Report- Design, Implementation, and Comparision of Systolic Array Architectures for Convolution

Done by: Pranes S, Vishva B
108121091, 108121031

INTRODUCTION:

Using GPUs is the first thing someone will consider while handling AI and ML operations. But due to more on-chip memory of FPGA, its latency is low compared to GPUs and is power efficient, and is also flexible. But it's popular to use GPUs for deep learning computations because GPUs are easy to use whereas Deep learning with FPGA demands the knowledge of both Machine learning and FPGA.

It is well known that matrix multiplication and convolution are the necessary operations to undergo CNN. This report briefly discusses the implementation of different systolic array architectures designed to undergo convolution and matrix multiplication.

CONVOLUTION:

It is the operation of inverting the given 1-D array, shifting it from – infinity to +infinity, and multiplying it with Another 1-D array.

To undergo this convolution, we use a matrix multiplication approach. To convolve two 1-D arrays, we construct two matrices such that multiplying them produces the result of convolution.

For example, consider two arrays …….

To undergo matrix multiplication, we construct a matrix multiplier constituting some interconnected elements referred to as Processing Elements (PE).

We use distinct PEs for different architectures(discussed in the further part).

The basic elements of a PE constitute a B-Float adder, a B-float multiplier, registers, and accumulators, and it uses pipelining (which helps use multiple Pes at the same time)

B-Float multiplier and adder

Processing Element :

The components of a PE include

1)B-float adder

2)B-float multiplier

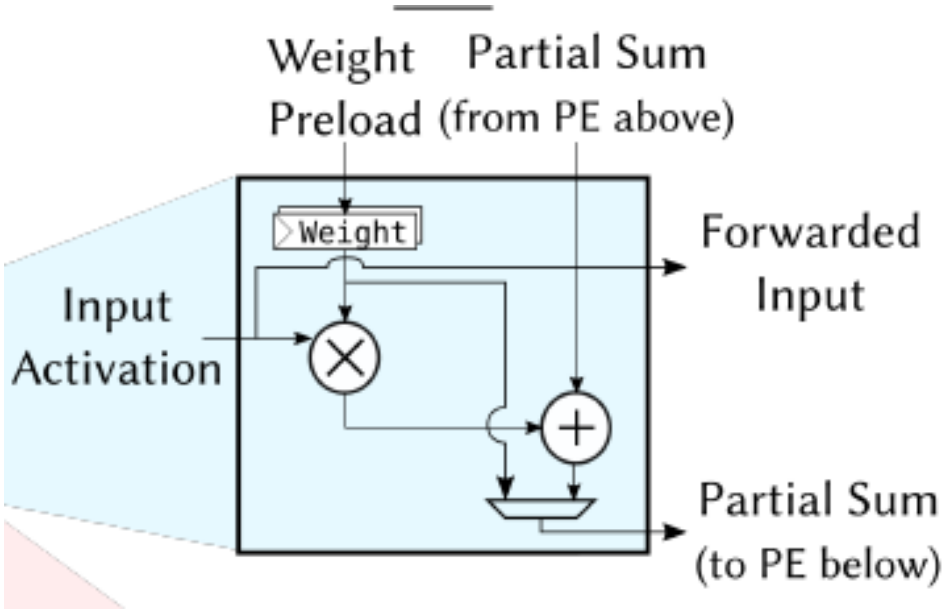
3)Reg a,b,c

A PE has two inputs and two outputs. The two inputs are two b-float numbers.

These inputs are fed to the reg a and reg b

The outputs from reg a and reg b are sent out as the two outputs of the PE and at the same time are given as inputs of the multiplier.

Then the product is sent to the b-float adder.



The two inputs of the b-float adder are the product from the multiplier and the output from register c.

Now the sum obtained from the adder is fed to the register c creating an accumulator(reg c).

Now to form a basic systolic array, the outputs of the Pes are connected to the inputs of the adjacent array as shown in fi.

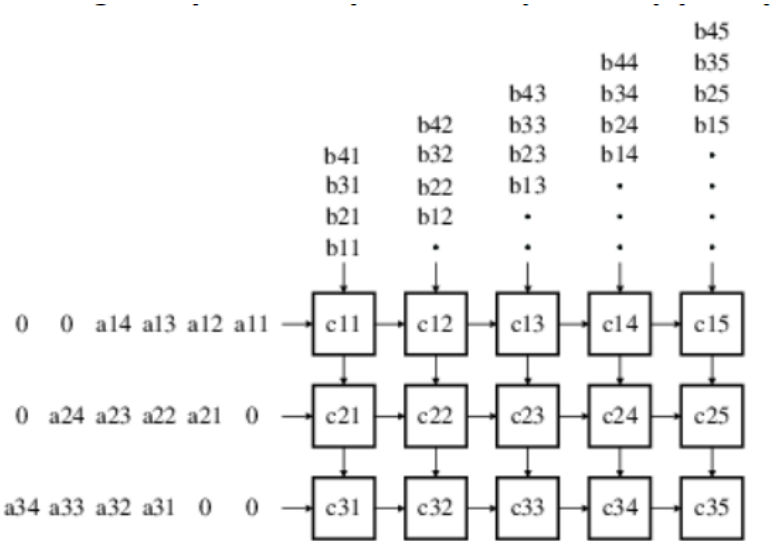
To perform matrix multiplication, the inputs of this systolic array are the rows and columns of the matrix.

Input matrices A and B.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix},$$

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} \end{pmatrix},$$

The inputs are padded in this way before injecting into the systolic array



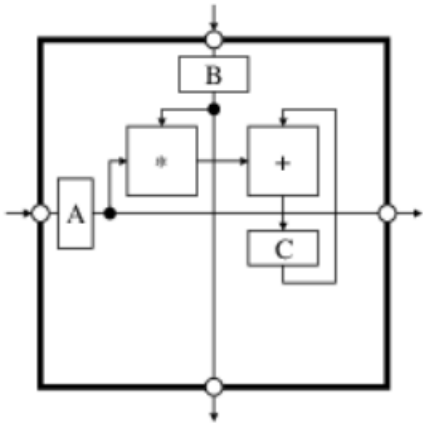
And when the inputs are padded and given in this way, the output of the matrix multiplication is obtained in each reg c of each PE.

Code explanation:
first, a module for the multiplier is created.

Then a module for the adder is created.

