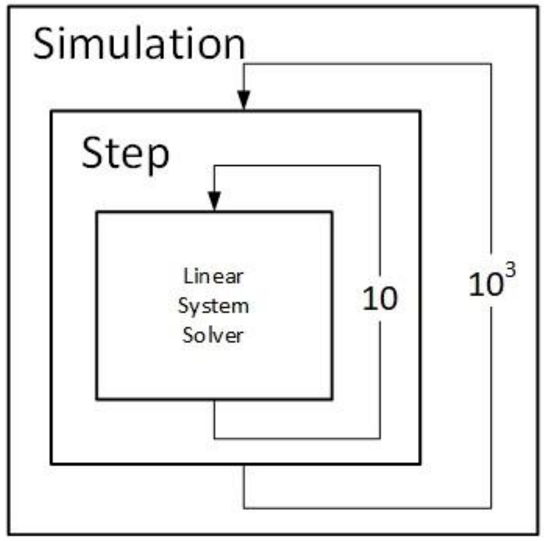
Parallel Sparse Matrix Solver for Circuit Simulations using FPGA

Yogesh Mahajan
Supervisor: Prof. Sachin Patkar

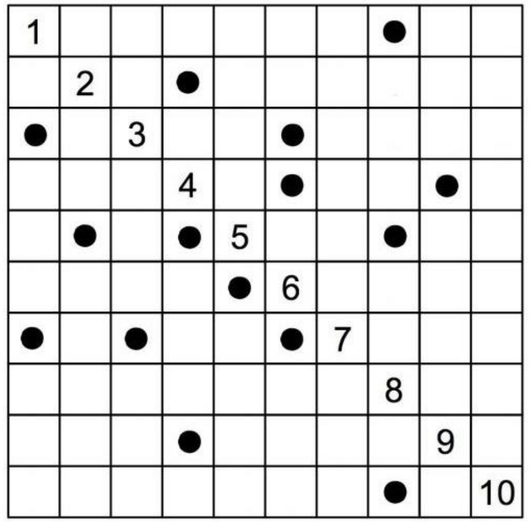
Electrical Engineering, IIT Bombay
October 2018

Motivation

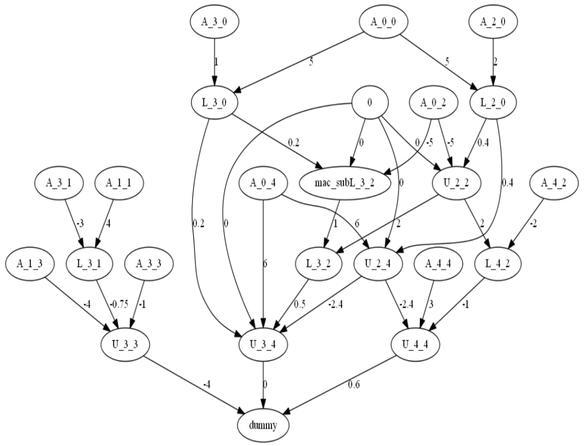
Why and How to Accelerate Circuit Simulations?



Simulation requires several thousand repeated solution of the network matrix



Structure of matrix remains the same in each iteration



Data manipulation can be modelled using a task graph to extract parallelism

Parallelizing the Sparse LU Decomposition

Pre-processing 1

- Convert matrix to sparse storage format
- Pre-order the matrix for fill in reduction
- Generate verification data

Symbolic Analysis 2

- Find the non-zero location in factors
- Generate the computation flow graph
- Assign priority to each operation

Scheduling 3

- Generate memory map
- Generate priority list based schedule for the computation flow graph

Numeric Computation 4

- Copy matrix data to FPGA
- Execute using the predefined schedule
- Write results back to main memory

Sparse Matrix Storage Formats

$$\begin{bmatrix} 5 & 0 & -5 & 0 & 6 \\ 0 & 4 & 0 & -4 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 1 & -3 & 0 & -1 & 0 \\ 0 & 0 & -2 & 0 & 3 \end{bmatrix}$$

Sparse Matrix

$$\begin{bmatrix} 5 & -5 & 6 \\ 4 & -4 & 0 \\ 2 & 0 & 0 \\ 1 & -3 & -1 \\ -2 & 3 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & -1 \\ 0 & -1 & -1 \\ 0 & 1 & 3 \\ 2 & 4 & -1 \end{bmatrix}$$

ELLPACK

Values	5	-5	6	4	-4	2	-	-3	-1	-2	3
Column Indices	0	2	4	1	3	0	0	1	3	2	4
Row Indices	0	0	0	1	1	2	3	3	3	4	4

Triplet Format

Values	5	2	1	4	-3	-5	-2	-4	-1	6	3
Row Indices	0	2	3	1	3	0	4	1	3	0	4
Column Pointers	0			3		5		7		9	

Compressed Column Format

Values	5	-5	6	4	-4	2	-	-3	-1	-2	3
Column Indices	0	2	4	1	3	0	0	1	3	2	4
Row Pointers	0			3		5		6		9	

Compressed Row Format

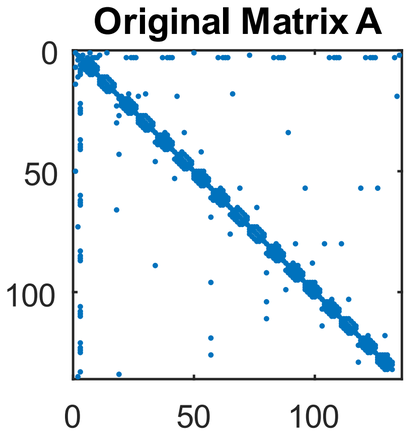
Approximate Minimum Degree Ordering

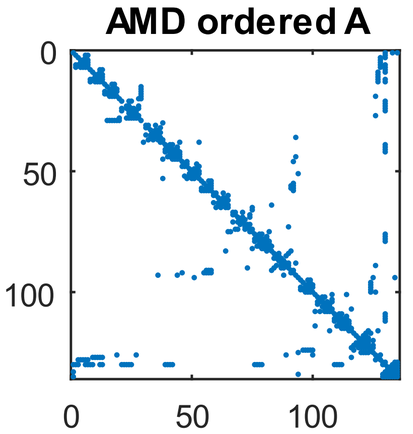
Find a permutation matrix $P = amd(A)$ such that the *Cholesky factor* of $P^T AP$ has less number of fill-ins than the factors of A, where A is a $n \times n$ symmetric matrix

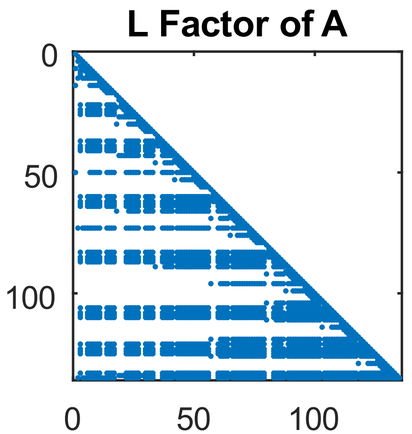
For asymmetric matrix $P = amd(A + A^T)$

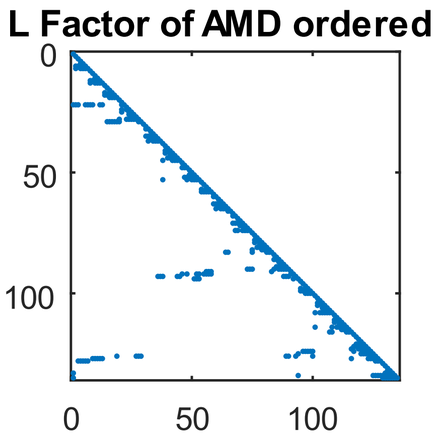
The linear problem $Ax = b$ can be solved by solving the reordered system

$$(P^T AP)(P^T x) = P^T b$$







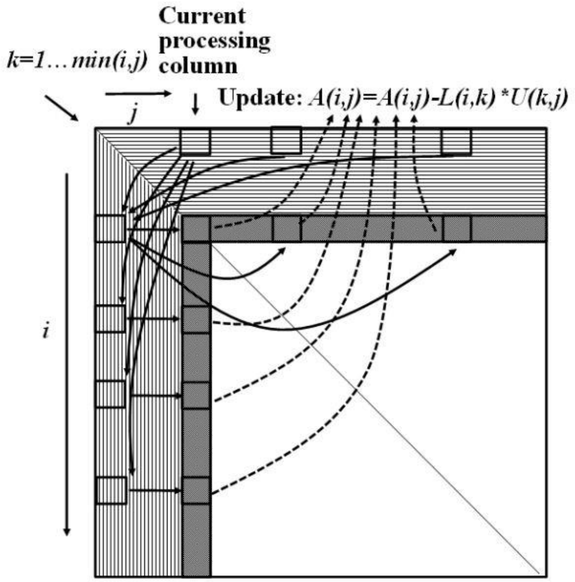


Symbolic Analysis

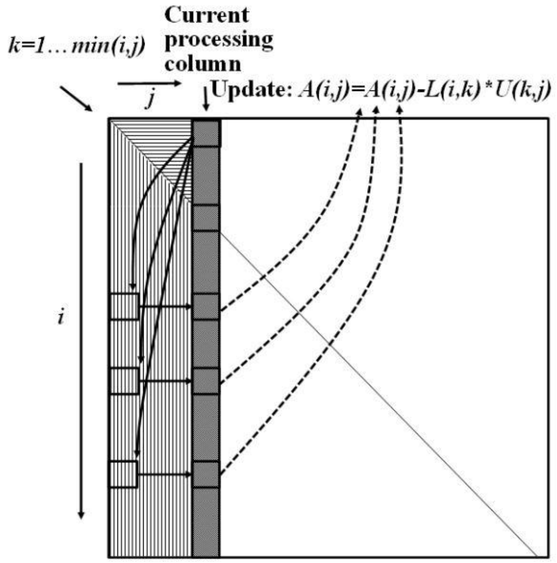
$$U_{(i,j)} = A_{(i,j)} - \sum_{k=1}^{i-1} L_{(i,k)} U_{(k,j)}$$

$$L_{(i,j)} = \frac{A_{(i,j)} - \sum_{k=1}^{j-1} L_{(i,k)} U_{(k,j)}}{U_{(j,j)}}$$

