Vivado HLS tool Report

Ahmed yahia

Advantages that we can use

- The tool interface is very easy
- We can write c++ code and turn it into Verilog
- Through simulation wave forms can be obtained
- We can target zynq fpga directly
- Specify the type of memory easily (FIFIO ,ROM ,BRAM , RAM)
- Using directives we can easily pipeline the codes
- Gives estimations to frequency , clock cycles and resources utilization
- Compare different solutions at time
- Test the generated code against the c++ testbench
- Useful flags inside the Verilog that help understanding the operation

Disadvantage

- The c++ have some limitations (no dynamic arrays -recursion – system command)
- The generated code is not easily readable due using pragamas for targeted devices and the Fsm generated

Tested projects

A. Matrix multiplication

CODE :

```
#ifndef __MATRIXMUL_H__
#define __MATRIXMUL_H__
#define matrix_row1 3
#define matrix_col1 3
#define matrix_row2 3
#define matrix_col2 3


void matrixmul(
    int matrix1[matrix_row1][matrix_col1],
    int matrix2[matrix_row2][matrix_col2],
    int result[matrix_row1][matrix_col2]
)
{


    for( int i = 0 ; i < matrix_row1;  i++)
    {
        for (int j = 0 ; j < matrix_col2 ; j++)
        {
            result[i][j] = 0;
            for(int k = 0 ; k < matrix_row2; k++)
            {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }

    }
}

```

Synthesized code output report :

Synthesis Report for 'matrixmul'

General Information

Date: Sat Jun 6 21:52:10 2020
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: matrixmul3
Solution: solution1
Product family: zynq
Target device: xc7z020-clg484-1

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.702 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
79	79	0.790 us	0.790 us	79	79	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-
Expression	-	-	0	115	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	69	-
Register	-	-	44	-	-
Total	0	1	44	184	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0