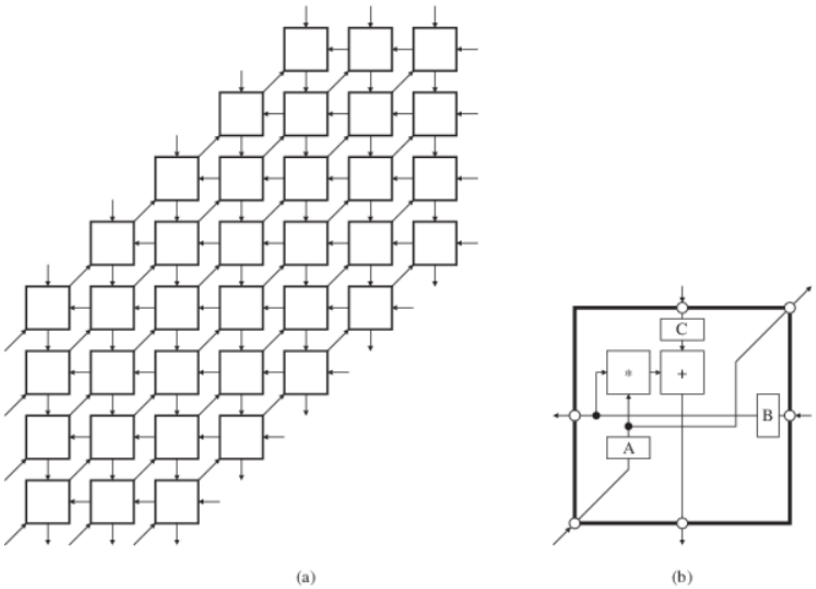
And a module for the register is also created and these modules are instantiated in a new module where each module is connected with wires to produce a PE.

The new module is created and the PE modules are instantiated and connected in such a way as to create a systolic array.



Hexagonal systolic array:

Hexagonal systolic array for matrix product. (a) Array structure and principle of the data input/output. (b) Cell structure.



A hexagonal systolic array is used in many architectures. It has three inputs and three outputs. At a particular clock pulse inputs a and b come in via register and get multiplied. On the other hand side input c enters the PE at the same time. Thus, finally, the multiplied output and the input c gets added and give the output which will be stored in the c register of adjacent(bottom) PE and it will be delivered at the next clock pulse. and the inputs a and b will enter the adjacent PE in the next clock pulse.

AS1 architecture:

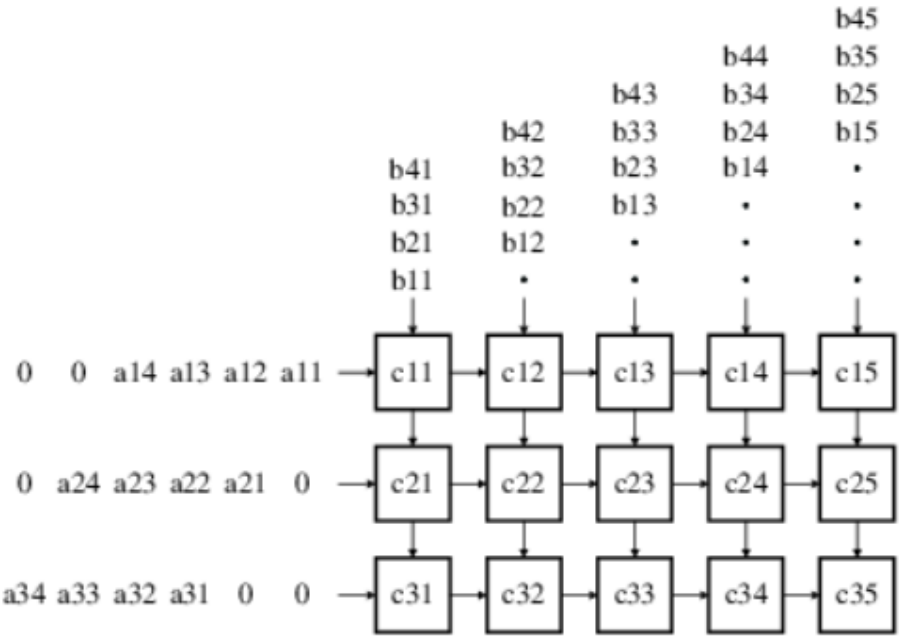
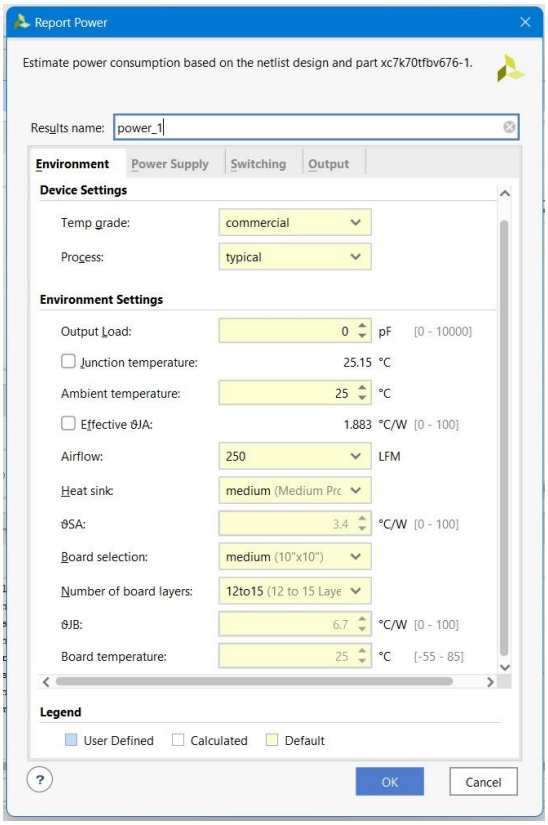


Figure 3: AS1 Architecture

AS1 architecture has two inputs(horizontal, and vertical) and two outputs(horizontal, and vertical). This architecture is the best method to find matrix multiplication and convolution. Firstly, the processing element(PE) for this architecture is designed. The systolic array receives inputs in the manner depicted in Figure 3. Using this PE, a systolic array for a 2*2 matrix(i.e. multiplication of two different 2*2 arrays – a and b) is been designed and thus the 2*2 matrix can be scaled to find the convolution of the m*n array. A Rectangular systolic array is designed for this architecture. Four PE is required for a 2*2 array and thus it can be expanded to an m*n matrix and can be used for convolution. The final output will be delivered at the fourth clock pulse.

Code Explanation:

A module is created for PE, where the module of the registers, multiplier, and adder is been instantiated. A separate module is created and the PE module is been instantiated. Since four PE is necessary for a systolic array of 2*2 matrix, the PE module is been instantiated four times. The output is depicted in Figure 3a. Inputs are given to the code by a testbench code. Output is shown in Figure 3a.