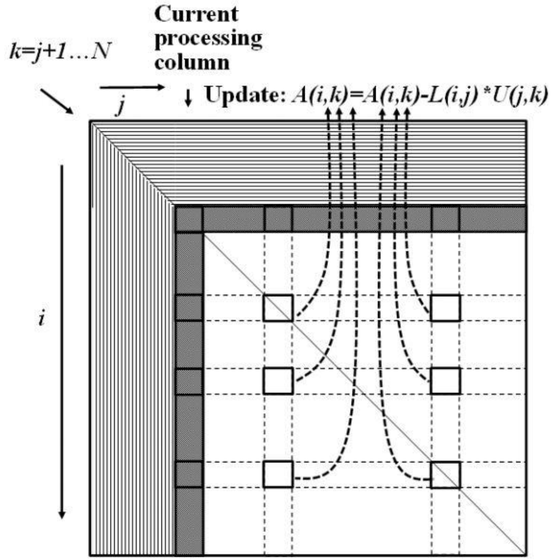
Crout's



Left-Looking



Right-Looking

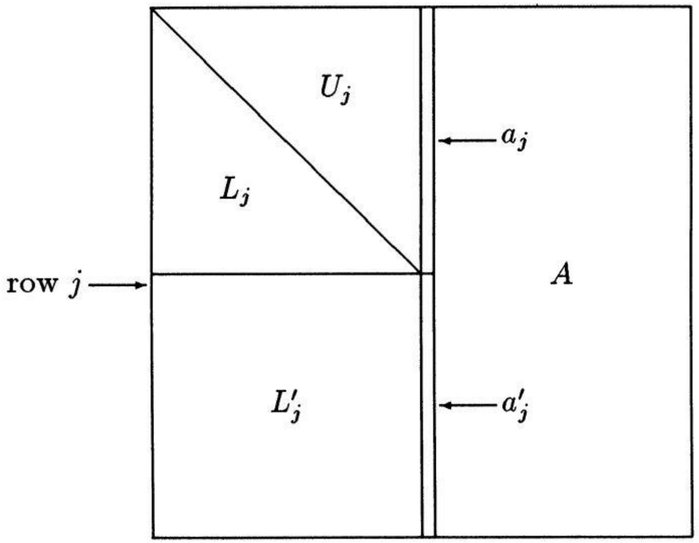


Gilbert-Peierls' Algorithm

Algorithm 1 Gilbert-Peierls Algorithm: A Column-Oriented LU Factorization

Precondition: A, a $n \times n$ asymmetric matrix

1 $L := I$
2 **for** $j := 1$ to n **do**
3 Solve $L_j u_j = a_j$ for u_j
4 $b'_j := a'_j - L'j u_j$
5 Do Partial Pivoting on $b'j$
6 $u_{jj} := b_{jj}$
7 $l'_j := b'_j / u_{jj}$



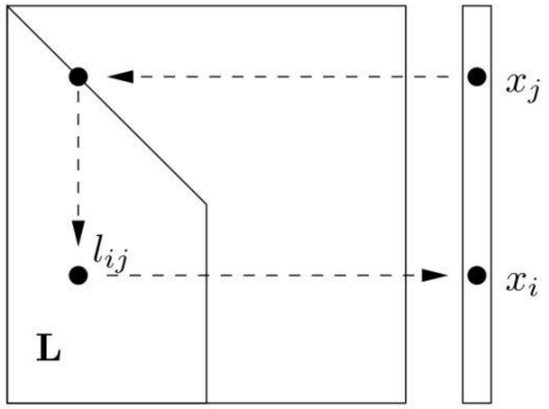
J. Gilbert and T. Peierls, "Sparse partial pivoting in time proportional to arithmetic operations," SIAM Journal on Scientific and Statistical Computing, vol. 9, no. 5, pp. 862–874, 1988.

Solving the Lower Triangular System

Algorithm 2 Gilbert-Peierls Algorithm: Solving Triangular System $L_j x = b$

Precondition: L_j is a lower triangular matrix, x, b are sparse column vectors

```
1 x := b
2 for each j ∈ χ do
3     for each i > j for which l_ij ≠ 0 do
4         x_i := x_i − l_ij x_j
```



$$(b_i \neq 0) \implies (x_i \neq 0)$$

$$(x_j \neq 0) \,||\, \wedge^{i} (l_{ij} \neq 0) \implies (x_i \neq 0)$$

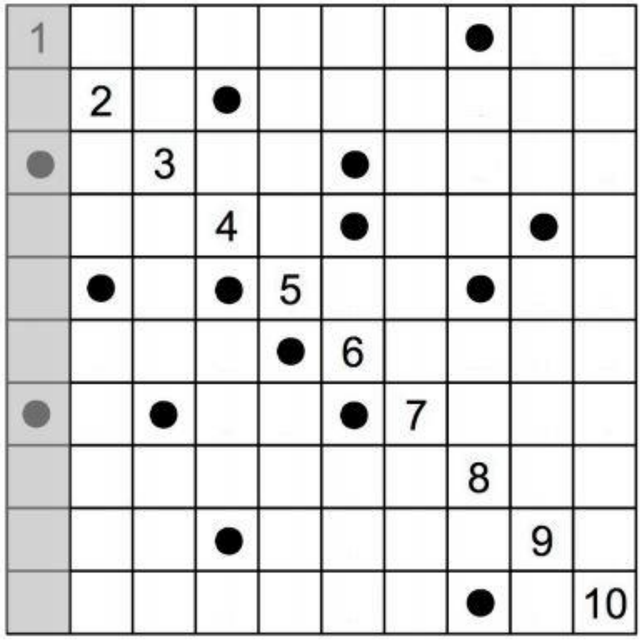
T. Nechma and M. Zwolinski, "Parallel sparse matrix solution for circuit simulation on fpgas," IEEE Transactions on Computers, vol. 64, pp. 1090–1103, April 2015.

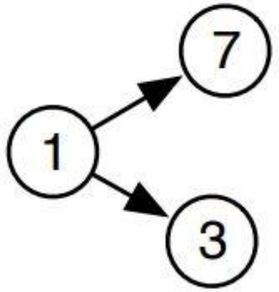
Finding Non-Zero Locations

Column dependence in the factorization process can be modelled using a DGA of columns.

Computational dependence for j^{th} column is given by $Reach$ of that column in column dependence DAG:

$$Reach(j) = \bigcup_{i \in \{i | x_{ij} \neq 0\}} Reach(i)$$





Hence the fill-ins are given by:

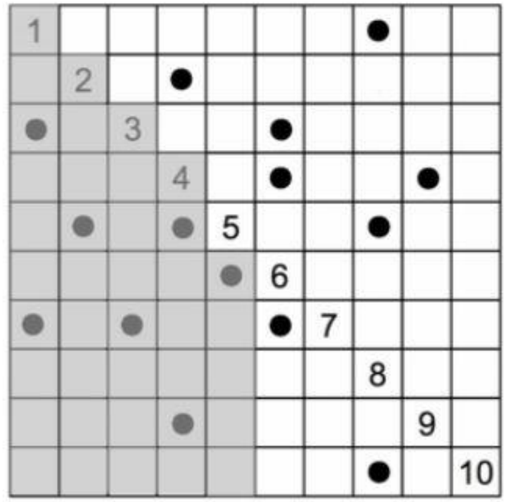
$$fillIn(j) = Reach(j) - \{i | x_{ij} \neq 0\}$$

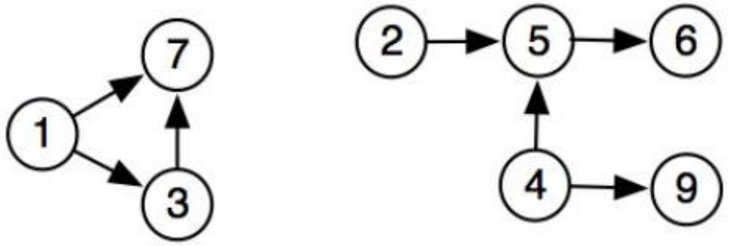
T. Nechma and M. Zwolinski, "Parallel sparse matrix solution for circuit simulation on fpgas," IEEE Transactions on Computers, vol. 64, pp. 1090–1103, April 2015.

Finding Non-Zero Locations(Example)

