BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CAD FOR IC DESIGN

(Semester 2, Academic year 2024-25)

PROJECT-1

Synthesis of Systolic Array Multiplier using Cadence Genus

Submitted to

Dr. Abhijit Asati

By

Arun R V (2024H1230173P)
Akshay Rajendra Modak (2024H1230171P)
Naveen R G (2024H1230170P)
Yatin Arora (2024H1230161P)

Aim

Generating different versions of an 8x8 systolic array matrix multiplier by changing design constraints, each optimized for area, power and performance.

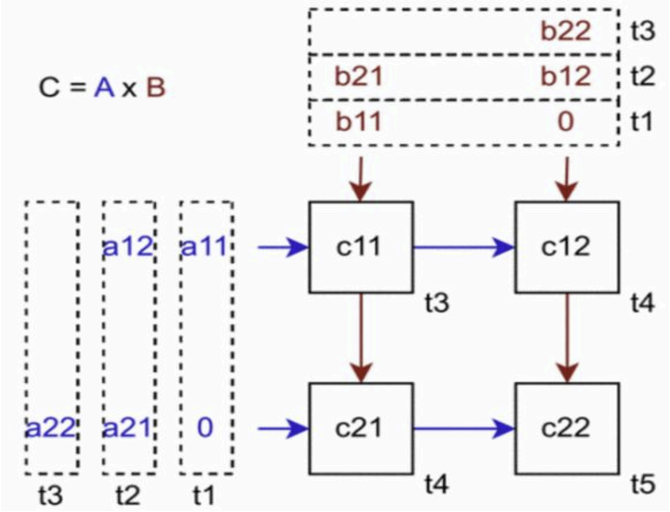
Architectural & Functional Details – Systolic Array Multiplier

Systolic arrays represent a specialized form of parallel computing architecture where data flows synchronously across a grid of processing elements (PEs). The typical systolic array for matrix multiplication consists of a 2D grid of PEs. Each PE contains:

1. **Multiplication Unit**: Performs the element-wise multiplication
2. **Accumulation Unit**: Adds the multiplication result to the running sum
3. **Storage Registers**: Holds intermediate values and control signals

Essentially, a PE is a MAC (multiply-and-accumulate) unit. For an m×n by n×p matrix multiplication using an output-stationary approach:

1. Matrix A elements enter from the left edge of the array
2. Matrix B elements enter from the top edge of the array
3. Each PE performs a multiply-accumulate operation
4. Input elements of A and B propagate rightward and downward, respectively
5. Elements of product matrix C are taken from the accumulate registers of each PE



An example of a 2x2 matrix multiplier is shown above. Each box indicates a PE and the above figure shows which element of the product matrix a particular PE holds. For the correct input elements to multiply together, every next row and column entry into the systolic array is delayed by one clock cycle. As the inputs are passed through to neighboring PEs, input elements need to be supplied only once to the systolic array, maximizing data reuse. Due to the need to provide inputs at designated time slots, a control unit is required. By default, the Genus synthesis tool uses CSA tree to implement the multiplication operation.

The multiplication is shown for a 2x2 systolic array matrix multiplier.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$$c_{11} = a_{11}b_{11} + a_{12}b_{21}$$
$$c_{12} = a_{11}b_{12} + a_{12}b_{22}$$
$$c_{21} = a_{21}b_{11} + a_{22}b_{21}$$
$$c_{22} = a_{21}b_{12} + a_{22}b_{22}$$

The cycle-by-cycle operation in the 2×2 output-stationary systolic array:

Clock Cycle 1:

- PE(1,1): Receives a_{11} and b_{11}, computes $a_{11}\cdot b_{11}$, stores in c_{11}
- Other PEs: Idle

Clock Cycle 2:

- PE(1,1): Receives a_{12} and b_{21}, computes $a_{12}\cdot b_{21}$, adds to c_{11}
- PE(1,2): Receives a_{11} and b_{12}, computes $a_{11}\cdot b_{12}$, stores in c_{12}
- PE(2,1): Receives a_{21} and b_{11}, computes $a_{21}\cdot b_{11}$, stores in c_{21}
- PE(2,2): Idle