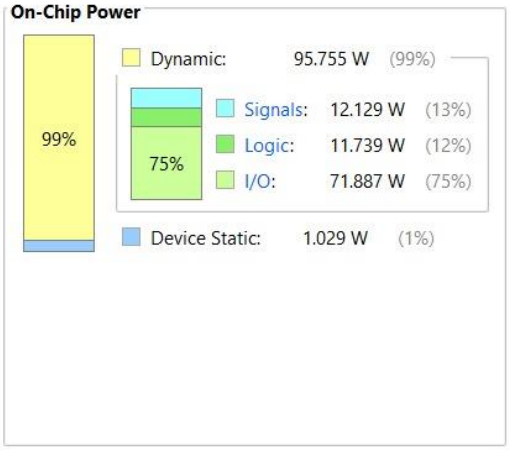


POWER REPORT

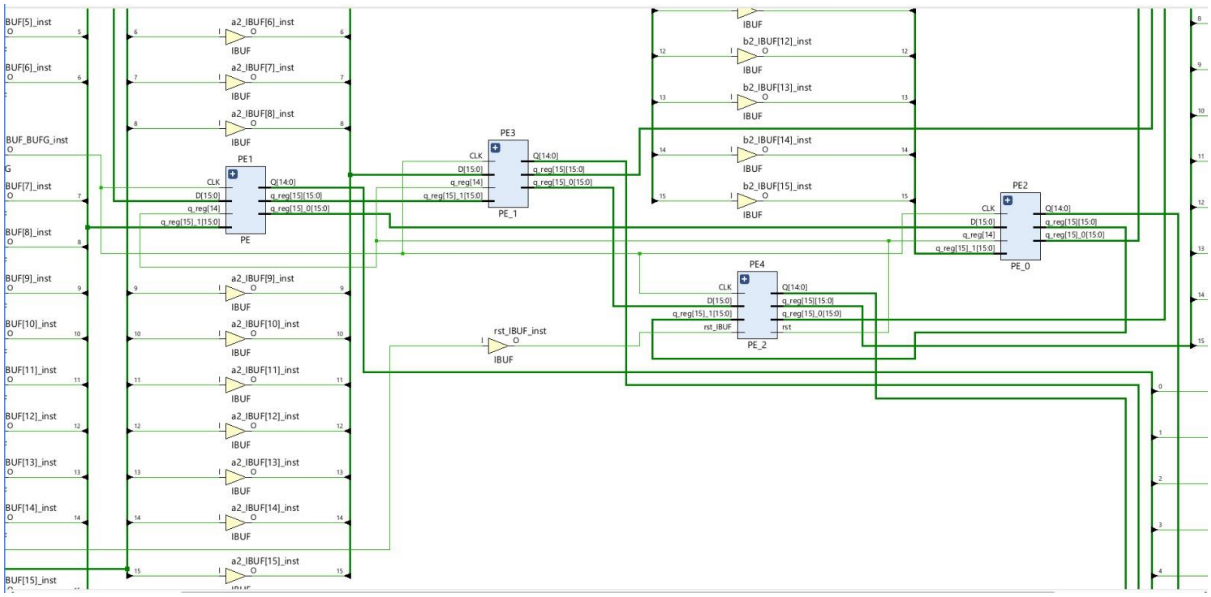
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

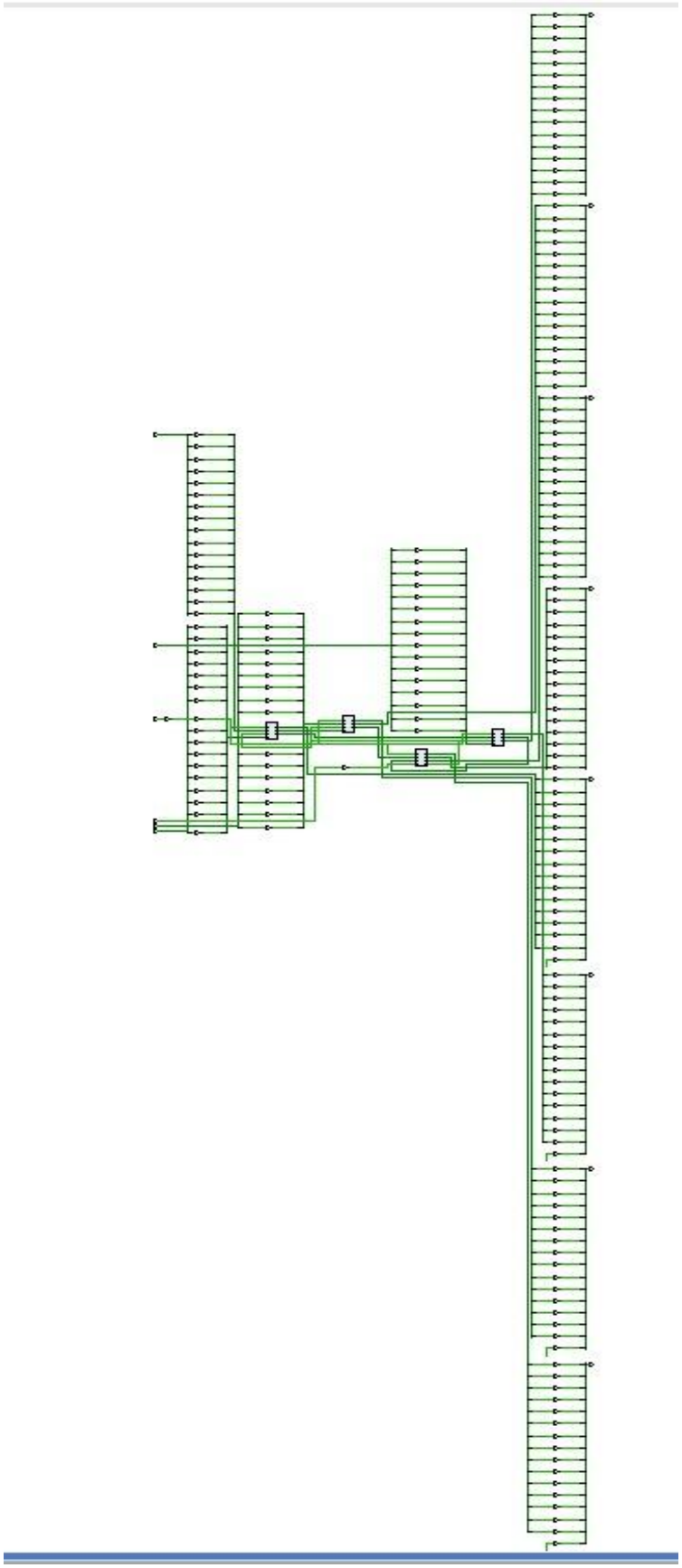
Total On-Chip Power:	**96.784 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**125.0°C**
Thermal Margin:	-122.3°C (-64.2 W)
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



POWER CONSUMPTION





SCHEMATIC LEVEL REPRESENTATION

Name	Slice LUTs (41000)	Slice Registers (82000)	Bonded IOB (300)	BUFGCTRL (32)
SYSARRAY_AB1	444	94	97	1
ab1 (AB1_arch_PE)	223	47	0	0
ab2 (AB1_arch_PE_0)	221	47	0	0

RESULTS

AS2 Architecture:

AS2 architecture has three inputs(horizontal, vertical, diagonal) and three outputs(horizontal, vertical, diagonal). It's helpful to find matrix multiplication and convolution. To obtain the result precisely, the vertical input of the first row PE is always set to zero. Firstly, the processing element(PE) for this architecture is designed. The systolic array receives inputs in the manner depicted in Figure 2a. Using this PE, a systolic array for a 2*2 matrix(i.e. multiplication of two different 2*2 arrays – a and b) is been designed and thus the 2*2 matrix can be scaled to find the convolution of the m*n array. A hexagonal systolic array is designed for this architecture. Seven PE is required for a 2*2 array and thus it can be expanded to an m*n matrix and can be used for convolution. Since the 2*2 matrix is been created, we need only seven PE(let's say PE1(0,0), PE2(1,0), PE3(0,1), PE4(1,1), PE5(2,1), PE6(1,2), PE7(2,2) in the Figure 2a). The final matrix's elements come as output from the second clock pulse. All the final output matrix(let's say C matrix) elements will be delivered only in PE 4,5,6,7. At the third clock pulse C00 (at PE 7), C01 (at PE 6), and C10 (at PE 5) come as the output. And at the fourth clock pulse, we get C11(at PE 7). Thus every element of the final matric C is been delivered within four clock pulses.