Step 5:
Column 6

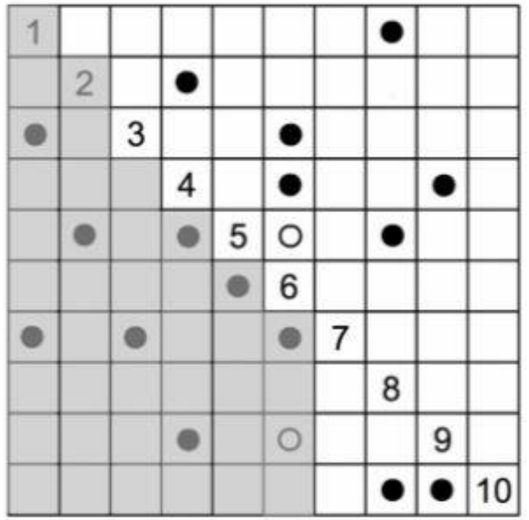
Reach(3,4,6,7) = {3,7,4,5,6,9}

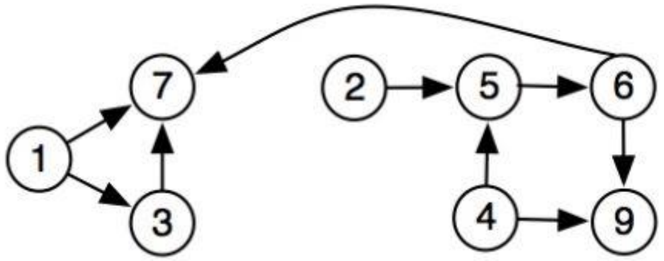




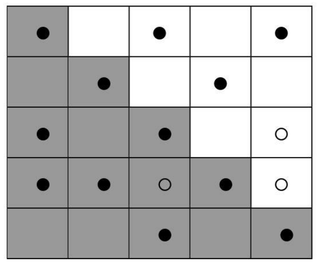
Step 6:
Column 7

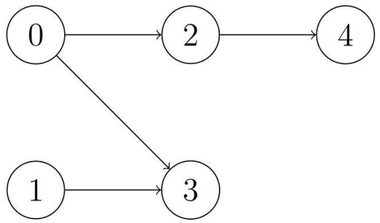
Rech(7) = {7}

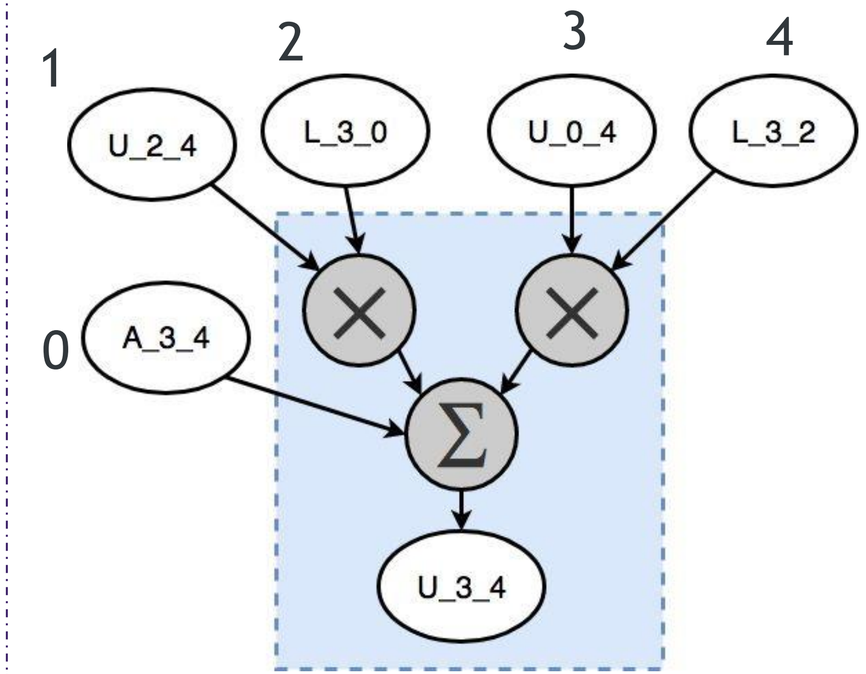


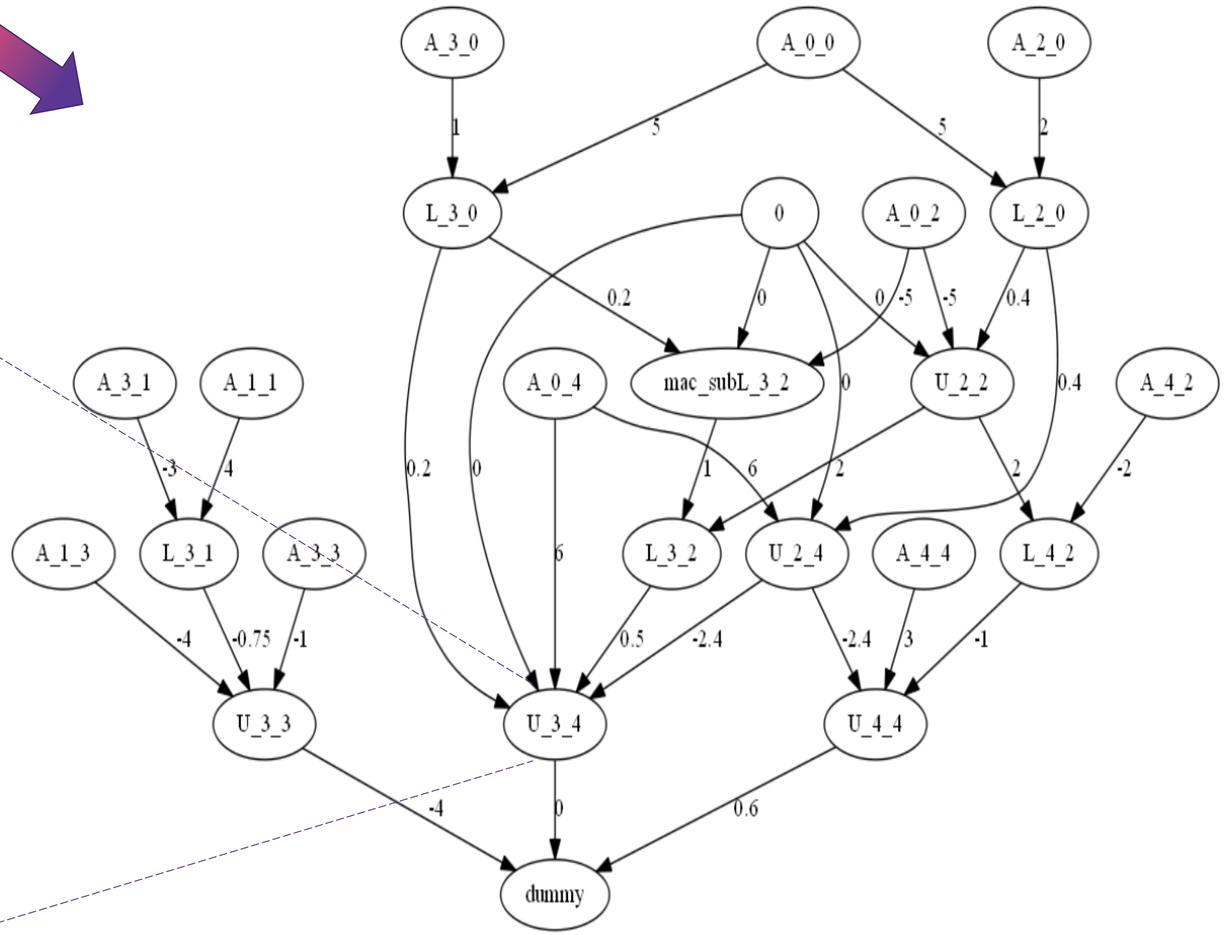


Computation Task Graph

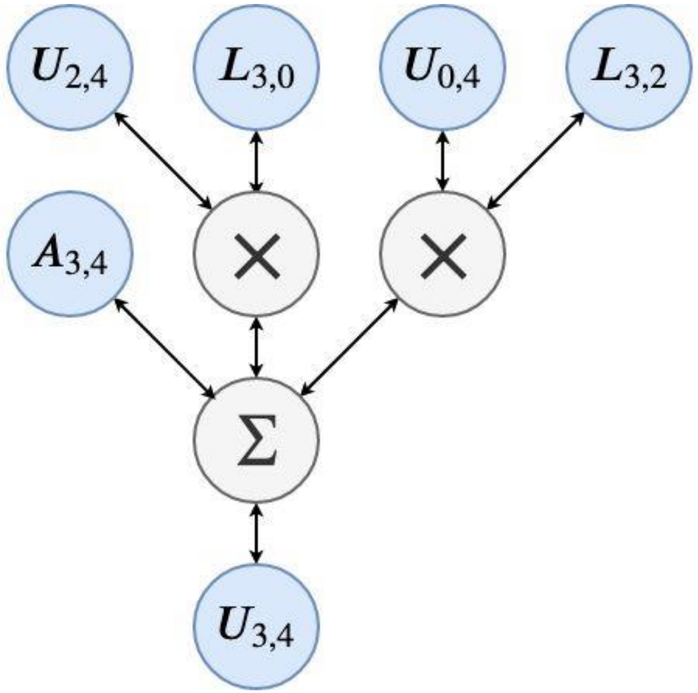






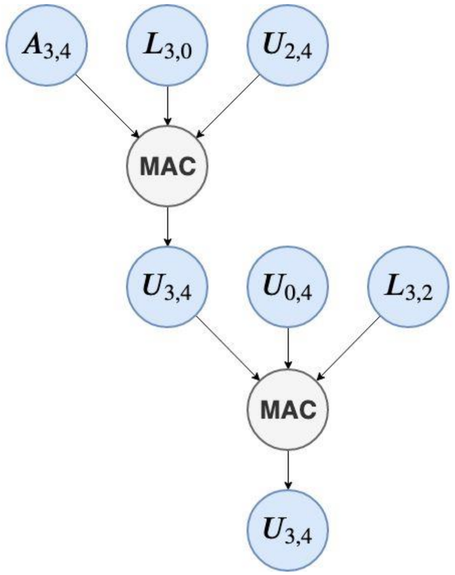


MAC Super Node

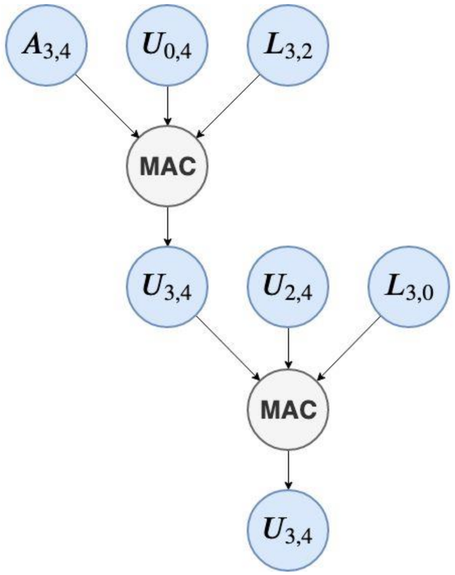


$$U_{3,4} = A_{3.4} - U_{3,4}L_{3,0} - U_{0,4}L_{3,2}$$

Total n! different execution sequences are possible for computing an element with n pairs of multiplicands



Sequence 1



Sequence 2

Scheduling

- Priority of the Operations
- Memory Allocation Policy
- Hardware Architecture
- Scheduling Algorithm

Priority of each node is computed recursively using

$$Priority(n) = \sum_{x \in parents(n)} Priority(n) + \sum_{i \in tasks(n)} Delay(i)$$

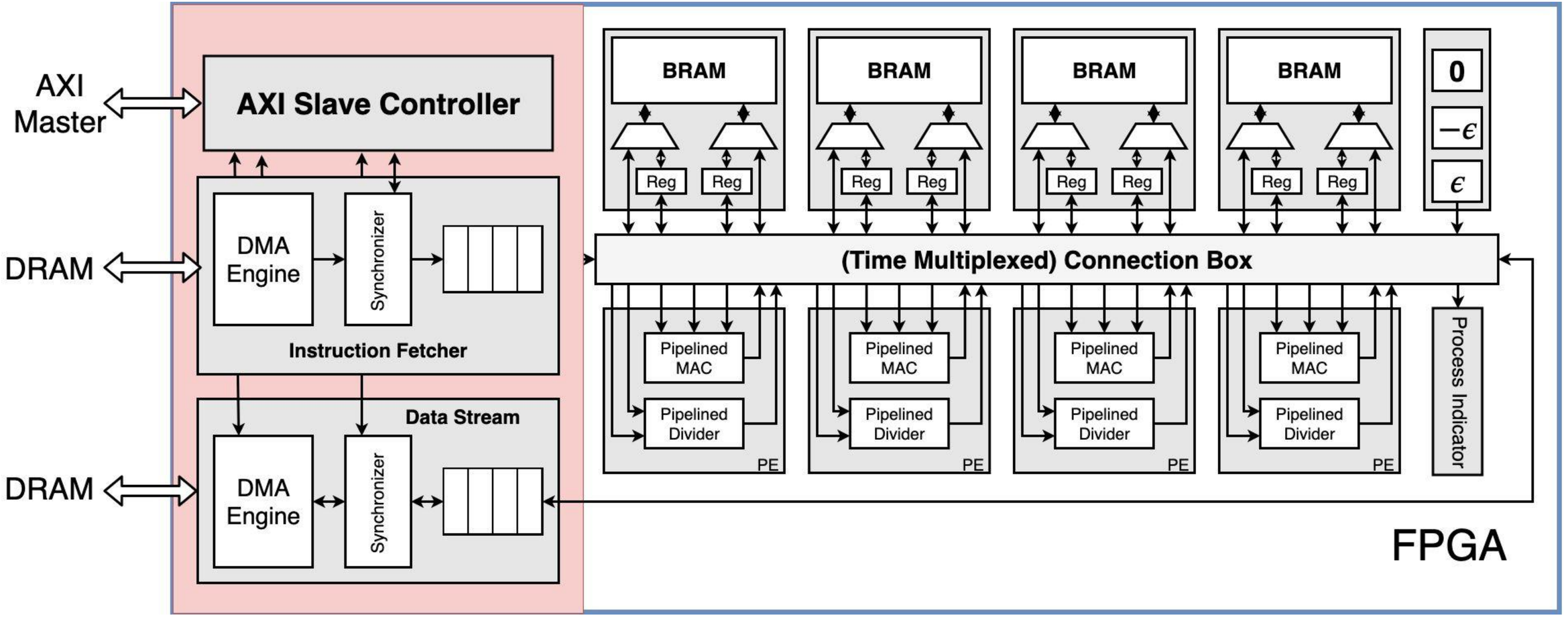
Column wise Memory Allocation

	BRAM 0	BRAM 1	BRAM 2
0	$A_{(0,0)}, U_{(0,0)}$	$A_{(1,1)}, U_{(1,1)}$	$A_{(0,2)}, U_{(0,2)}$
1	$A_{(2,0)}, L_{(2,0)}$	$A_{(3,1)}, L_{(3,1)}$	$U_{(2,2)}$
2	$A_{(3,0)}, L_{(3,0)}$		$L_{(3,2)}$
3			$A_{(4,2)}, L_{(4,2)}$
4			