Clock Cycle 3:

- PE(1,1): Computation for c_{11} complete ($c_{11} = a_{11}\cdot b_{11} + a_{12}\cdot b_{21}$)
- PE(1,2): Receives a_{12} and b_{22}, computes $a_{12}\cdot b_{22}$, adds to c_{12}
- PE(2,1): Receives a_{22} and b_{21}, computes $a_{22}\cdot b_{21}$, adds to c_{21}
- PE(2,2): Receives a_{21} and b_{12}, computes $a_{21}\cdot b_{12}$, stores in c_{22}

Clock Cycle 4:

- PE(1,2): Computation for c_{12} complete ($c_{12} = a_{11}\cdot b_{12} + a_{12}\cdot b_{22}$)
- PE(2,1): Computation for c_{21} complete ($c_{21} = a_{21}\cdot b_{11} + a_{22}\cdot b_{21}$)
- PE(2,2): Receives a_{22} and b_{22}, computes $a_{22}\cdot b_{22}$, adds to c_{22}

Clock Cycle 5:

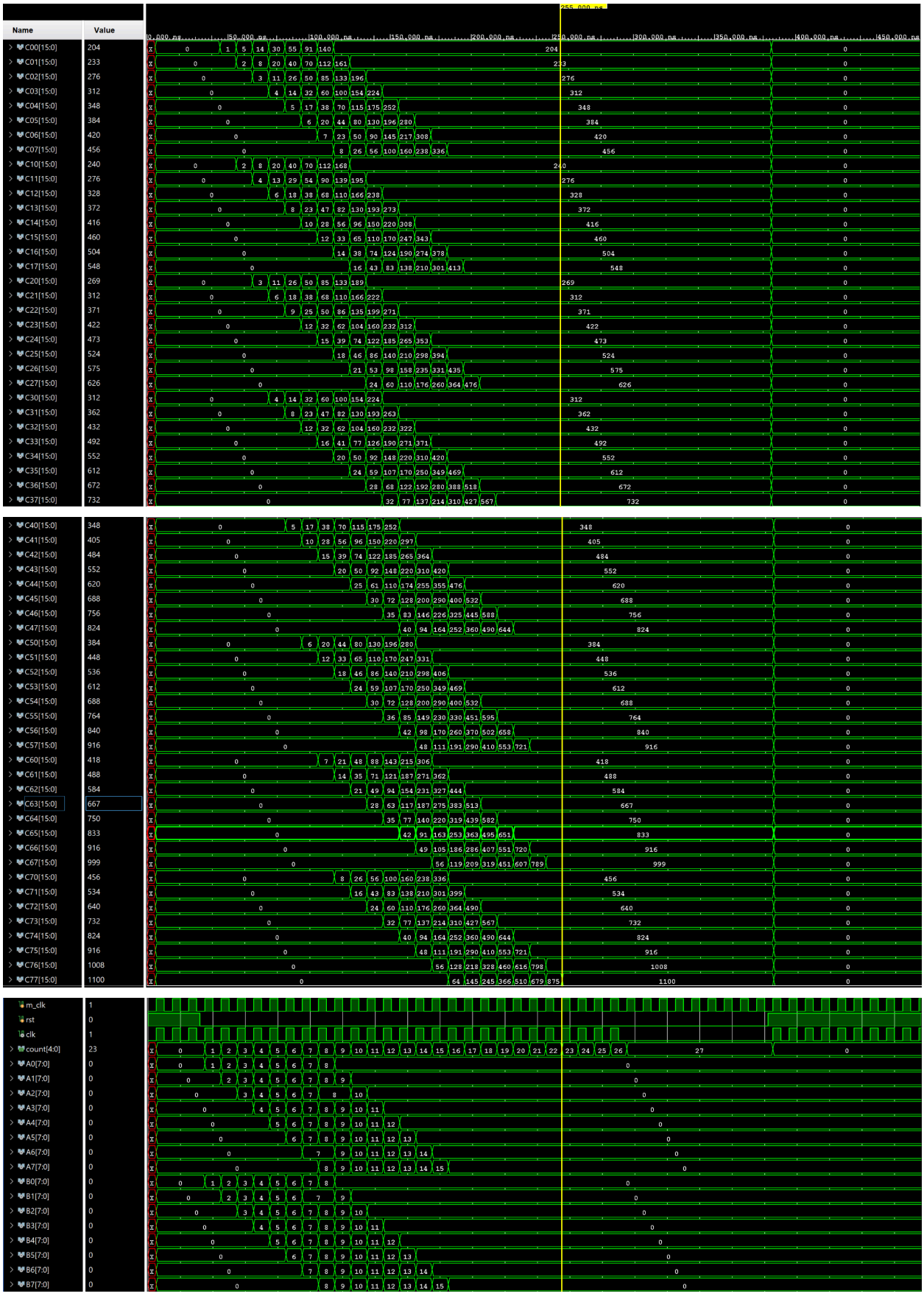
- PE(2,2): Computation for c_{22} complete ($c_{22} = a_{21}\cdot b_{12} + a_{22}\cdot b_{22}$)