Code Explanation:

A module is created for PE, where the module of the registers, multiplier, and adder is been instantiated. A separate module is created and the PE module is been instantiated. Since seven PE is necessary for a systolic array of 2*2 matrix, the PE module is been instantiated seven times. The output is depicted in Figure 2b. . Inputs are given to the code by a testbench code.

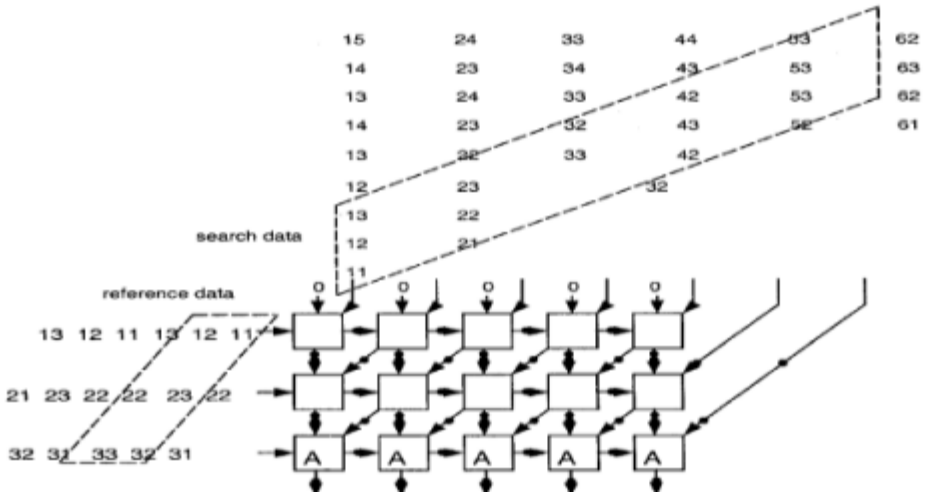


Figure 2: AS2 Architecture

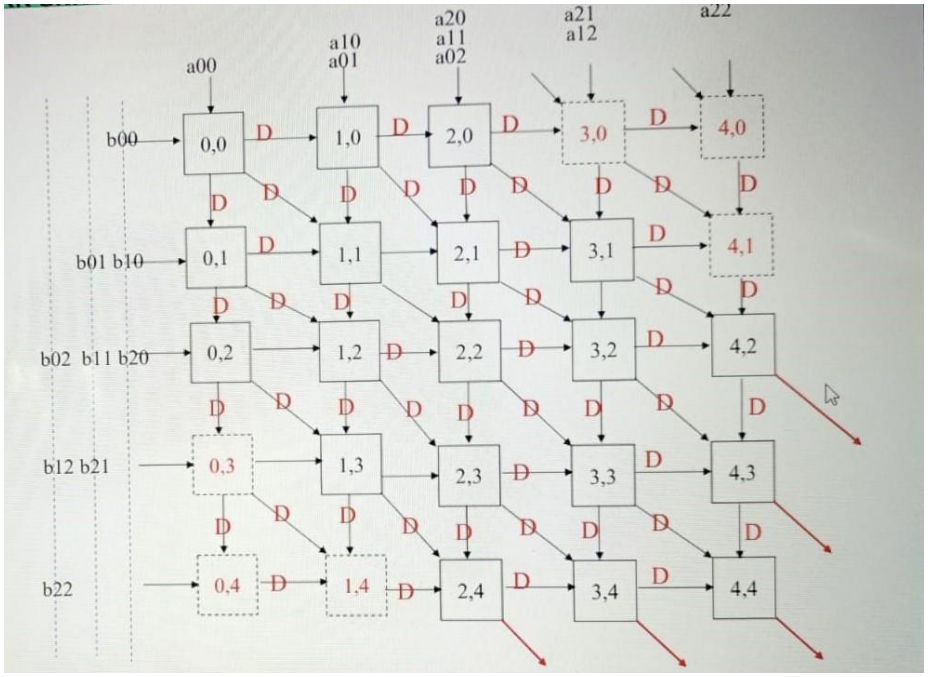


Figure 2a: Inputs given at regular clock pulses.

