# BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI



# **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.

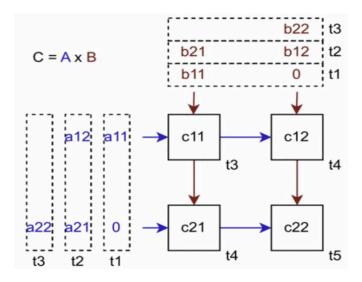
## Architectural & Functional Details – Systolic Array Multiplier

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<sub>11</sub> and b<sub>11</sub>, computes a<sub>11</sub>·b<sub>11</sub>, stores in c<sub>11</sub>
- Other PEs: Idle

## **Clock Cycle 2:**

- PE(1,1): Receives a<sub>12</sub> and b<sub>21</sub>, computes a<sub>12</sub>·b<sub>21</sub>, adds to c<sub>11</sub>
- PE(1,2): Receives a<sub>11</sub> and b<sub>12</sub>, computes a<sub>11</sub>·b<sub>12</sub>, stores in c<sub>12</sub>
- PE(2,1): Receives a<sub>21</sub> and b<sub>11</sub>, computes a<sub>21</sub>·b<sub>11</sub>, stores in c<sub>21</sub>
- 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<sub>12</sub> and b<sub>22</sub>, computes a<sub>12</sub>·b<sub>22</sub>, adds to c<sub>12</sub>
- PE(2,1): Receives a<sub>22</sub> and b<sub>21</sub>, computes a<sub>22</sub>·b<sub>21</sub>, adds to c<sub>21</sub>
- PE(2,2): Receives a<sub>21</sub> and b<sub>12</sub>, computes a<sub>21</sub>·b<sub>12</sub>, stores in c<sub>22</sub>

#### **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<sub>22</sub> and b<sub>22</sub>, computes a<sub>22</sub>·b<sub>22</sub>, adds to c<sub>22</sub>