Thus, the product matrix C is fully computed at the end of the 5th clock cycle. The implemented 8x8 systolic array matrix multiplier computes the product matrix in 23 clock cycles.

The same mechanism can be extended to any output-stationary systolic array matrix multiplier of any order. The main principle is that every row and column entry is delayed by one clock cycle with respect to the previous row and column entry.

Functional Verification

The functional simulation timing diagrams for an example of 8x8 matrix multiplication is below.



Version 1 – Optimized for Performance

Methodology

Changing the clock's time period also affects the synthesized design's area. This suggests that the synthesis tool selects different cells to meet the timing constraints specified in the SDC file. To achieve maximum performance, we should gradually reduce the time period in the SDC file until the slack becomes negative. Since the maximum frequency has to be found out, we use “slow.lib” as the target library for synthesis.

Results

Time Period	Data-path delay	Slack	Setup	Total Area	Total Power	Cell count
6500	4873	1384	223	148182.851	8.12E-03	15559
6000	4749	1004	226	148282.005	8.90E-03	16402
5500	5129	130	221	148658.184	9.74E-03	16939
5000	4372	386	223	150432.358	1.09E-02	17991
4500	4176	81	223	151464.77	1.25E-02	19418
4000	3730	27	223	157550.246	1.42E-02	17650
3500	3258	1	221	158666.673	1.62E-02	18264
3000	2799	0	191	164133.763	2.11E-02	23862
2500	2358	0	132	190310.393	3.08E-02	29944
2000	2218	-387	159	232618.077	5.06E-02	41192

The slack becomes negative after the time period becomes less than 2.5 ns. Therefore, this value is used to determine the maximum operating frequency, which is 400 MHz.

Version 2 – Optimized for Area

Methodology

We have used the “retime -min_area” command with a time period of 10 ns. We have also enabled clock gating which also further reduces the area. The synthesis has been performed for all three process corners (slow, typical, fast). The results for all three target libraries are shown in the table below.

Results

Corner	Total Area	Cell count	Data-path delay	Slack	Setup	Total Power
Slow	141546.352	17408	4684	5114	182	4.9E-03
Typical	137339.501	15011	2576	7309	95	7.15E-03
Fast	137336.474	14812	2038	7854	88	9.46E-03

Version 3 – Optimized for Power

Methodology

Power reduction is made possible by using clock gated cells in the design. We have also created a separate power domain for the Processing Element block of the systolic array, which when integrated inside a system with controllers can have low power consumption. Since the maximum power consumption(worst case) has to be found out, “fast.lib” was used as the target library for synthesis.

Results

	Leakage power	Internal power	Switching power	Total Power	Data-Path delay	Total Area
With clock gating	1.18E-03	7.64E-04	2.60E-04	2.20E-03	1305	141618.257
Without clock gating	1.26E-03	2.38E-03	9.36E-04	4.57E-03	1532	147618.96

References

1. A. Puşcaşu, C. B. Ciobanu and O. Buiu, "Systolic Array Matrix Multiplication Accelerator," 2024 International Semiconductor Conference (CAS), Sinaia, Romania, 2024, pp. 207-210, doi: 10.1109/CAS62834.2024.10736842.