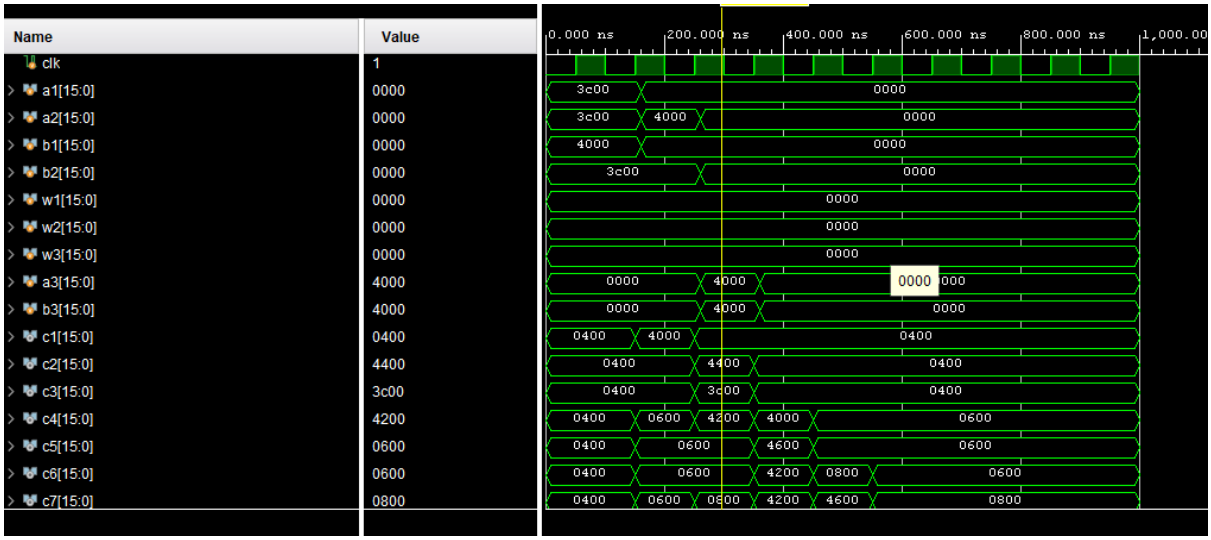
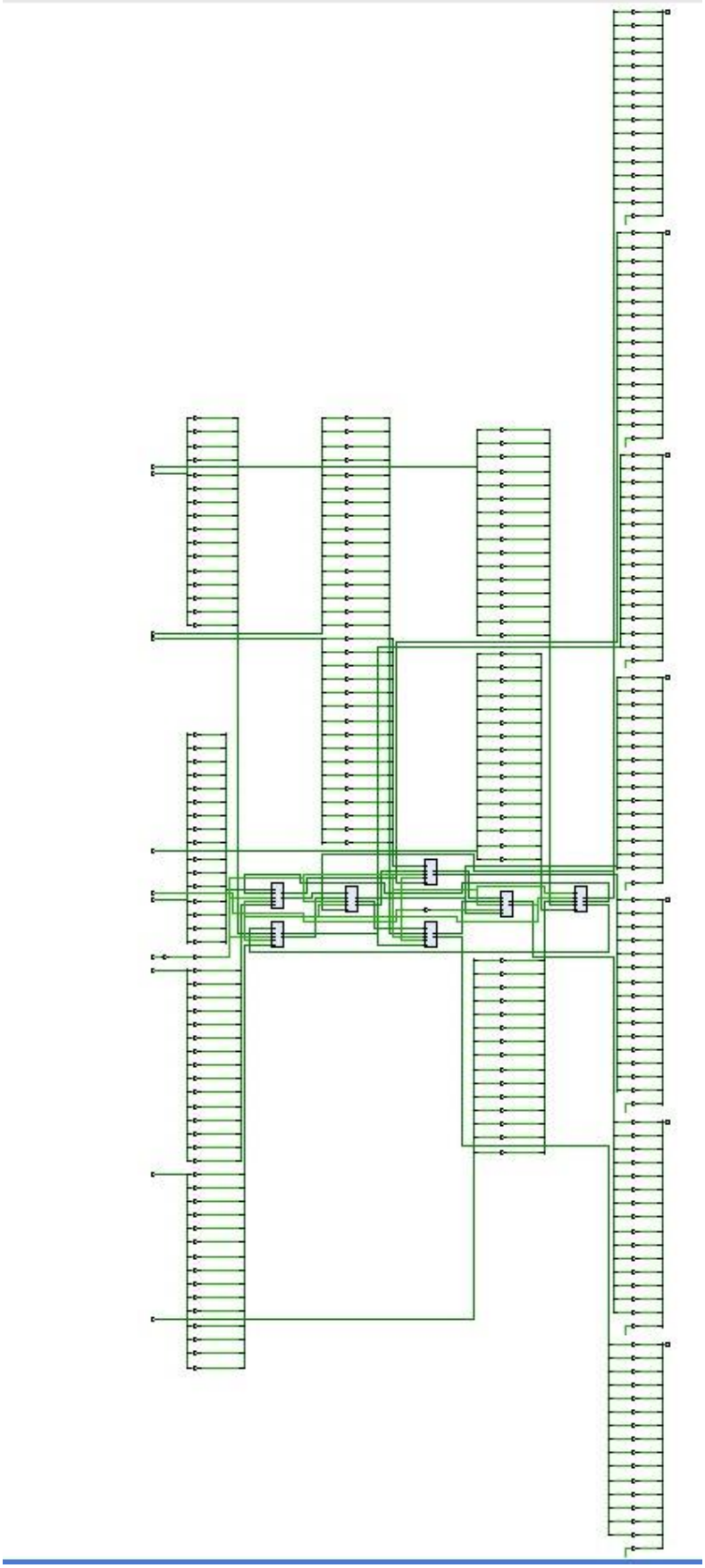
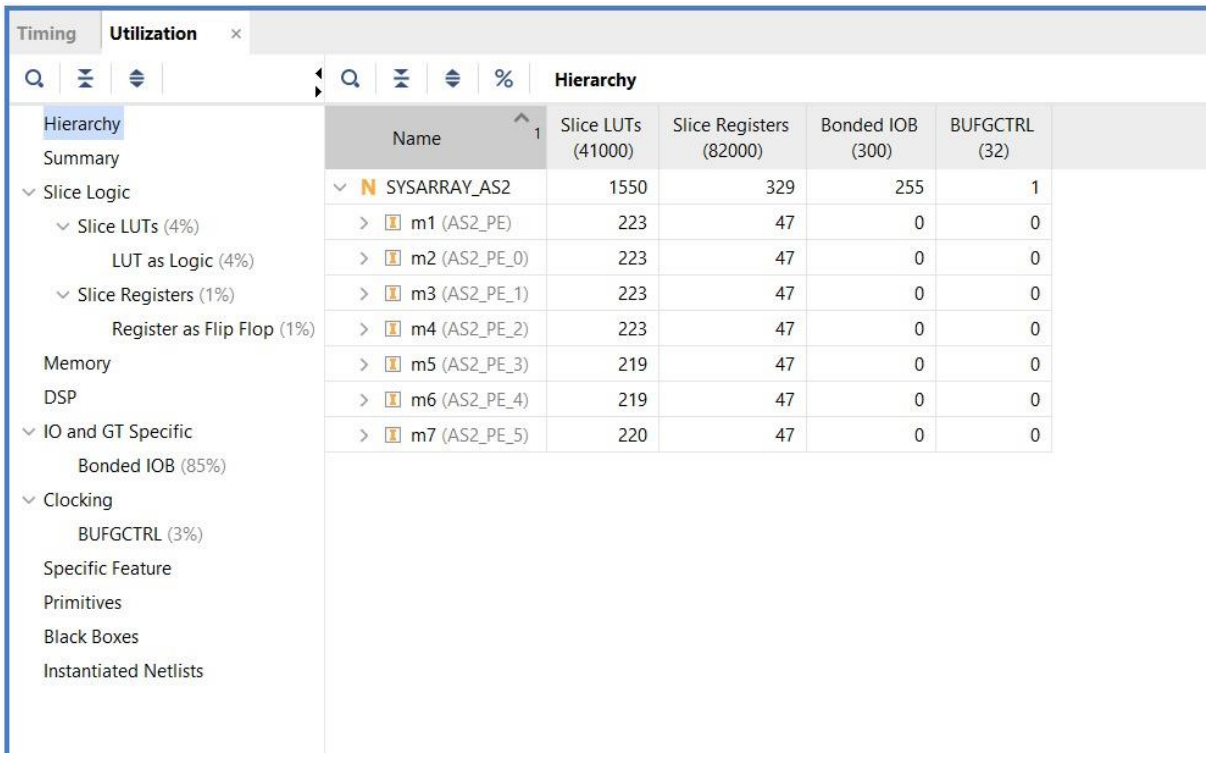