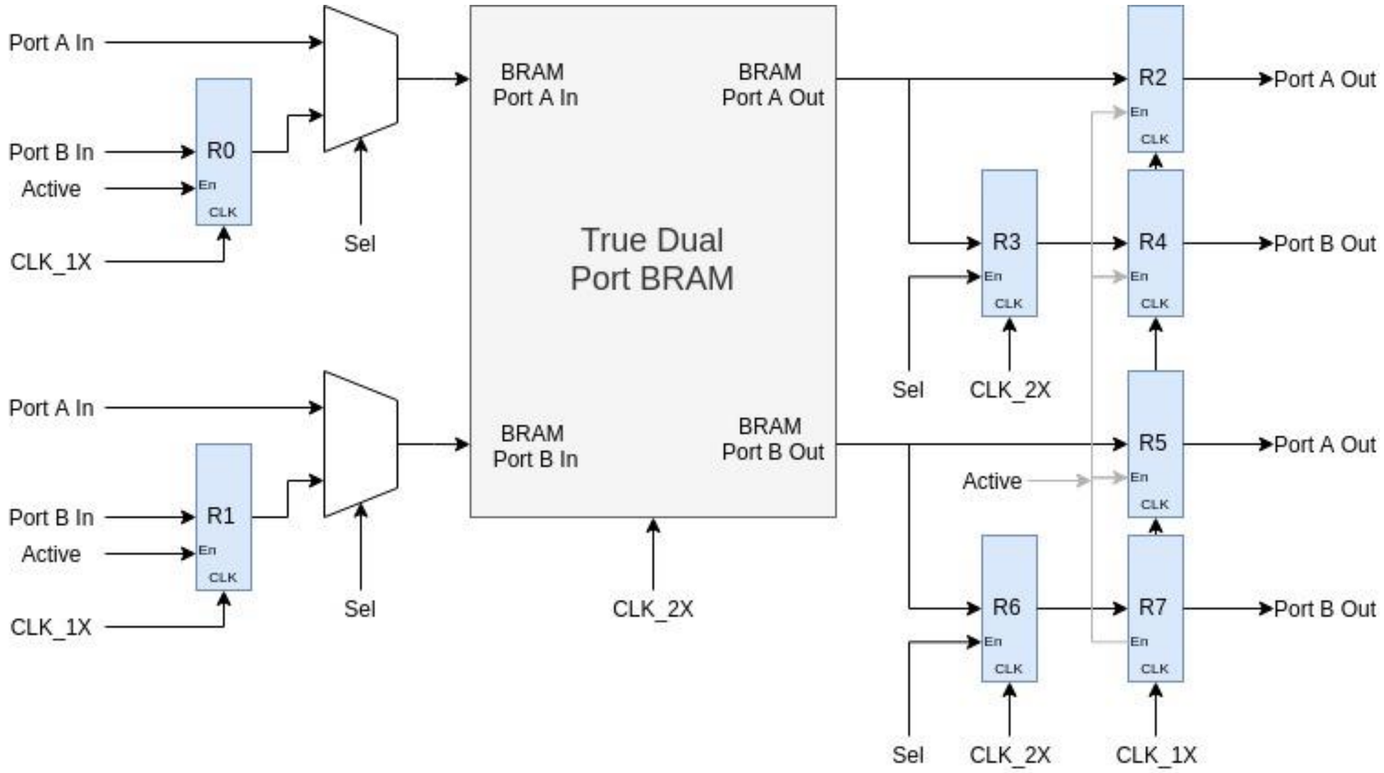
Circular Memory Allocation

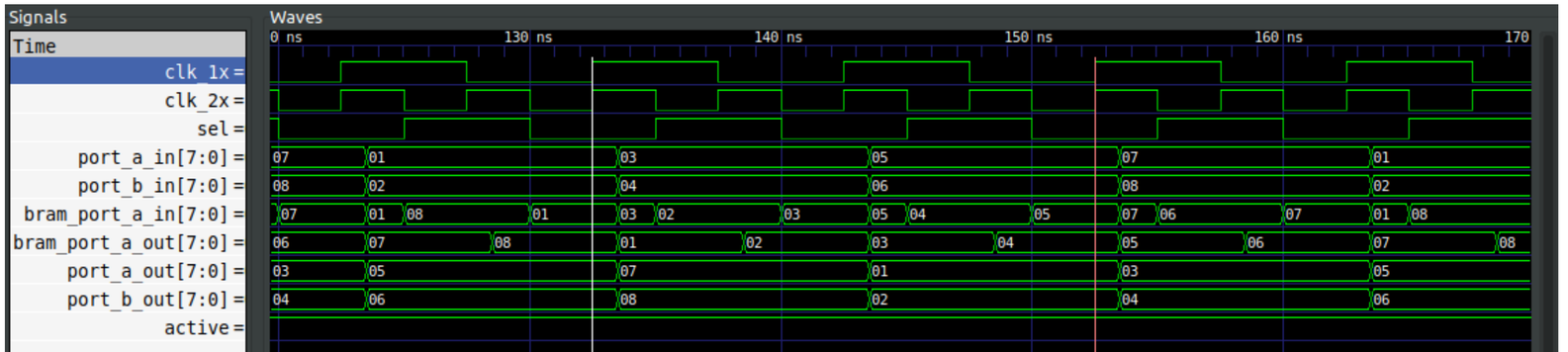
	BRAM 0	BRAM 1	BRAM 2
0	$A_{(0,0)}, U_{(0,0)}$	$A_{(2,0)}, L_{(2,0)}$	$A_{(3,0)}, L_{(3,0)}$
1	$A_{(1,1)}, U_{(1,1)}$	$A_{(3,1)}, L_{(3,1)}$	$A_{(0,2)}, U_{(0,2)}$
2	$U_{(2,2)}$	$L_{(3,2)}$	$A_{(4,2)}, L_{(4,2)}$
3			
4			

Hardware Architecture

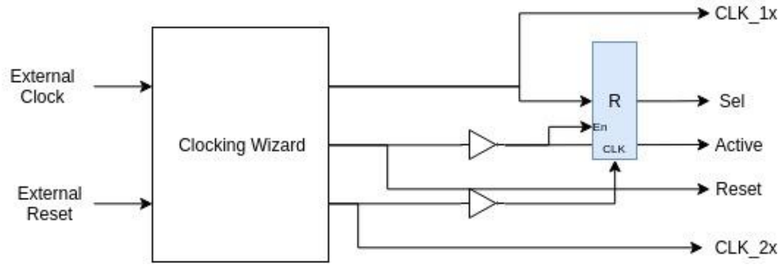


Quad Port BRAM



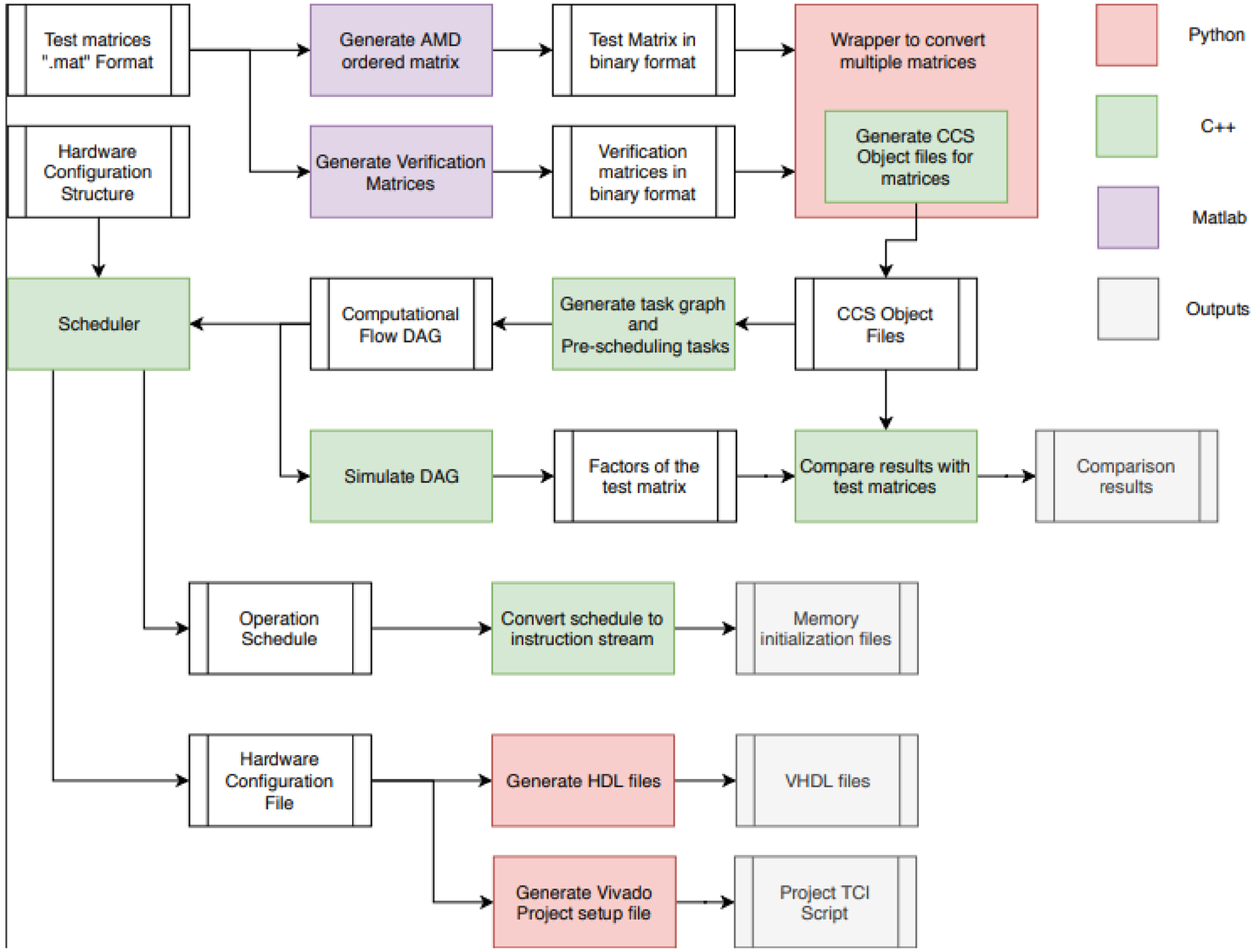


BRAMs can operate at the clock frequencies up to 400 MHz, which is at least two time faster than the processing elements