#### **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.

The fund	ctional	simulation timing diagrams for an example o	of 8x8 matrix multiplication is below.
			255 000 ng
Name	Value		
> <b>*</b> C00[15:0]	204	0.000 ns	0.000.ns.  300.000.ns.  350.000.ns.  400.000.ns.  450.000.ns
> <b>U</b> C01[15:0]	233	x 0 (2 8 20 40 70 112 161) 23	3
> ♥ C02[15:0]	276	x 0 (3 (11 )26 (50 )85 (133 )196	276
> ♥C03[15:0]	312	X 0 (4)(14)(32)(60)(100)(154)(224)	312
> ♥ C04[15:0]	348	X 0 (5)(17)(38)(70)(115)(175)(252)	348
> <b>V</b> C05[15:0]	384	0 6 20 44 80 130 196 280	384 0
> ♥ C06[15:0]	420	x 0 7 23 50 90 145 217 308	420 0
> • C07[15:0]	456	X 0 (8 (26 (56 )100 )160 )238 )336 )	456 0
> ♥C10[15:0] > ♥C11[15:0]	240 276	X 0 2 8 20 40 70 112 168 24 X 0 4 13 29 54 90 139 199	0, , , , , , , , , , , , , , , , , , ,
> <b>U</b> C12[15:0]	328	x 0 4 13 29 54 90 139 195 x 0 6 18 38 68 110 166 238	276 0 328 0
> <b>U</b> C13[15:0]	372	x 0 8 23 47 82 130 193 273	372 0
> ♥C14[15:0]	416	X 0 (10 (28 )56 (96 )150 (220 )308	416 0
> ♥C15[15:0]	460	x 0 (12 (33 (65 (110 (170 (247 (343 )	460
> <b>♥</b> C16[15:0]	504	0 (14) 38 (74) 124 (190) 274 (378)	504
> ♥C17[15:0]	548	X 0 (16 43 (83 )138 210 301 413 (	548 0
> ♥ C20[15:0] > ♥ C21[15:0]	269 312	X 0 (3 (11 (26 (50 (85 (133 (189 (	269 0
> <b>U</b> C22[15:0]	371	x 0 6 18 38 68 110 166 222 x 0 9 22 50 88 133 199 273	312 0
> <b>U</b> C23[15:0]	422	x 0 (9 (25 (30) (86 (33) (199) 2/1) x 0 (12 (32 (62 (104) (160) (232) 312)	422
> ₩C24[15:0]	473	X 0 (15 (39 (74 )122 )185 (265 )353 )	473 0
> ♥C25[15:0]	524	x 0 (18 (46 (86 )140 210 298 394 (	524 0
> <b>V</b> C26[15:0]	575	X 0 (21 (53 ) 98 158 235 331 435	575
> ♥C27[15:0]	626	X 0 (24 (60 )110)176/260/364/476)	626
> W C30[15:0]	312	Z 0 4 14 32 60 100 154 224	312
> ♥ C31[15:0] > ♥ C32[15:0]	362 432	X 0 (8 23 47 82 130 193 263 X X 0 (12 32 62 104 156 232 322 X	362 0 0 0 432
> <b>*</b> C32[15:0] > <b>*</b> C33[15:0]	492	x 0 (12 / 32 / 62 / 104 / 160 / 232 / 322 / 2 x 0 (16 / 41 / 77 / 126 / 190 / 271 / 371 /	432 0
> <b>U</b> C34[15:0]	552	X 0 20 50 92 148 220 310 420	552 0
> <b>♥</b> C35[15:0]	612	24 (59 107 170 250 349 469	612 0
> <b>♥</b> C36[15:0]	672	X 0 28 68 122 192 280 388 518	672
> ♥C37[15:0]	732	x ( )32 (77 )137)214 (310 )427 )567 (	732
> <b>W</b> C40[15:0]	348	0 (5 (17 )38 (70 )115 (175/252)	348
> <b>W</b> C41[15:0]	405	I 0 10 28 56 96 150 220 297	405
> <b>U</b> C42[15:0]	484	0 (15) 39 (74) 122 (185) 265 (364)	484 0
> <b>W</b> C43[15:0]	552 620	20 50 92 148 220 310 420	552 0
> W C44[15:0] > W C45[15:0]	688	x 0 (25) (61) (110) (174) (255) (355) (476) x 0 (30) (72) (128) (200) (290) (400) (532)	620 0 638 0
> <b>V</b> C46[15:0]	756	T 0 35 (22 220 )250 (430 )352	756 0
> ₩C47[15:0]	824	T 0 (40 94 164 252 360 490 644)	824 0
> ♥C50[15:0]	384	z 0 (6 (20 (44 (80 )130 196 280 (	384 0
> <b>U</b> C51[15:0]	448	0 (12 33 65 110 170 247 331	448
> ♥ C52[15:0]	536	18 46 86 140 210 298 406	536 0
> <b>©</b> C53[15:0] > <b>©</b> C54[15:0]	612 688	0 (24 59 107 170 250 349 469)	612 0
> <b>W</b> C55[15:0]	764	x 0 (30 / 72 / 128 / 200 / 290 / 400 / 532 / x 0 (36 (85 / 149 / 230 / 330 / 451 / 595 )	688 0 764 0
> <b>U</b> C56[15:0]	840	x 0 (38 (39 )(49 )(230 )(330 )(33 )(39 )(39 )	840 0
> ♥ C57[15:0]	916	z 0 (48 l11 l91 290 410 553 721)	916 0
> <b>V</b> C60[15:0]	418	0 (7)(21)(48)(88)(143)(215)(306)	418 0
> <b>V</b> C61[15:0]	488	0 (14 35 71 121 187 271 362	488 0
> ♥ C62[15:0]	584	21 49 94 154 231 327 444	584 0
> <b>©</b> C63[15:0] > <b>©</b> C64[15:0]	667 750	28 63 117 187 275 383 513	667 0
> <b>W</b> C65[15:0]	833	x 0 (35 (77 )440 (220 )319 (439 582 ) x 0 (42 )91 (163 (253 )363 (495 (651 )	750 0 833 0
> <b>U</b> C66[15:0]	916	T 0 (49 105 186 286 407 551 720)	916 0
> ♥ C67[15:0]	999	T 0 (56 )119 209 319 451 607 789	999 0
> ₩C70[15:0]	456	Z 0 (8 (26 (56 )100 (160 (238 )336 )	456 0
> <b>V</b> C71[15:0]	534	0 (16 43 83 138 210 301 399	534 0
> W C72[15:0]	640	0 (24 60 110 176 260 364 490	640 0
> WC73[15:0] > WC74[15:0]	732 824	x 0 (32 (77 )337 214 310 427 567 ) x 0 (40 (94 )164 252 360 (490 )644 )	732 0
> <b>U</b> C74[15:0] > <b>U</b> C75[15:0]	916	x 0 (40 (94 )164 252 360 490 644 x 0 (48 )111 191 290 410 553 721	916 0
> <b>U</b> C76[15:0]	1008	T 0 (48 )111 (191 (290 (410 )555 )721 (191 (191 )555 )721 (191 )555 (191 )55	1008 0
> <b>V</b> C77[15:0]	1100	Z 0 64 145 245 366 510 679 875	1100 0
₩m_clk	1		
⊌rst	0		
¹å clk	1		
> # count[4:0]	23	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	23 24 25 26 27 0
> <b>*</b> A0[7:0]	0	0 12345678	0
> MA1[7:0]	0	0 2 3 4 5 6 7 8 9	
> <b>W</b> A2[7:0] > <b>W</b> A3[7:0]	0	2 0 3 4 5 6 7 8 10 11 2 0 4 5 6 7 8 9 10 11 0	0
> <b>%</b> A4[7:0]	0	Z 0 (4 (5 (6 (7 ) 8 ) 9 (10 (11 ) Z 0 (5 (6 (7 ) 8 (9 (10 (11 ) 12 )	
> MA5[7:0]	0	2 0 (6 7 (8 9 (10 (11 (12 (13 (	0
> <b>W</b> A6[7:0]	0	x 0 7 9 10 11 12 13 14	0
> • A7[7:0]	0	X 0 (8 (9 (10 (11 (12 (13 (14 (15 )	0
> <b>*</b> B0[7:0]	0	0 (1)(2)(3)(4)(5)(6)(7)(8)	0
> <b>W</b> B1[7:0]	0	0 2 3 4 5 6 7 9	0
> <b>W</b> B2[7:0]	0	0 34567789910	
> <b>W</b> B3[7:0] > <b>W</b> B4[7:0]	0	0 (4 5 6 7 8 9 10 11 ) 2 0 (5 6 7 8 9 10 11 12 )	0
> <b>8</b> B5[7:0]	0	2 0 \( \lambda 5 \lambda 6 \lambda 7 \lambda 8 \lambda 9 \lambda 10 \lambda 11 \lambda 12 \\ 2 \\ 0 \\ \lambda 6 \lambda 7 \lambda 8 \lambda 9 \lambda 10 \lambda 11 \lambda 12 \lambda 13 \\ \lambda 1 \lambd	0
> • B6[7:0]	0	T 0 7 8 9 10 11 12 13 14 1	0
> <b>*</b> B7[7:0]	0	0 (8 (9 (10 (11 (12 (13 (14 (15 )	0

# 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.