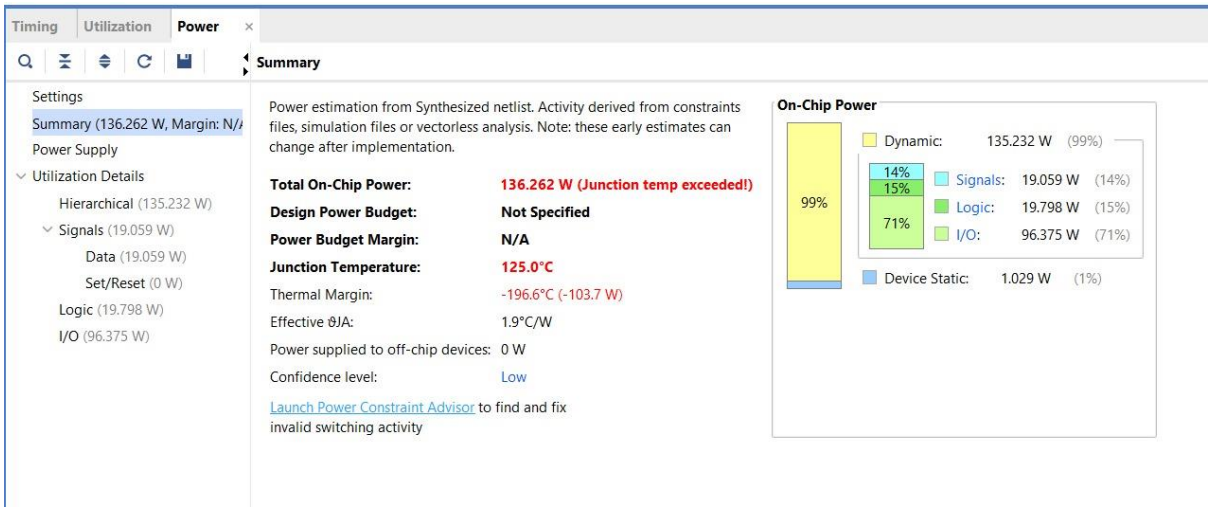
Figure 2b: Output



SCHEMATIC LEVEL REPRESENTATION



UTILIZATION RESULTS



POWER RESULTS

AB1 Architecture:

AB1 architecture is a 1-D systolic array shown in Figure 1. It is a basic architecture to find matrix multiplication and convolution. In this type of architecture, inputs(elements of an array) are given at each clock pulse and it is given at our convenience in such a way that we get the elements of the final result matrix. Firstly, the processing element(PE) for this architecture is created. This PE has three inputs and an output. Using this PE, a systolic array for a 2*2 matrix(i.e. multiplication of two different 2*2 arrays – a and b) is been designed and thus the 2*2 matrix can be scaled to find the convolution of the m*n array. Two PE is required for a systolic array of 2*2 matrix. Inputs are given to PE at different clock pulses as shown in Figure 1. The outputs show up after the 2^{nd} to 5^{th} clock pulse. Zero is a fixed input for the first PE. In the first posedge of the clock, a11 gets multiplied with b11, added with 0, and stored in the vertical input of the second PE. Simultaneously, zero is given as the input to the second PE at the first clock pulse and it gets multiplied. Finally at the second clock pulse a11*b11 comes as an input to the second PE and gets added with the multiplied inputs of the second PE i.e. a12*b21 and gives the output which is the first element of the final matrix.

Code Explanation:

A module is created for PE in which the module of the registers, multiplier, and adder is been instantiated. Since two PE is required for a systolic array, a separate module is created and the PE module is been instantiated seven times. Inputs are given to the code by a testbench code.

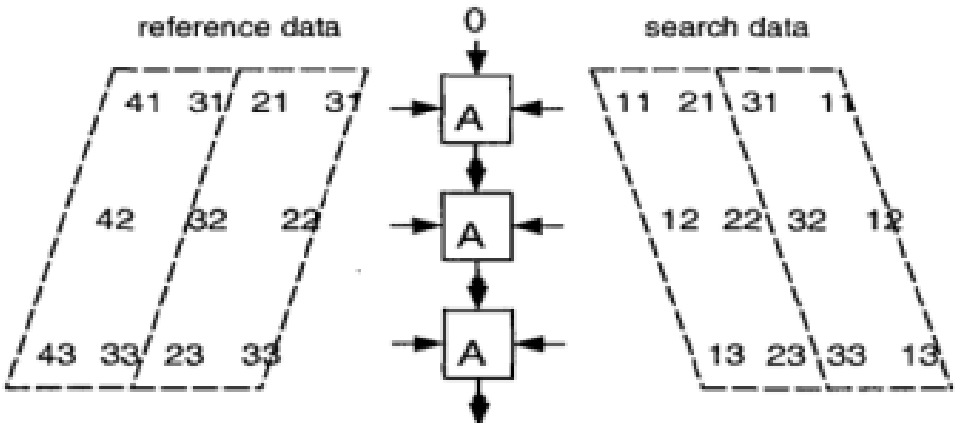
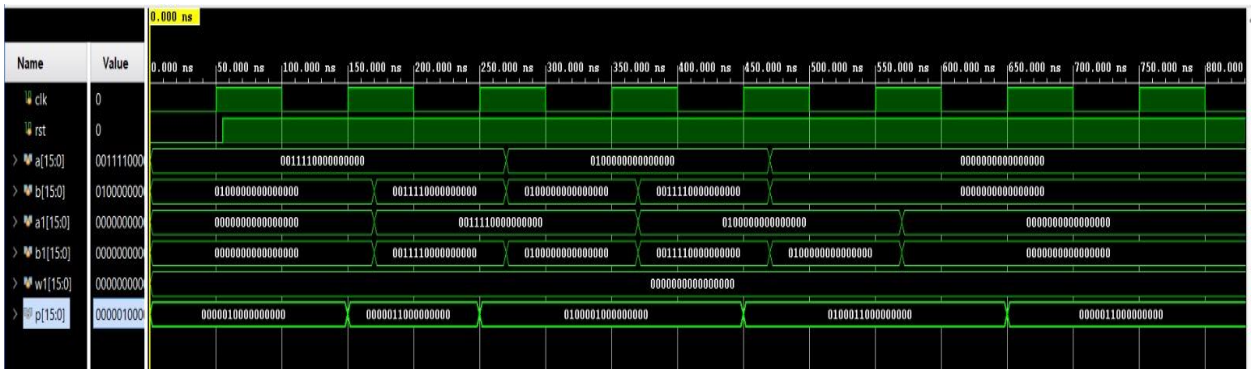
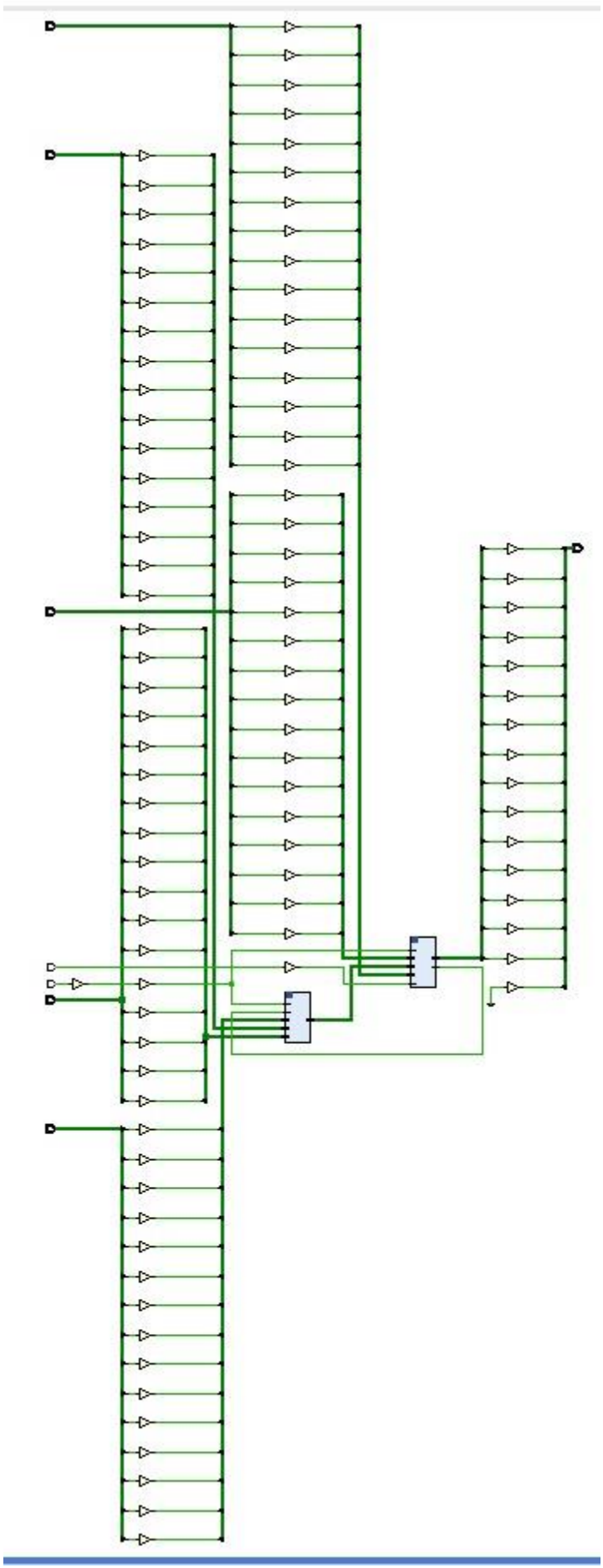