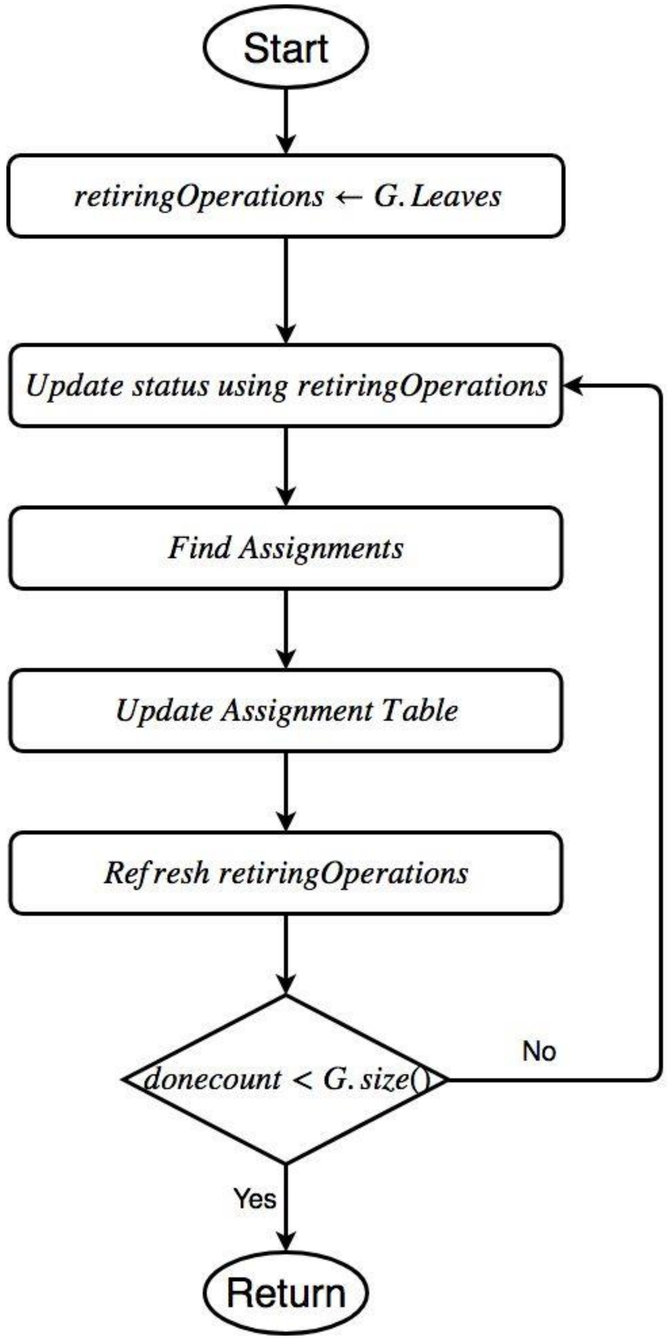


Clock and Reset Generation

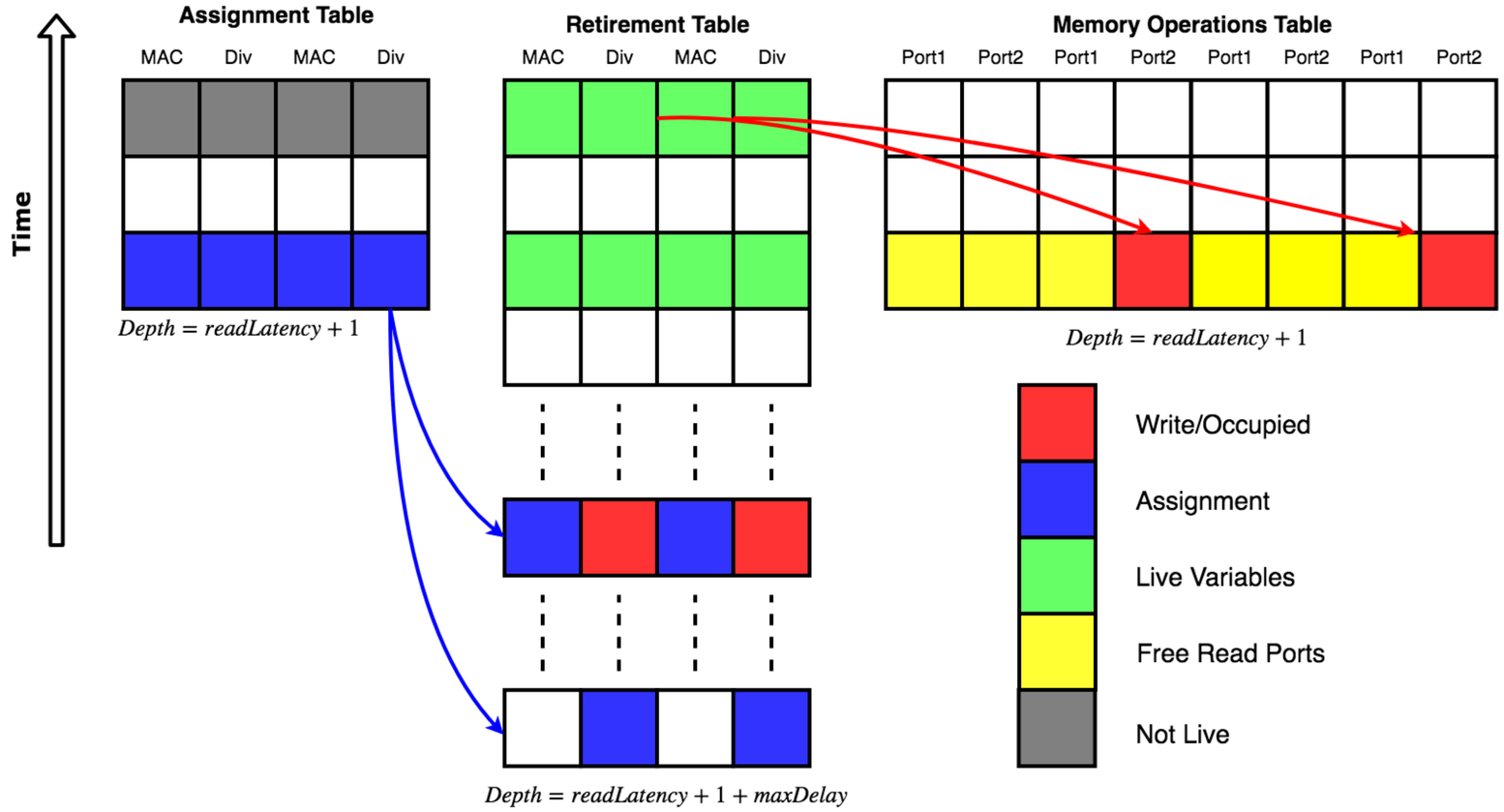
Overall Control Flow



Scheduling

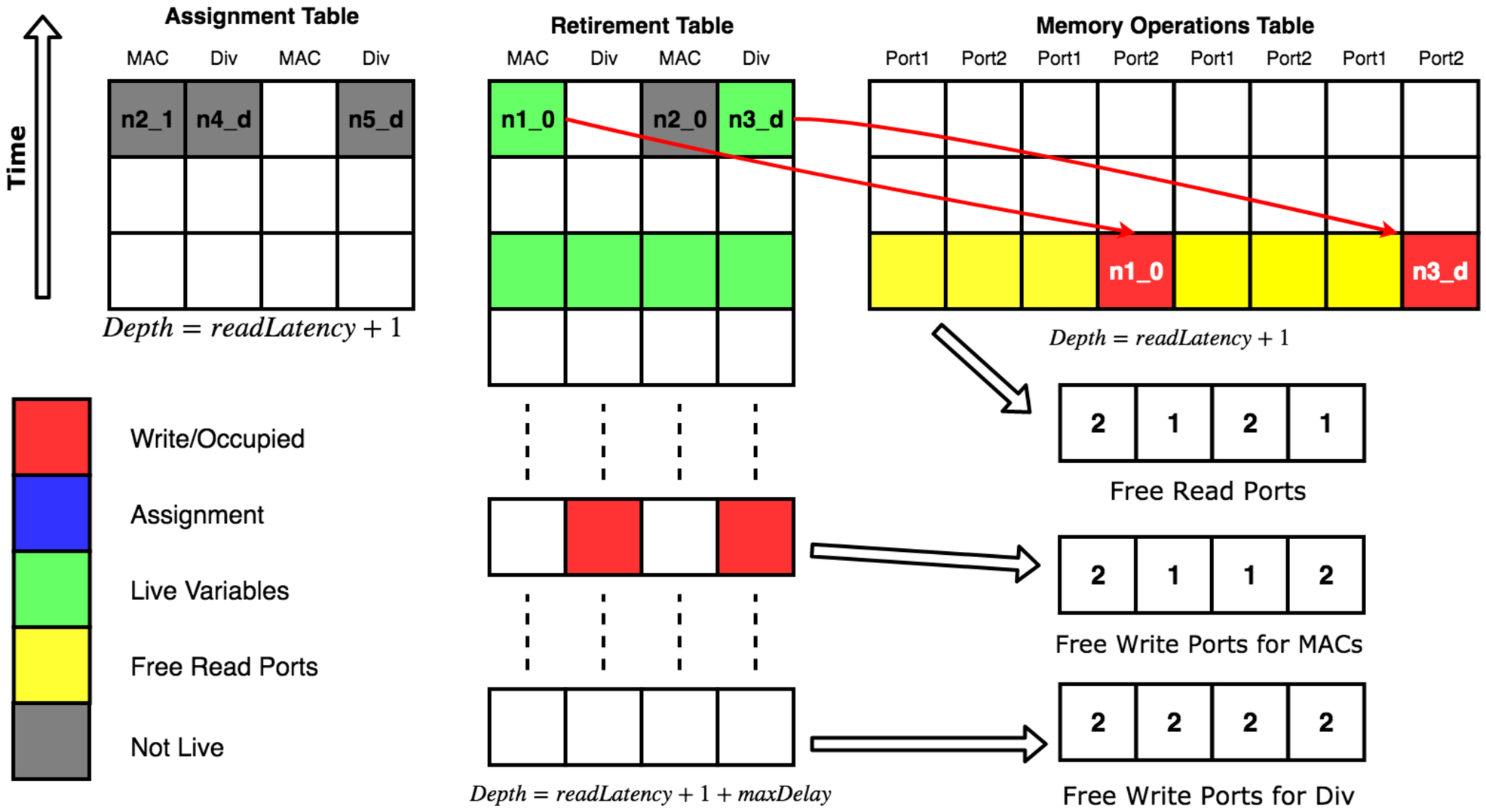


Read latency = 2



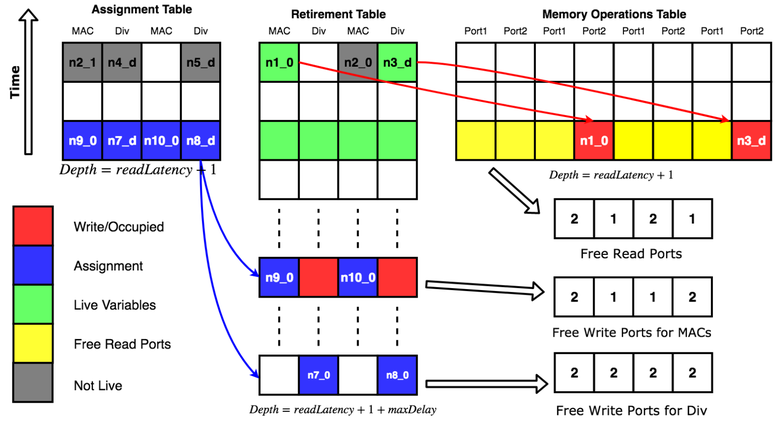
Scheduler Table

Finding Assignments



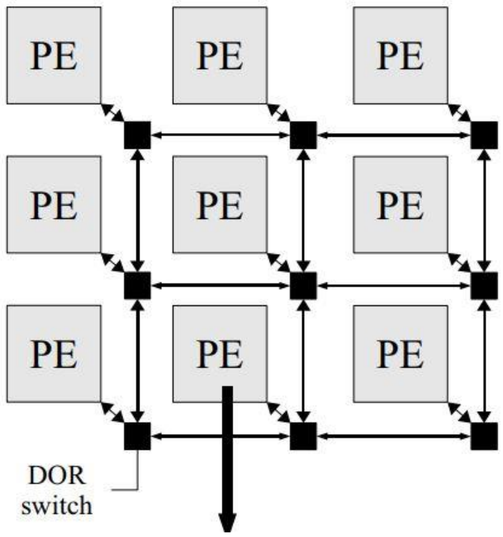
Grouping

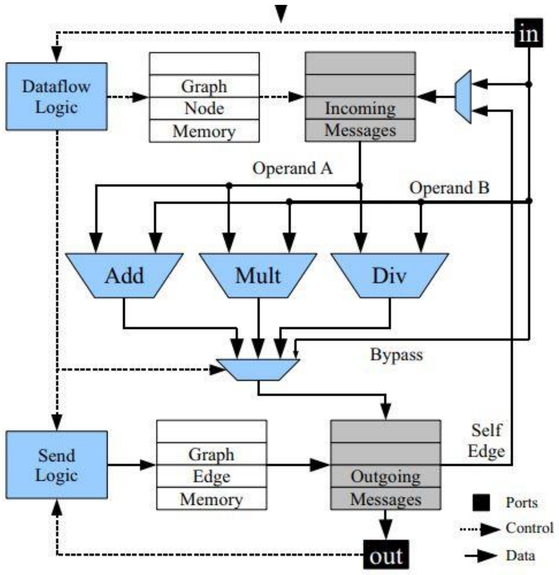
Node Ids	Available Read Ports	Required Read Ports	Available Write Ports	Required Write Ports
	2 1 2 1		2 1 1 2	
n9_0		1 1		1
n7_d		1		
