DSD Midterm Project

Gauss-Seidel Iteration Machine

Speaker：Alan

Advisor：Prof. An-Yeu Wu

Date：2022/03/22

Background

- Large linear system of equations is required to be solved in many engineering simulations and scientific computing applications
- Several iterative methods is used to accelerate the computing due to their simplicity, such as Jacobi method, **Gauss-Seidel iteration model (GSIM)**, Conjugate gradient…

$$\begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} \Longrightarrow \begin{array}{c} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N = b_2 \\ \vdots \\ a_{N1}x_1 + a_{N2}x_2 + \cdots + a_{NN}x_N = b_N \end{array}$$

Gauss-Seidel Iteration Model (GSIM)

- **Iterative method** to solve a linear system of equations

- **Ax = b** $\Longrightarrow$

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b}=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \qquad (1)$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N &= b_2 \\ &\vdots \\ a_{N1}x_1 + a_{N2}x_2 + \cdots + a_{NN}x_N &= b_N \end{aligned}$$

- Expand eq (1)

$$\begin{aligned} x_1^1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^0 - \cdots - a_{1N}x_N^0) \\ x_2^1 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^1 - a_{23}x_3^0 - \cdots - a_{2N}x_N^0) \\ &\vdots \\ x_N^1 &= \frac{1}{a_{NN}}(b_N - a_{N1}x_1^1 - a_{N2}x_2^1 - \cdots - a_{NN-1}x_{N-1}^1) \end{aligned}$$

- Change the order

x_3^0: initial value of x_3

x_2^1: first iteration result of x_2

$$x_i^{k+1} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{N} a_{ij}x_j^{k}\right]$$

- Final equation

Project Problem

- Given a fixed matrix **A**
- Input Different matrix **b**
- After *k* iterations (define by yourself!), output final results **x**

$$A = \begin{bmatrix}
20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20 & -13 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & -13 & 20
\end{bmatrix}$$

At most 7 non-zero terms one time

By the previous equation:

$$\begin{aligned}
x_1^1 &= \frac{1}{20}[b_1+13(x_2^0+0)-6(x_3^0+0)+(x_4^0+0)] \\
x_2^1 &= \frac{1}{20}[b_2+13(x_3^0+x_1^1)-6(x_4^0+0)+(x_5^0+0)] \\
x_3^1 &= \frac{1}{20}[b_3+13(x_4^0+x_2^1)-6(x_5^0+x_1^1)+(x_6^0+0)] \\
x_4^1 &= \frac{1}{20}[b_4+13(x_5^0+x_3^1)-6(x_6^0+x_2^1)+(x_7^0+x_1^1)] \\
&\vdots \\
x_{16}^1 &= \frac{1}{20}[b_{16}+13(0+x_{15}^1)-6(0+x_{14}^1)+(0+x_{13}^1)]
\end{aligned}$$

Only divide 20

Score Criteria

- 評分一：Error rate E^2
  - $E^2 = \sum_{i=1}^{16} \sum_{j=1}^{16} (a_{ij} x_j^k - b_i)^2$

A 級：			E^2	< 0.000001
B 級：	0.000001	≦	E^2	< 0.000005
C 級：	0.000005	≦	E^2	< 0.000010
D 級：	0.000010	≦	E^2	< 0.000050
E 級：	0.000050	≦	E^2	< 0.000100
F 級：	0.000100	≦	E^2	< 0.001000
G 級：	0.001000	≦	E^2	< 0.005000
H 級：	0.005000	≦	E^2	< 0.010000
I 級：	0.010000	≦	E^2	< 0.100000
J 級：	0.100000	≦	E^2	< 0.300000
K 級：	0.300000	≦	E^2	

- 評分二：AT score
  - $AT = area \times total\ timing$
  - Area: synthesis cell area
  - Timing: total execution time (tb1+tb2+…+tb5)

```
--------------------------------------------------

Your Score Level: A

Congratulations! GSIM's Function Successfully!

------------------------PASS-----------------------

Simulation complete via $finish(1) at time 3734500 PS + 0
./testfixture5.v:213       #(`CYCLE/2); $finish;
ncsim> exit
```