Figure 1: AB1 Architecture



OUTPUT SCREEN



SCHEMATIC LEVEL

REPRESENTATION

Name	Slice LUTs (41000)	Slice Registers (82000)	Bonded IOB (300)	BUFGCTRL (32)
SYSARRAY_AB1	444	94	97	1
ab1 (AB1_arch_PE)	223	47	0	0
ab2 (AB1_arch_PE_0)	221	47	0	0

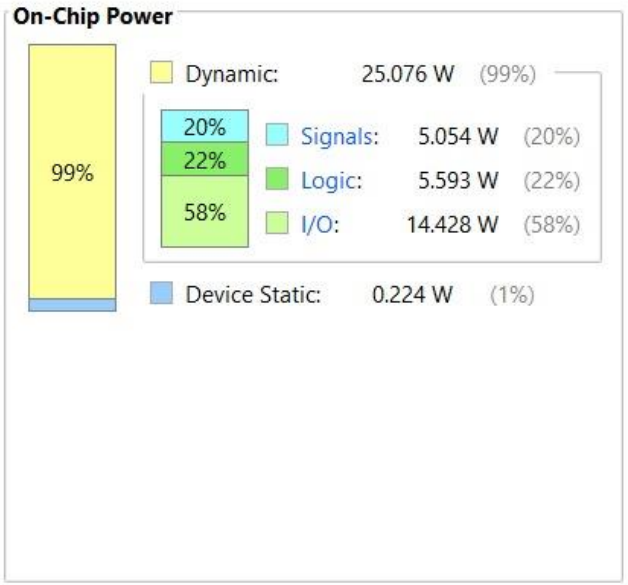
POWER CONSUMPTION REPORT

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**25.299 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**72.6°C**
Thermal Margin:	12.4°C (6.5 W)
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



POWER REPORT

Processing Element of NSA:

This PE has two inputs and two outputs in it. The components inside it are three registers, a multiplier, an adder, and a multiplexer. At the first clock pulse, inputs A and B enter PE via R1 and R2 registers and get multiplied and thus get added to the data stored in R3. This output is stored in R3 and delivered as the output. Finally, the multiplexer decides whether input B or the added value(out in Fig 4) should be the final output.

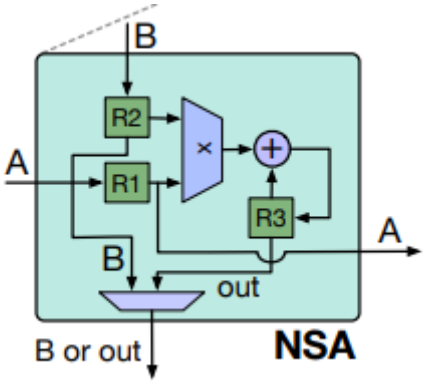
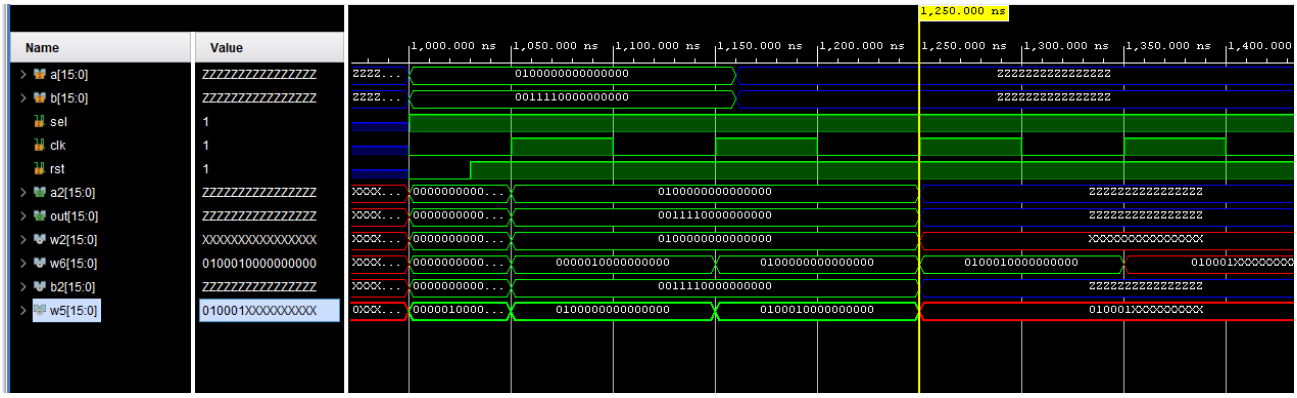