n6_2		1 1 1		1
n10_0		1 1		1
n8_d		1		
	2 2 2 2			
				1
				1
				1

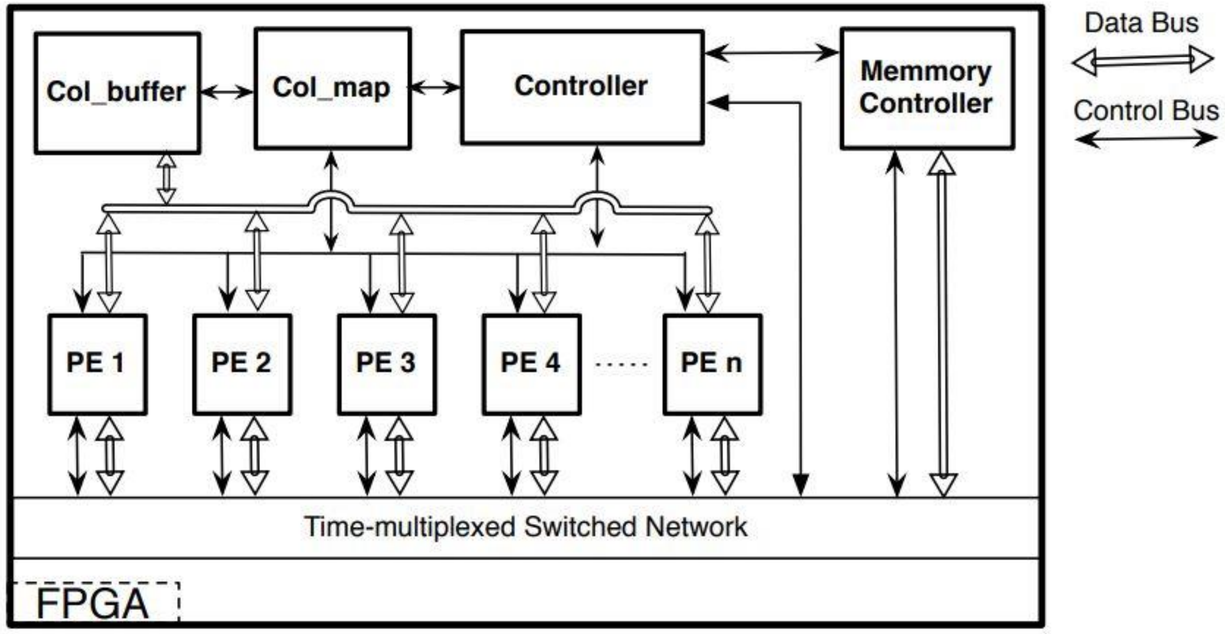


Previous Work

T. Nechma and M. Zwolinski, “Parallel sparse matrix solution for circuit simulation on fpgas,” IEEE Transactions on Computers, vol. 64, pp. 1090–1103, April 2015.

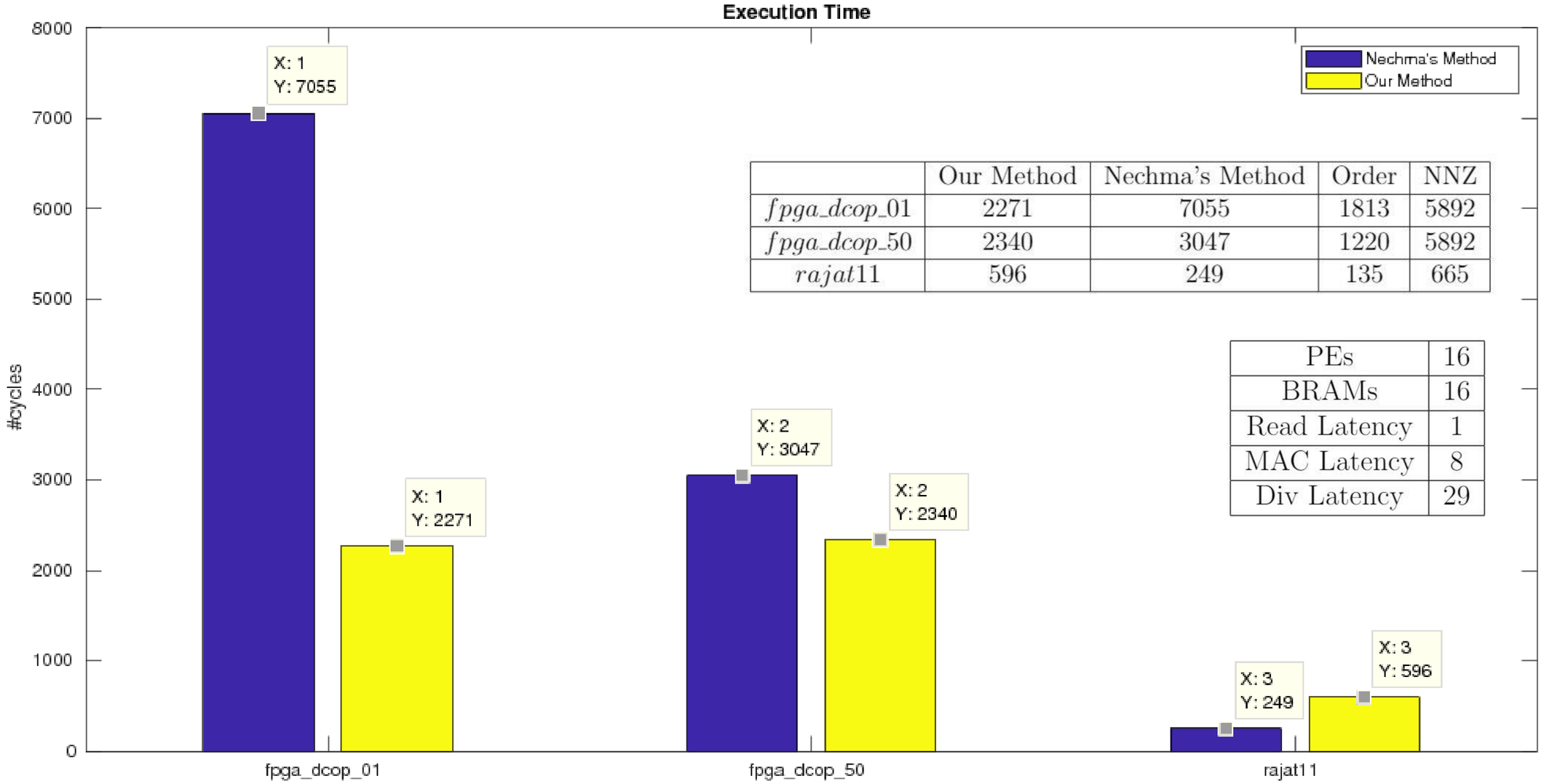






N. Kapre and A. DeHon, “Parallelizing sparse matrix solve for spice circuit simulation using fpgas,” in 2009 International Conference on Field Programmable Technology, pp. 190–198, Dec 2009.