Combinational area:	3875.164193
Buf/Inv area:	434.534396
Noncombinational area:	1147.442383
Macro/Black Box area:	0.000000
Net Interconnect area:	48580.242432
Total cell area:	5022.606576
Total area:	53602.849008

Design Guidelines

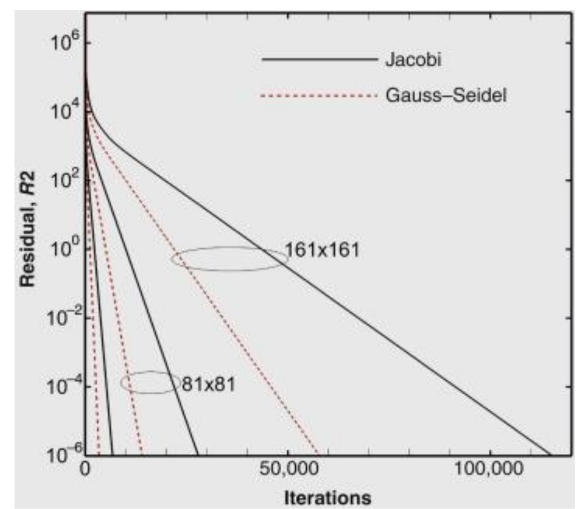
- Algorithm level
  - Mixture with other algorithms (i.e. Jacobi Method)
  - Relaxation for iterative processing
- Architecture level
  - Data path scheduling
  - Parallel processing (unfolding)
  - Pipelining
- Computation unit level
  - Constant multiplier, constant divider
  - Decimal analysis

Algorithm Level Optimization (1/2)

Gauss-Seidel: $x_i^{k+1} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{N} a_{ij} x_j^{k}\right]$ $\Rightarrow$ Data dependent

Jacobi: $x_i^{k+1} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k} - \sum_{j=i+1}^{N} a_{ij} x_j^{k}\right]$ $\Rightarrow$ Not data dependent

	Jacobi	Gauss-Seidel
Update $\boldsymbol{x}^{(k+1)}$	Simultaneously	Sequentially
Parallelable	No	Yes
Iteration count	More (about 2x)	Less
Computation time	Less (about 0.5x)	More



Algorithm Level Optimization (2/2)

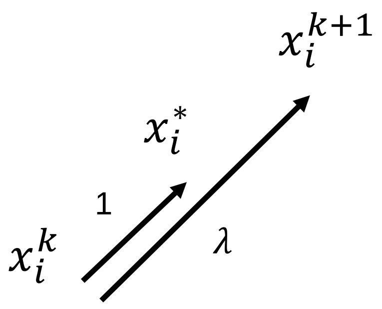
- Iteration relaxation (Nesterov gradient descent)
  - Motivation: speed up convergence

w/o relaxing: $$x_i^{k+1} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{N} a_{ij}x_j^{k}\right]$$

w/ relaxing: $$x_i^{*} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^{k}\right]$$

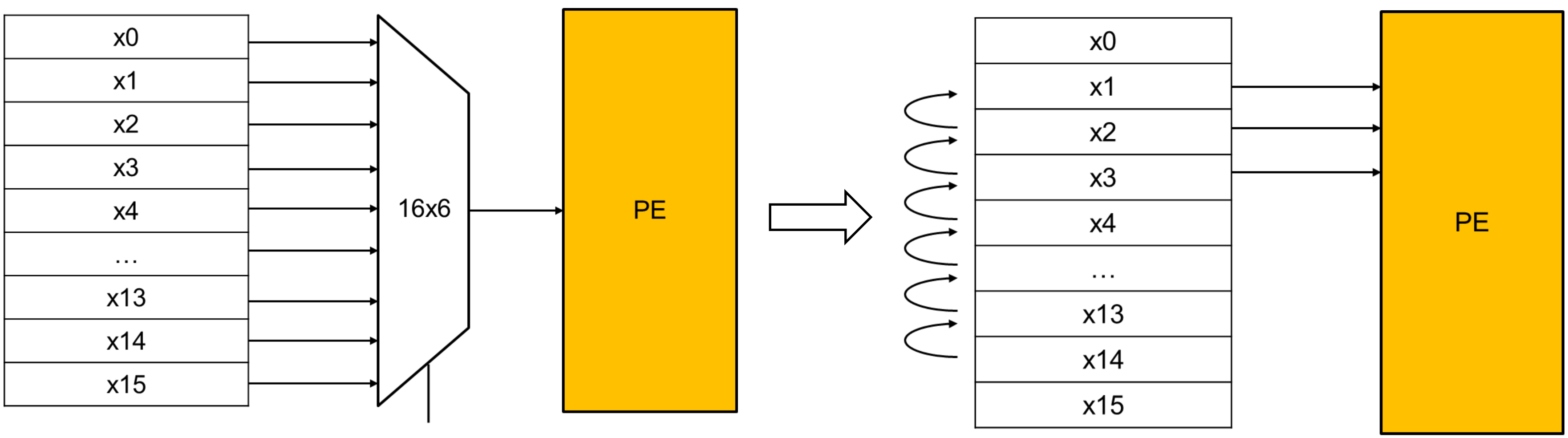
$$x_i^{k+1} = \lambda x_i^{*} + (1-\lambda)x_i^{k}$$