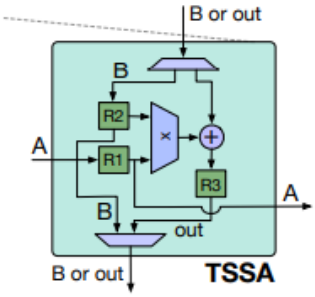
Figure 3: PE of NSA

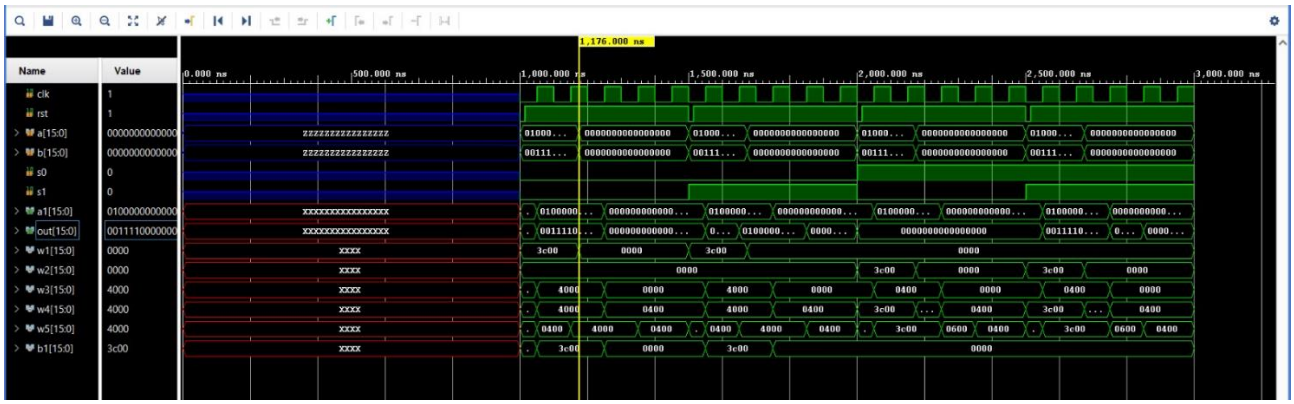


The output of NSAs' PE

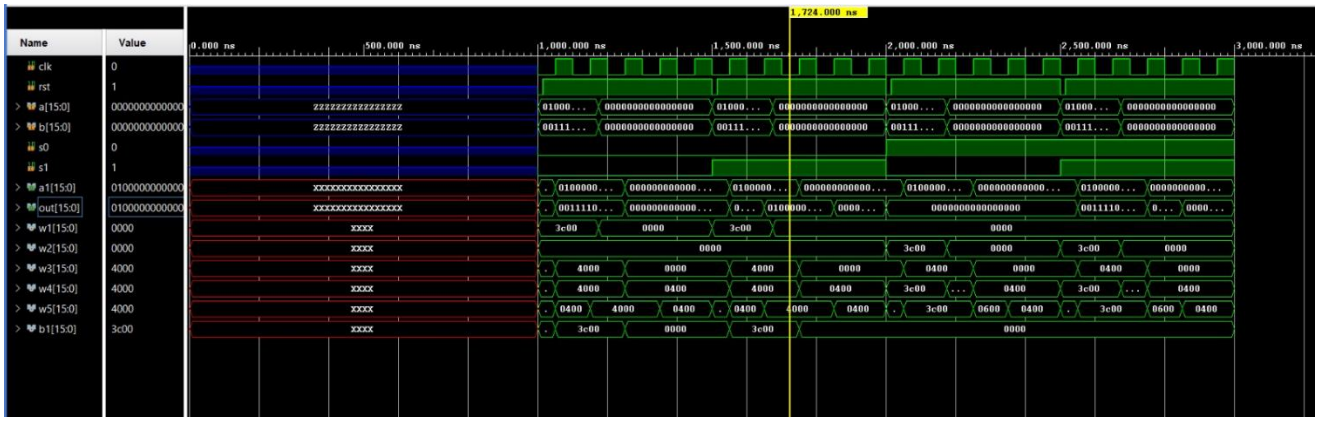
Processing Element of TSSA:

This PE has two inputs (A and B) and two outputs one of which is A itself (pipelining technique) and the other can be B or out. Out is the accumulated product of A and B. We have a MUX and a DEMUX in this PE which serves for selection. The DEMUX gives us the option to either stream the input B to register B and Multiplier or directly get into the adder. The MUX helps us to select the output type that we need. Either the accumulated product as output or the pipelined B to feed the next PE.

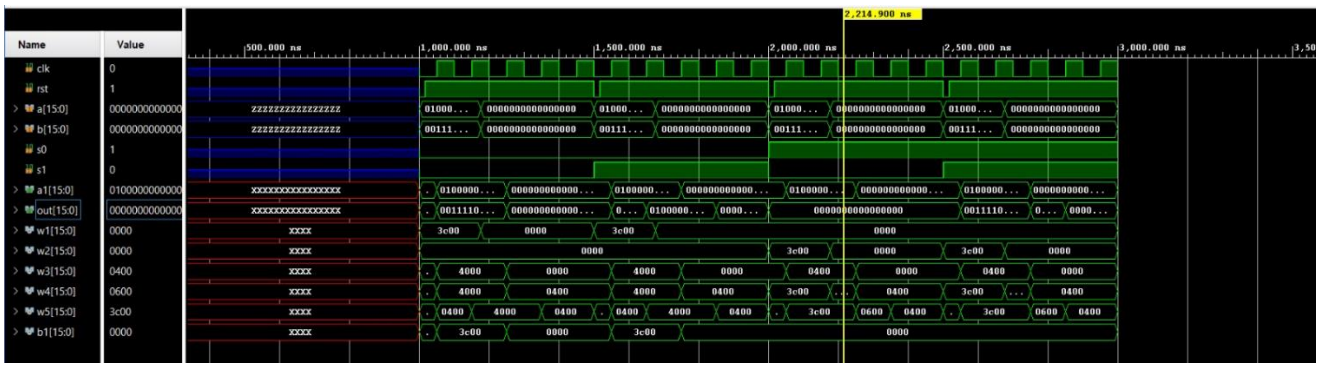




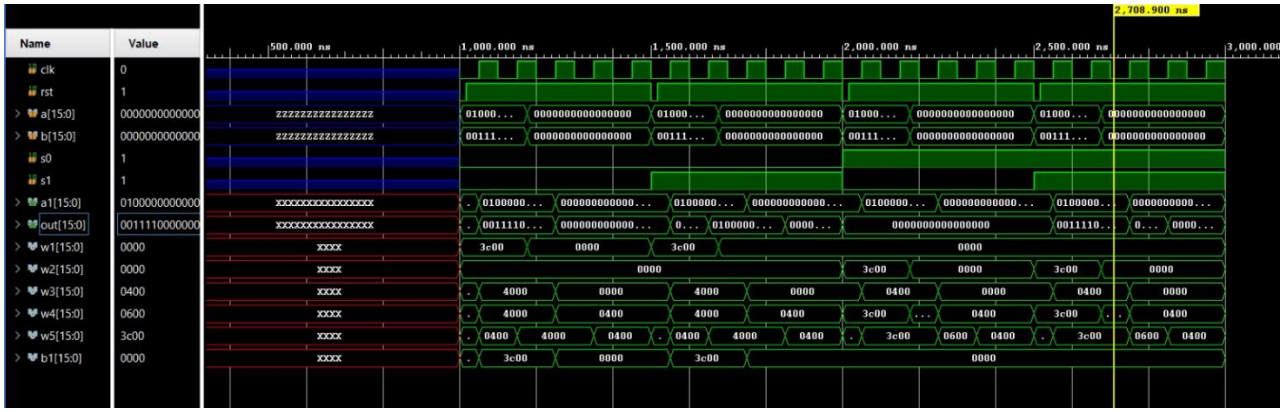
When Bothe the select lines are 0 and 0.



When Bothe the select lines are 0 and 1.



When Bothe the select lines are 1 and 0.



When Bothe the select lines are 1 and 1.