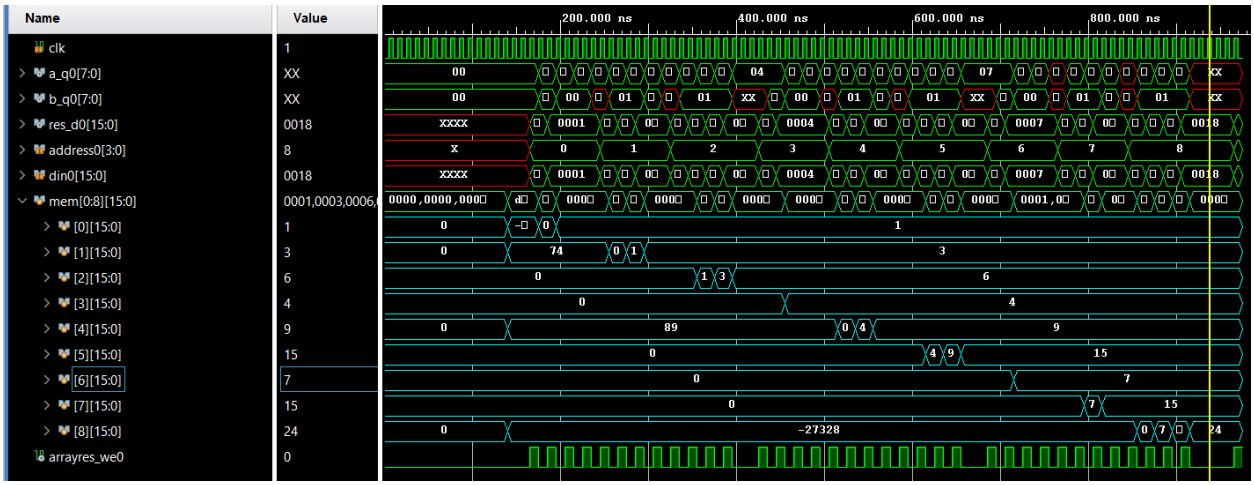
Detail

Interface

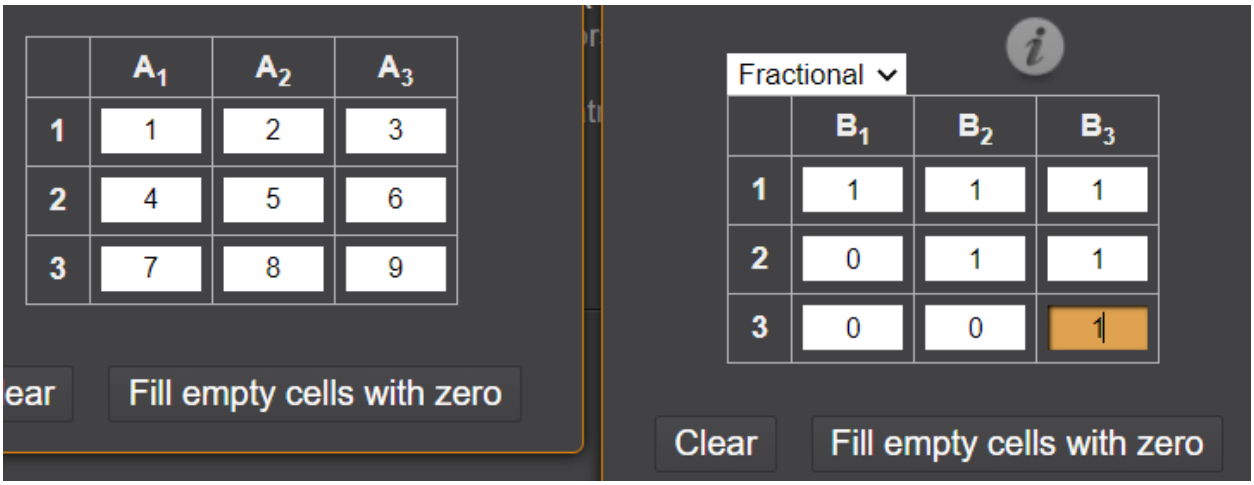
Summary

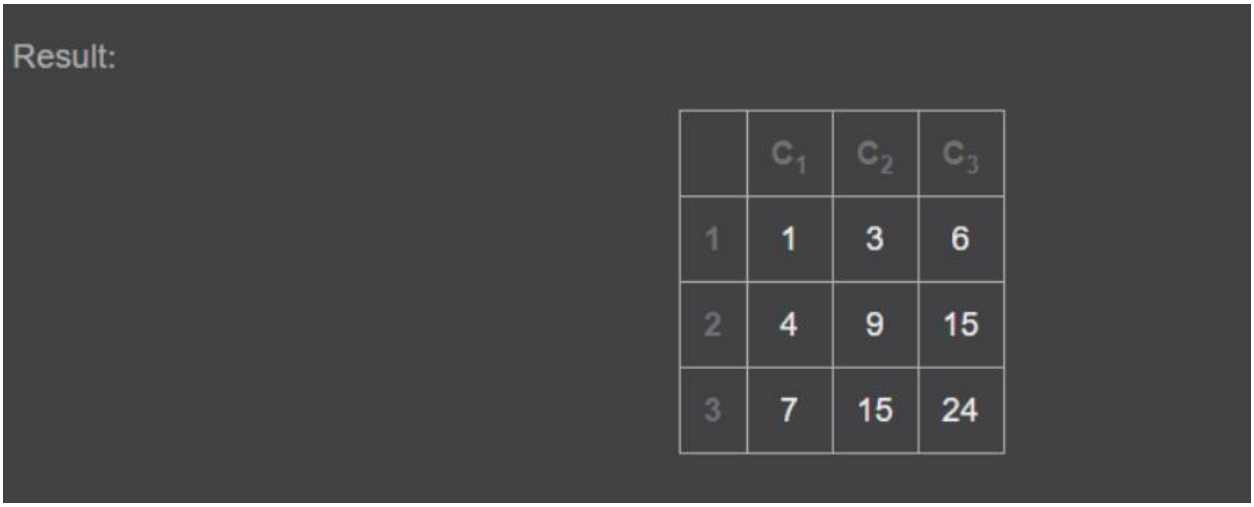
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	matrixmul	return value
ap_rst	in	1	ap_ctrl_hs	matrixmul	return value
ap_start	in	1	ap_ctrl_hs	matrixmul	return value
ap_done	out	1	ap_ctrl_hs	matrixmul	return value
ap_idle	out	1	ap_ctrl_hs	matrixmul	return value
ap_ready	out	1	ap_ctrl_hs	matrixmul	return value
a_address0	out	4	ap_memory	a	array
a_ce0	out	1	ap_memory	a	array
a_q0	in	8	ap_memory	a	array
b_address0	out	4	ap_memory	b	array
b_ce0	out	1	ap_memory	b	array
b_q0	in	8	ap_memory	b	array
res_address0	out	4	ap_memory	res	array
res_ce0	out	1	ap_memory	res	array
res_we0	out	1	ap_memory	res	array
res_d0	out	16	ap_memory	res	array

Simulation screenshot



The memory in the aqua is output array that has the res_d0 saved in it .It shows clearly that it is the expected output. By tracing the wave form of inputs and the result the timing of every thing goes as expected .the testbench of matrices are





Notes :