- Relaxation factor λ
  - $0 < \lambda < 2$
  - Under-relaxation: $0 < \lambda < 1$
    - Make non-convergent systems converge
  - Over-relaxation: $1 < \lambda < 2$
    - Speed up convergence of a convergent system



Architecture Level Optimization (1/3)

- Reading data
  - Arbitrary reading: Using several MUXs to load data
  - Structural reading: Similar computation dataflow



Scheduling by removing 6 MUXs

Architecture Level Optimization (2/3)

- Parallel processing (unfolding)
  - Using multi-core to compute
  - Necessity: No data dependancy
- Reordering computation
  - Processing elements: 1, 5, 9, 13, …

20	-13	6	-1	0	0	0	0	0	0	0	0	0	0	0	0
-13	20	-13	6	-1	0	0	0	0	0	0	0	0	0	0	0
6	-13	20	-13	6	-1	0	0	0	0	0	0	0	0	0	0
-1	6	-13	20	-13	6	-1	0	0	0	0	0	0	0	0	0
0	-1	6	-13	20	-13	6	-1	0	0	0	0	0	0	0	0
0	0	-1	6	-13	20	-13	6	-1	0	0	0	0	0	0	0
0	0	0	-1	6	-13	20	-13	6	-1	0	0	0	0	0	0
0	0	0	0	-1	6	-13	20	-13	6	-1	0	0	0	0	0
0	0	0	0	0	-1	6	-13	20	-13	6	-1	0	0	0	0
0	0	0	0	0	0	-1	6	-13	20	-13	6	-1	0	0	0
0	0	0	0	0	0	0	-1	6	-13	20	-13	6	-1	0	0
0	0	0	0	0	0	0	0	-1	6	-13	20	-13	6	-1	0
0	0	0	0	0	0	0	0	0	-1	6	-13	20	-13	6	-1
0	0	0	0	0	0	0	0	0	0	-1	6	-13	20	-13	6
0	0	0	0	0	0	0	0	0	0	0	-1	6	-13	20	-13
0	0	0	0	0	0	0	0	0	0	0	0	-1	6	-13	20