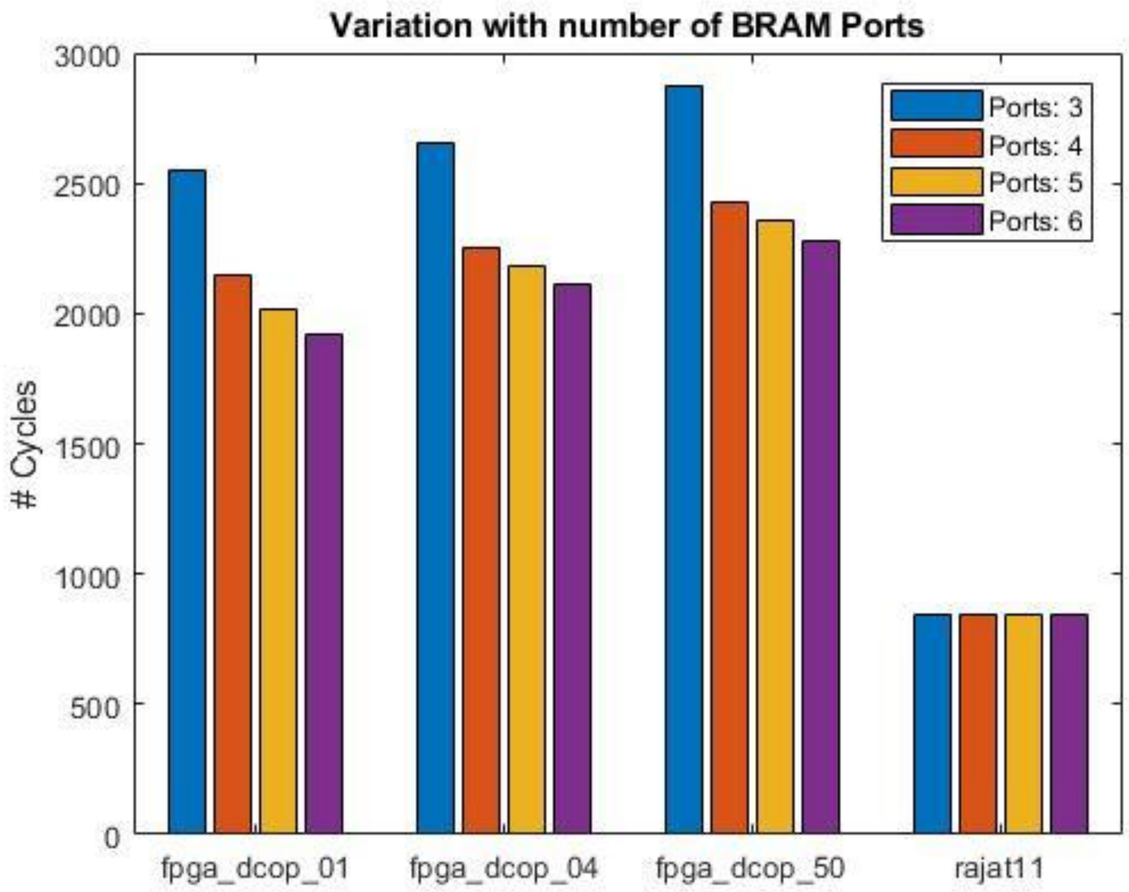
Results

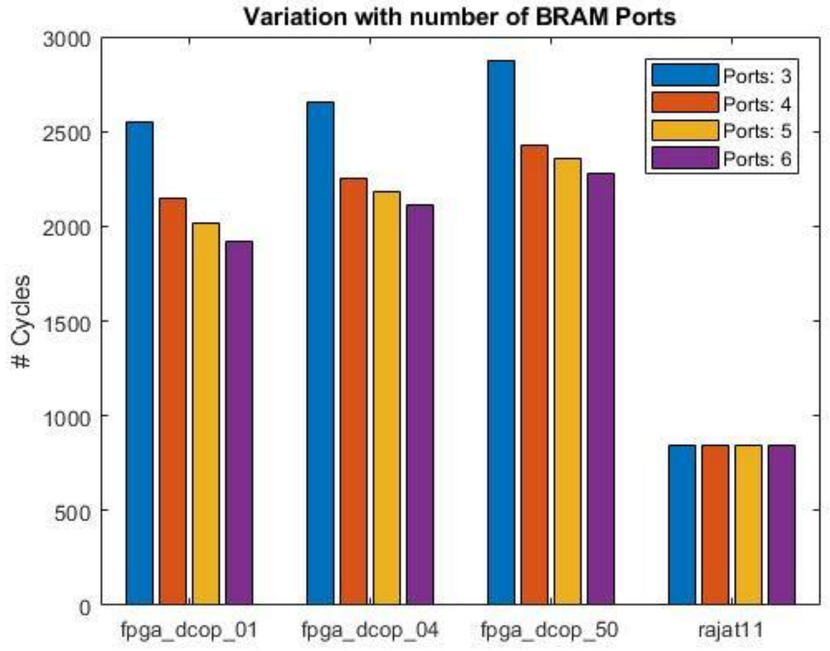


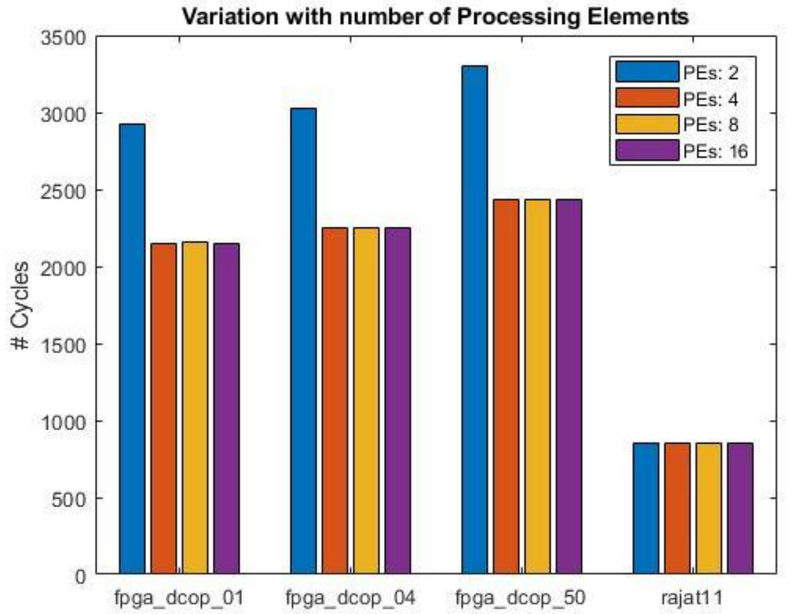
	Our Method	Nechma's Method	Order	NNZ
fpga_dcop_01	2271	7055	1813	5892
fpga_dcop_50	2340	3047	1220	5892
rajat11	596	249	135	665

PEs	16
BRAMs	16
Read Latency	1
MAC Latency	8
Div Latency	29

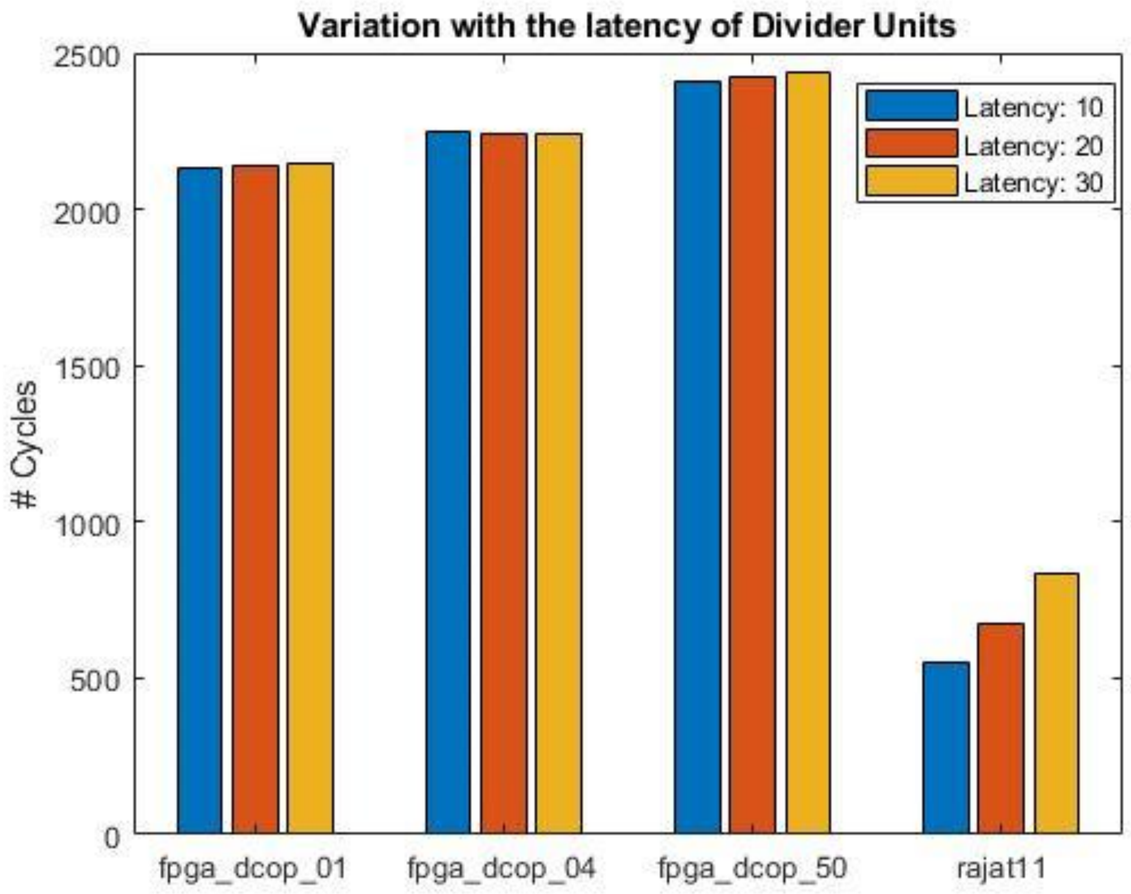
Performance Variation with the Number of Units