-pipelining was used to reduce latency and clock cycles but increased resources

On many levels the inner for loop and the outer one

B.CONVOLUTION

CODE :

```
#define image_size 5 //final size = 5*5=25
#define kernel_size 3
#define conv_op_size image_size-kernel_size+1
void convo ( int input_image[image_size][image_size],
             int kernel[kernel_size][kernel_size],
             int conv_op[conv_op_size][conv_op_size]);

void convo ( int input_image[image_size][image_size],
             int kernel[kernel_size][kernel_size],
             int conv_op[conv_op_size][conv_op_size]){


     for (int i=0; i< conv_op_size; i++){
         for (int j=0; j< conv_op_size; j++){
             int irow= i;
             int  icol= j;
             int sum=0;
             int mul;
                     for (int krow=0; krow<kernel_size ; krow++){
                         for (int kcol=0; kcol<kernel_size ;kcol++){

                             mul= kernel[krow][kcol]*input_image[irow][icol];
                             sum += mul;
                             if(icol == (j+kernel_size-1)){
                                 irow++ ;
                                 icol=j;
                             }
                             else{
                                 icol++;
                             }
                         }
                     }
                conv_op[i][j]= sum;
         }
     }
}

```

Synthesized code output report :

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.035 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
322	322	3.220 us	3.220 us	322	322	none

Detail

Utilization Estimates

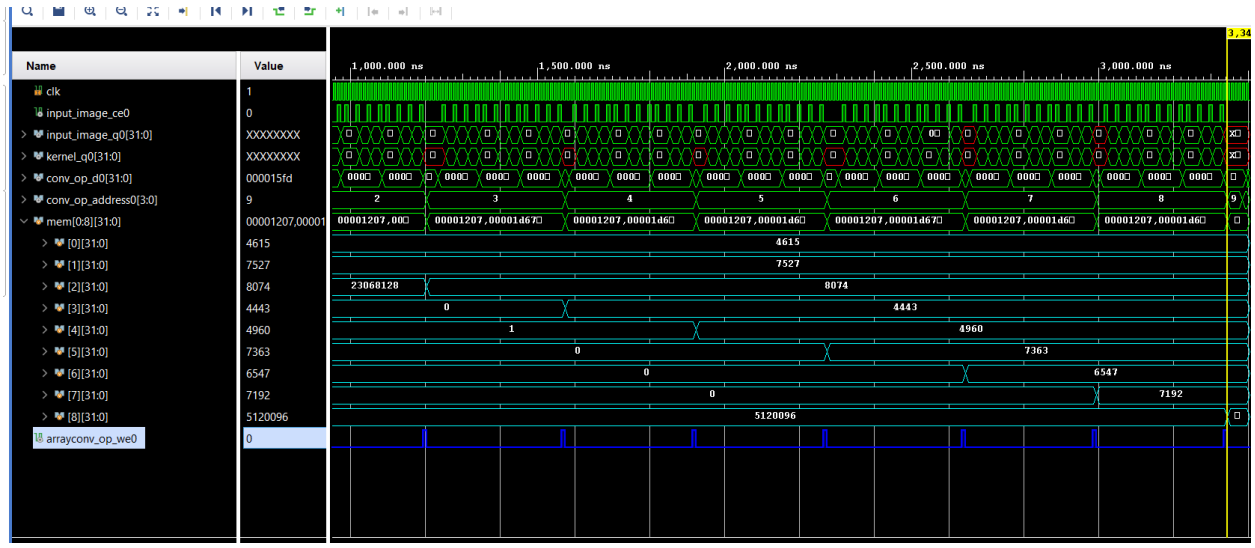
Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	3	0	379	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	131	-
Register	-	-	328	-	-
Total	0	3	328	510	0
Available	2060	2800	607200	303600	0
Utilization (%)	0	~0	~0	~0	0

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	convo	return value
ap_rst	in	1	ap_ctrl_hs	convo	return value
ap_start	in	1	ap_ctrl_hs	convo	return value
ap_done	out	1	ap_ctrl_hs	convo	return value
ap_idle	out	1	ap_ctrl_hs	convo	return value
ap_ready	out	1	ap_ctrl_hs	convo	return value
input_image_address0	out	5	ap_memory	input_image	array
input_image_ce0	out	1	ap_memory	input_image	array
input_image_q0	in	32	ap_memory	input_image	array
kernel_address0	out	4	ap_memory	kernel	array
kernel_ce0	out	1	ap_memory	kernel	array
kernel_q0	in	32	ap_memory	kernel	array
conv_op_address0	out	4	ap_memory	conv_op	array
conv_op_ce0	out	1	ap_memory	conv_op	array
conv_op_we0	out	1	ap_memory	conv_op	array
conv_op_d0	out	32	ap_memory	conv_op	array

Simulation screenshot



The simulation shows that convolution process is carried out as intend in the c++ code and below output is output of c++ testbench and the output exactly the same. Noting that the we0 (write signal in the output memory (array)is high only when the correct result signals has arrived can be specified by the clock period during project)

```
input_image matrix
11 21 17 5 13
17 12 5 12 4
9 15 37 8 20
39 19 14 10 18
26 38 47 31 7
Kernel matrix
44 26 2
33 39 46
41 1 35
conv_op matrix
4615 7527 8074
4443 4960 7363
6547 7192 5629
```

