map and required memory

See the Figure 2.2.1 There is all of the memory allocation information and memory size. In our IP, we use 16bit fixed point. Thus, the data width is ’16bit’.

The details like in below.

① 0x0000 ~ 0x0002 : Start value; 1EA 2Byte.

② 0x0002 ~ 0x0004 : Done value; 1EA 2Byte.

③ 0x1000 ~ 0x1020 : Weight data value; 25EA 50Byte.

④ 0x2000 ~ 0x2200 : Input data value; 1,024EA 2Kbyte(2,048byte).

⑤ 0x3000 ~ 0x3200 : Feature data value; 1,024EA 2Kbyte(2,048byte).

Total 4Kbyte(4,150Byte) is required.

3. Finite State Machine

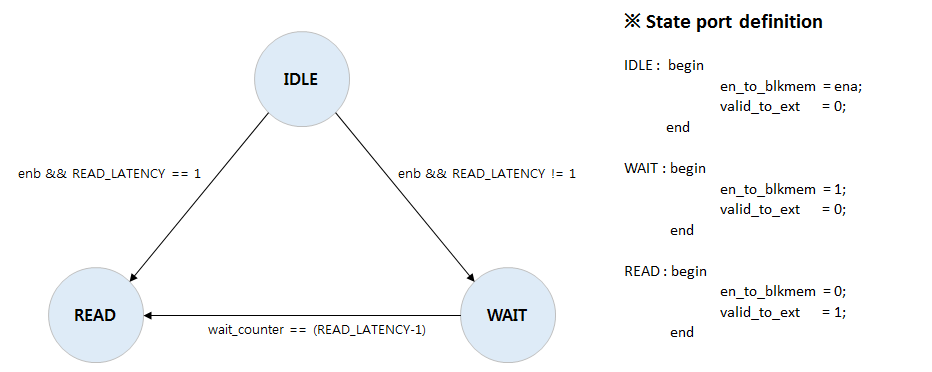
3.1. bramwrapper\_xxx module

Figure 3.1.1 Finite State Machine of bramwrapper\_xxx module

See the Figure 3.1.1 there are 3 states for operation of bramwrapper\_xxx module. the more details like in below.

① IDLE state : when enb is ‘high’ and READ\_LATENCY == 1, the next state will be READ state. And en\_to\_blkmem signal is going to follow ena data and valid\_to\_ext signal will be ‘low’.

② WAIT state : This state is for sustain enable signal until wailt\_counter value is same with READ\_LATENCY. Thus, the transition condition to READ state is wait\_counter == (READ\_LATENCY-1) and en\_to\_blkmem signal is going to be ‘high’ and valid\_to\_ext signal is still ‘low’.

③ READ state : The valid\_to\_ext value is ‘high’ in this state Therefore user can get the memory value from the block memory. And en\_to\_blkmem is changed from ‘high’ to ‘low’. And the next state will be IDLE.

3.2. conv\_ctrlr module

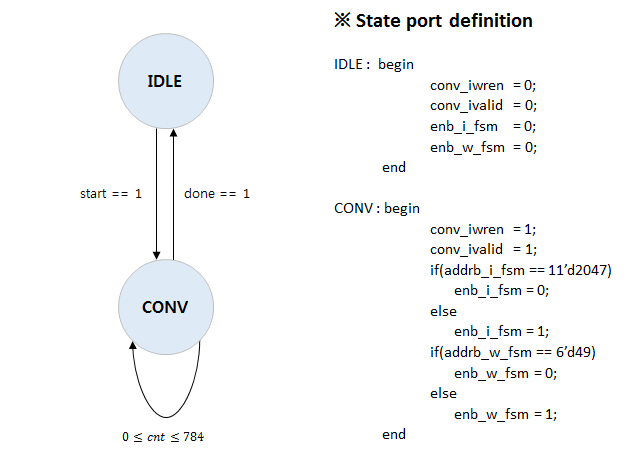


Figure 3.2.1 Finite State Machine of conv\_ctrlr module

See the Figure 3.2.1 there are 2 states for operation of conv\_ctrlr module. The more details like in below.

① IDLE state : IDLE state function is literally ‘IDLE. Thus, all of signals that is related with valid to operate the other modules are ‘Low’. And if the start signal is ‘high’ the state is transited to CONV state.

② CONV state : Until the done signal is not ‘high’, the conv\_L1 module in Figure 1.2.1 is received all data such as x, w, valid and so on. And if the inner counter ‘cnt’ is going to be 785, the done signal is transited to IDLE state.

4. Pseudo code

localparam IDEL = 2'b00,

SET = 2'b01,

WXREG = 2'b10,

CONV = 2'b11;

always @(posedge iclk, negedge irst) begin

if(!irst)

cState <= IDLE;

else

cState <= nState;

end

always @(\*) begin

case (cState)

IDLE : if(sRead)

nState = SET;

else

nState = IDLE;

SET : if(wxRead)

nState = WXREG;

oenb\_sREAD = 1;

oaddtb\_sREAD = 0 ~ 8;

else(Done)

nState = IDLE;

else

nState = SET;

WXREG : if(xcnt==225^2 && wcnt==kernelsize^2-1 && wDone)

nState = CONV;

oenb\_xREAD = 1;

oenb\_wREAD = 1;

oaddrb\_xREAD = 1;

oaddrb\_wREAD = 1;

else

nState = WXREG;

CONV : if(ccnt==((224-kernelsize+2\*padding)/stride + 1))

nState = SET;

else

nState = CONV;

conv C0 #(.stride)(clk, rst, x, w, FEATURE);

oena = 1;

owea = 1;

oaddra = 0 ~ 50,126;

odina = FEATURE;

endcase

end