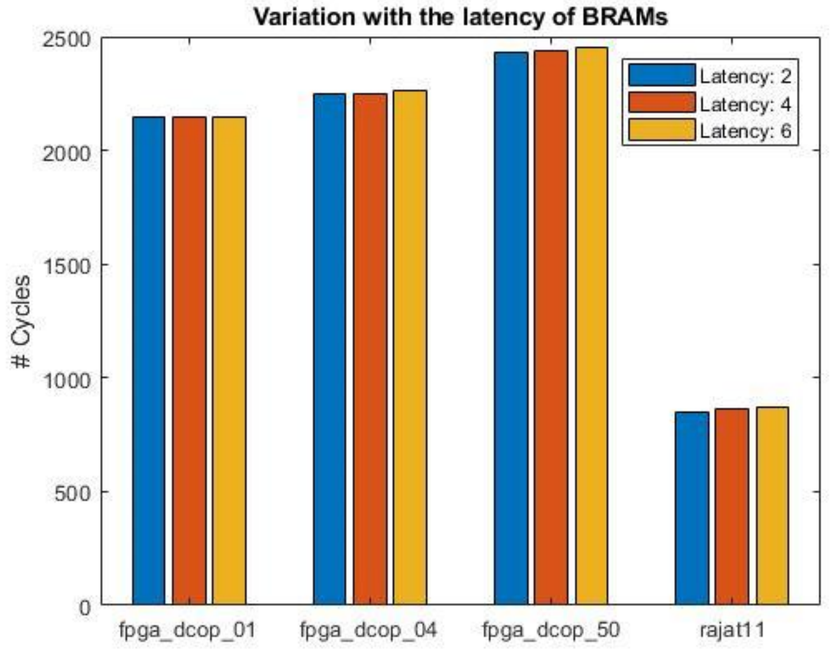


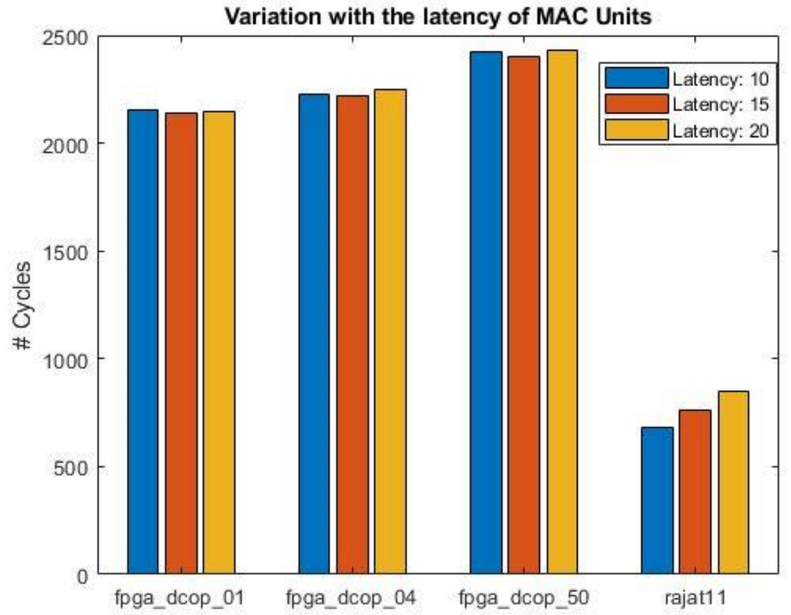




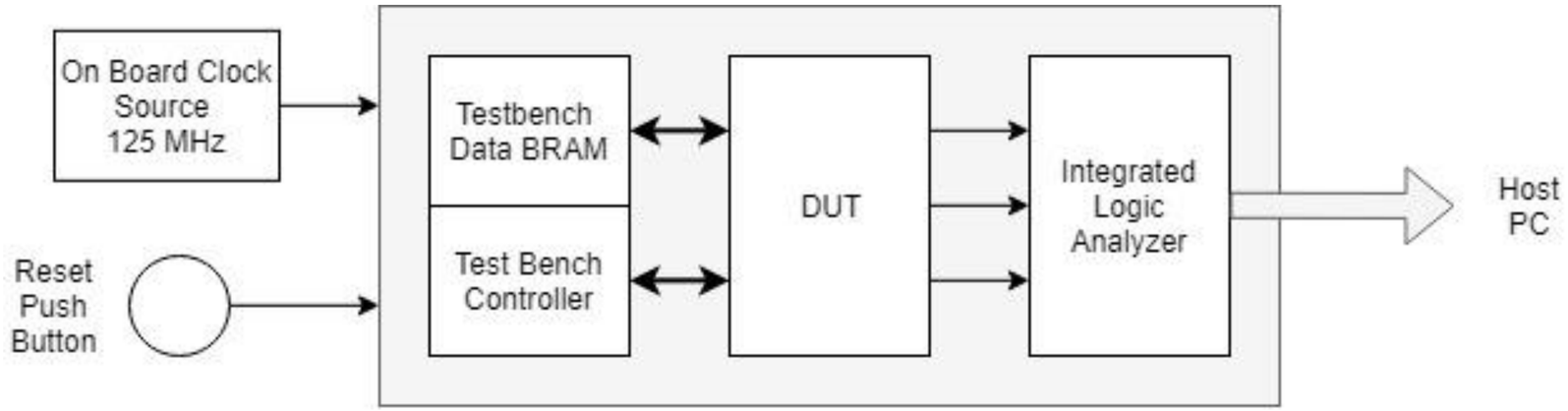
Performance Variation with the Latency of Units





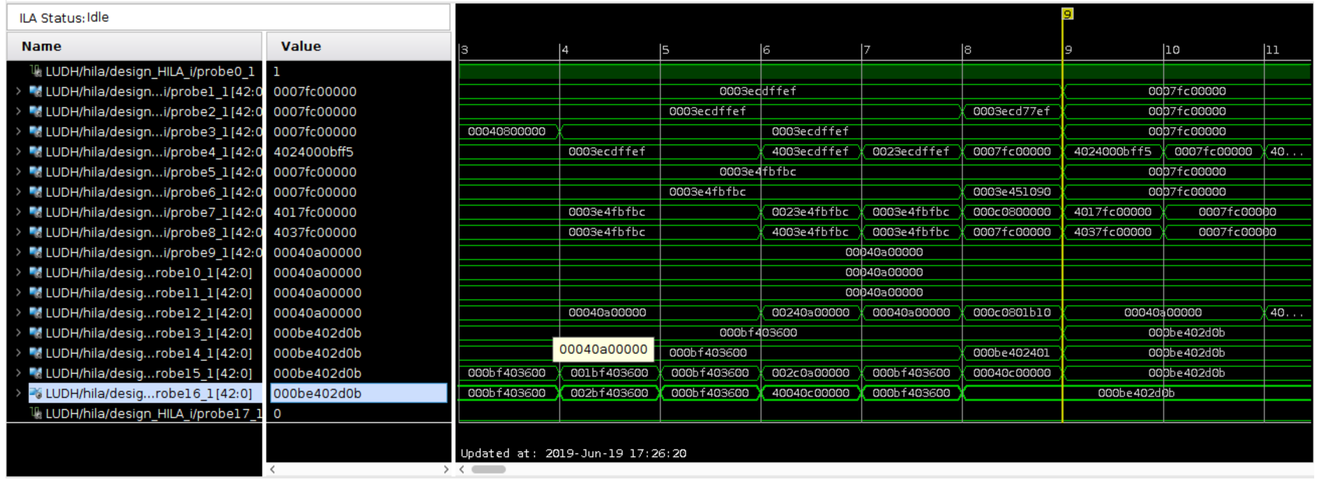


Hardware Testing

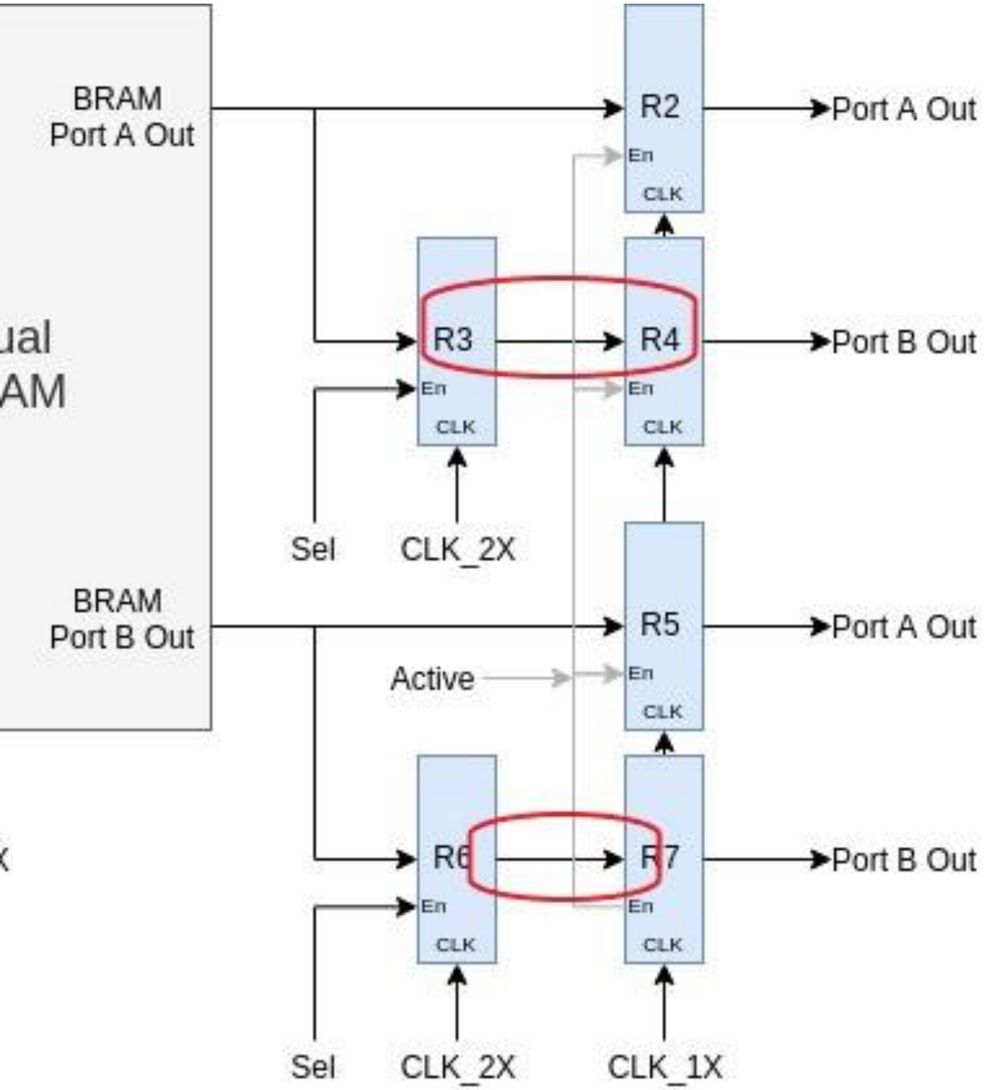


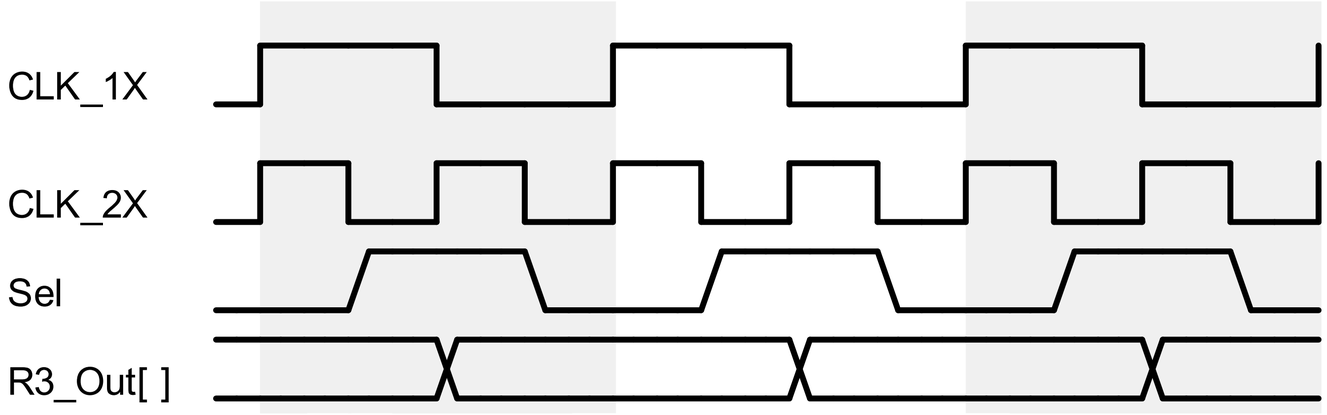
ILA ports are connected to the inputs and outputs of the BRAM ports

Hardware Configuration Parameter	Value
Number of PEs	4
Number of BRAMs	4
Number of Ports/ BRAM	4
Read Latency of BRAM	2
Latency of MAC Units	20
Latency of Diver Units	31



False Hold Violation





The path between R3 and R4 is not triggered at the rising edge of the CLK_1X hence the hold violation at this edge must be ignored

Future Work

- Synthesize BRAM
- Integrate with the on-chip ARM core
- New scheduling strategies

Thank you

Future Work

- Synthesize BRAM
- Port the tool on for on-chip ARM core
- New scheduling strategies

Parallelizing the Sparse LU Decomposition