C. Padding

Code

```
#define matrix_dim_input 3
#define number_ofpadding 1
#define HW_COSIM

void matrixpadding(
    int matrix_input[matrix_dim_input][matrix_dim_input],
    int matrix_paded[matrix_dim_input+number_ofpadding*2][matrix_dim_input+number_ofpadding*2]
)
{


    for( int i = number_ofpadding,m=0 ; i < (matrix_dim_input+number_ofpadding*2)-number_ofpadding;  i++,m++)
    {

        for (int j = number_ofpadding,n=0; j <(matrix_dim_input+number_ofpadding*2)-number_ofpadding ; j++,n++)
        {


                matrix_paded[i][j] = matrix_input[m][n];

        }

    }
}
```

Synthesized code output report :

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	4.644 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
25	25	0.250 us	0.250 us	25	25	none

Detail

- Instance
- Loop

Summary

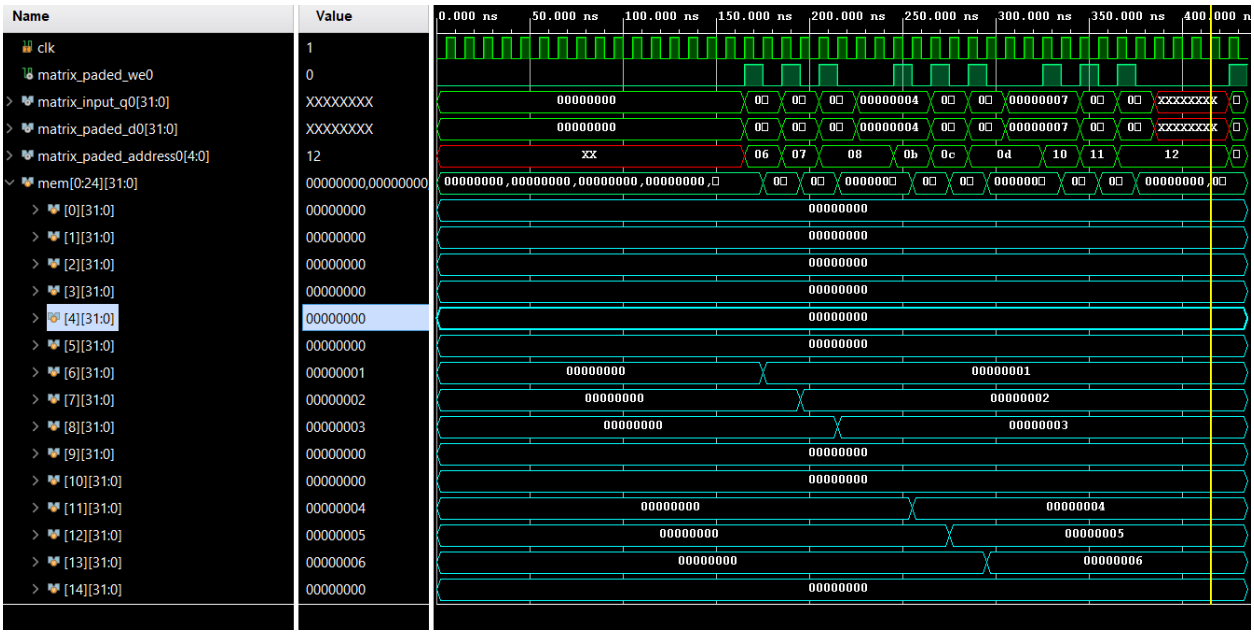
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	121	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	63	-
Register	-	-	38	-	-
Total	0	0	38	184	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	matrixpadding	return value
ap_rst	in	1	ap_ctrl_hs	matrixpadding	return value
ap_start	in	1	ap_ctrl_hs	matrixpadding	return value
ap_done	out	1	ap_ctrl_hs	matrixpadding	return value
ap_idle	out	1	ap_ctrl_hs	matrixpadding	return value
ap_ready	out	1	ap_ctrl_hs	matrixpadding	return value
matrix_input_address0	out	4	ap_memory	matrix_input	array
matrix_input_ce0	out	1	ap_memory	matrix_input	array
matrix_input_q0	in	32	ap_memory	matrix_input	array
matrix_paded_address0	out	5	ap_memory	matrix_paded	array
matrix_paded_ce0	out	1	ap_memory	matrix_paded	array
matrix_paded_we0	out	1	ap_memory	matrix_paded	array
matrix_paded_d0	out	32	ap_memory	matrix_paded	array

Simulation screenshot



Equal to c++ simulation output

```
00000
01230
04560
07890
00000INFO:
```