Core 1

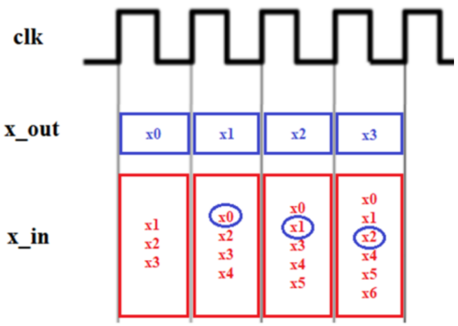
Core 2

20	0	0	0	-13	0	0	0	6	0	0	0	-1	0	0	0
0	20	0	0	-1	-13	0	0	6	6	0	0	-13	-1	0	0
0	0	20	0	0	-1	-13	0	0	6	6	0	0	-13	-1	0
0	0	0	20	0	0	-1	-13	0	0	6	6	0	0	-13	-1
-13	-1	0	0	20	0	0	0	-13	0	0	0	6	0	0	0
0	-13	-1	0	0	20	0	0	-1	-13	0	0	6	6	0	0
0	0	-13	-1	0	0	20	0	0	-1	-13	0	0	6	6	0
0	0	0	-13	0	0	0	20	0	0	-1	-13	0	0	6	6
6	6	0	0	-13	-1	0	0	20	0	0	0	-13	0	0	0
0	6	6	0	0	-13	-1	0	0	20	0	0	-1	-13	0	0
0	0	6	6	0	0	-13	-1	0	0	20	0	0	-1	-13	0
0	0	0	6	0	0	0	-13	0	0	0	20	0	0	-1	-13
-1	-13	0	0	6	6	0	0	-13	-1	0	0	20	0	0	0
0	-1	-13	0	0	6	6	0	0	-13	-1	0	0	20	0	0
0	0	-1	-13	0	0	6	6	0	0	-13	-1	0	0	20	0
0	0	0	-1	0	0	0	6	0	0	0	-13	0	0	0	20

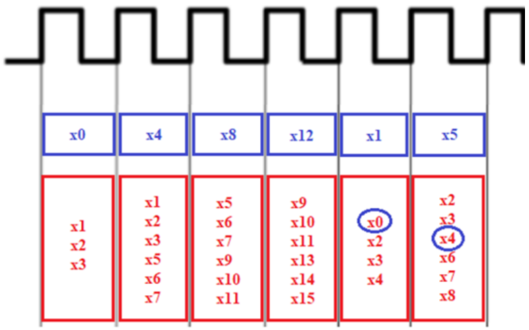
rescheduling

Architecture Level Optimization (3/3)

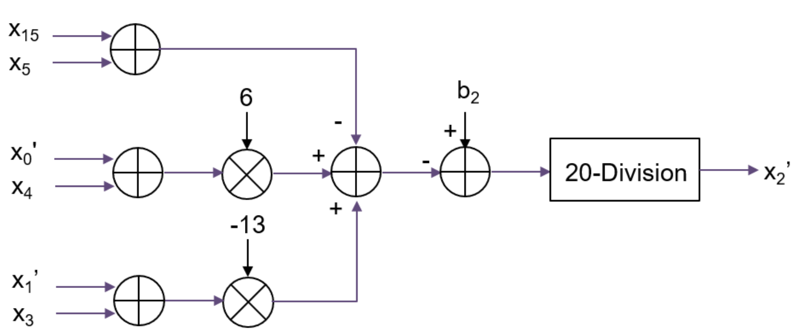
- Pipelining
  - Divide computation into several cycles

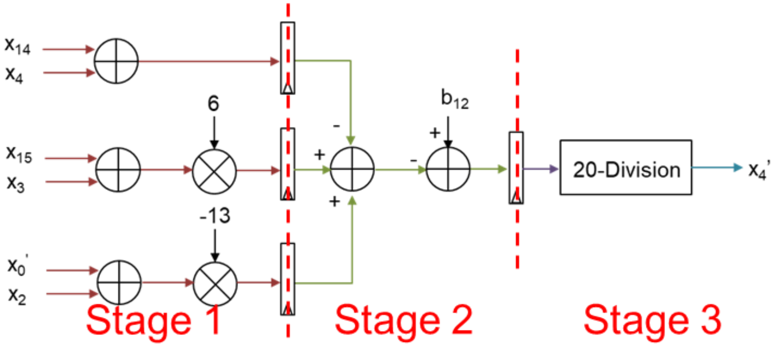


re-scheluding
& pipelining



e.g. 3-stages pipeling





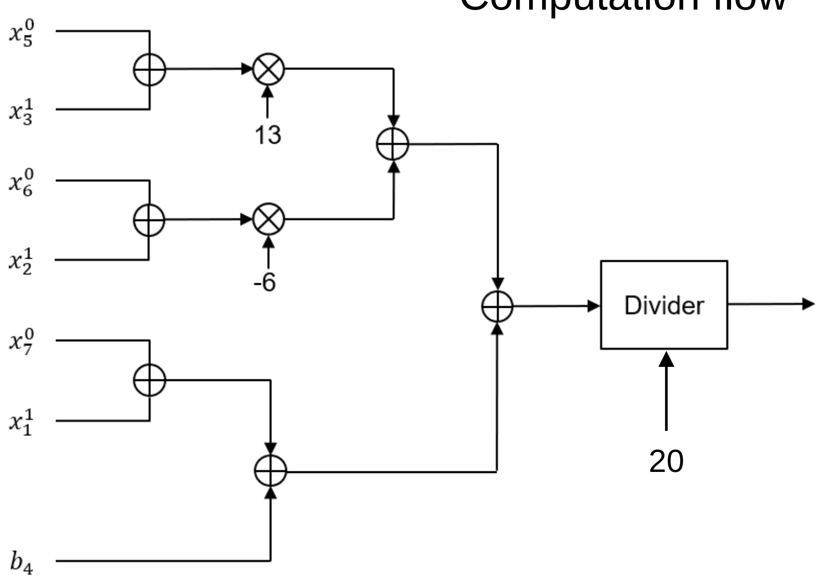
Computation Unit Level (1/2)

1 Division 2 Multiplication

$$Ex:\ x_4^1 = \frac{1}{20}\left[b_4 + 13\times(x_5^0+x_3^1)+(-6)\times(x_6^0+x_2^1)+(x_7^0+x_1^1)\right]$$

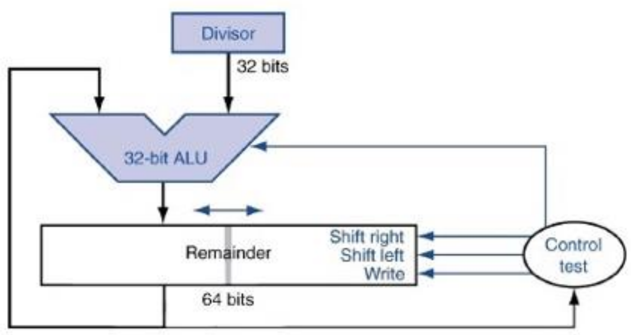
Division and multiplication is complicated.
It need large area and long computation time.

Computation flow



Computation Unit Level (2/2)

- Multiplier
  - Power-of-two method (e.g. $6 = 110_2$, $13 = 1101_2$)
- Divider
  - Conventional (for arbitrary input): requires 32 cycles
  - Constant divider: requires 3 cycles
    - Canonic Signed Digit (CSD) code
    - For each odd integer d, there exists an odd integer m such that $d \times m = 2^n - 1$



$$\frac{1}{d} = \frac{m}{2^n - 1} = \frac{m}{2^n(1 - 2^{-n})}$$

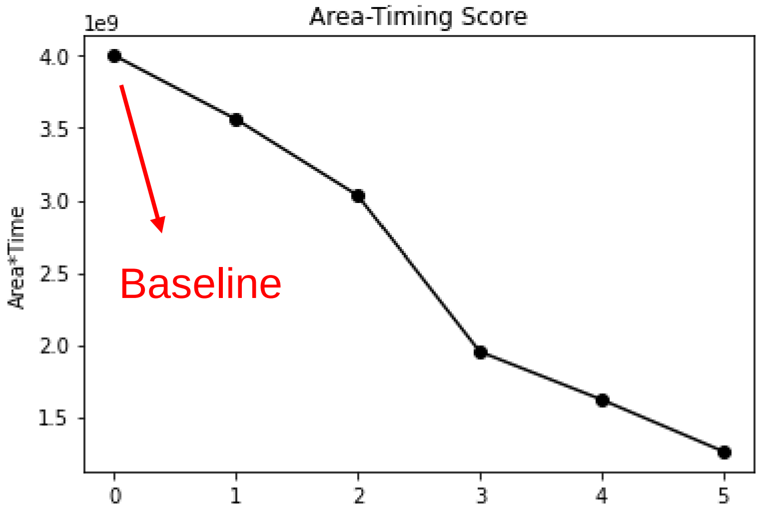
$$= \frac{m}{2^n}(1 + 2^{-n})(1 + 2^{-2n})(1 + 2^{-4n})\ldots$$

Need: 1 Adder, 1 shifter
Time: 32 cycles

Need: 1 Adder, 1 shifter
Time: 3 cycles

Notification

- Deadline: 4/5 23:59
  - Submit your result by group
  - Submit RTL file, SYN file (.sdf, .v files)
- Midterm presentation
  - Date: TBD
  - Materials: Your optimization towards improvements



Baseline: 4.0×10^9

We will score by performance of each team