Harnessing CIM techniques for accelerating sum operations in FPGA-DRAM architectures

NATIONAL TECHNICAL UNIVERSITY OF ATHENS

School of Electrical and Computer Engineering

Microprocessors and Digital Systems Lab

Supervisor: D. Soudris
Co-Supervisor: G. Lentaris

Theocharis Diamantidis

Context

Introduction

Background & Related Work

Proposed Solution & Methodology

Experiments & Results

Conclusions & Future Work

Introduction

Modern Systems are Processor-Centric

- ❖ Repeated Data Movement between CPU & Memory
- ❖ High Energy Consumption
- ❖ Reduced Speed

"Memory Wall”

- ❖ Multicore Architectures
- ❖ Memory Latency Bottleneck

Near Memory Computing

Computation is moved near memory

Compute In Memory

Computation is executed directly in memory

Introduction

Compute In Memory

Computation is executed directly in memory

Computations in Commercial DRAM chips

- Previous work already show simple logical operations (AND/OR/NOT/ROW COPY)
  - e.g Group Safari / O. Mutlu [1] [2] [3]
- Current Thesis / Motivation:
  It is possible to develop extensions towards more complex logic/arithmetic functions
  - in already available chips, in the field

[1] Safari, Yuksel, I E PULSAR: Simultaneous Many-Row Activation for Reliable and High-Performance Computing in Off-the-Shelf DRAM Chips

[2] Safari, Yuksel, I E Functionally-Complete Boolean Logic in Real DRAM Chips: Experimental Characterization and Analysis

[3] Safari Nastanar Hajinazar SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM

Contributions

1. Simulation Level approximation of the timing Constraints

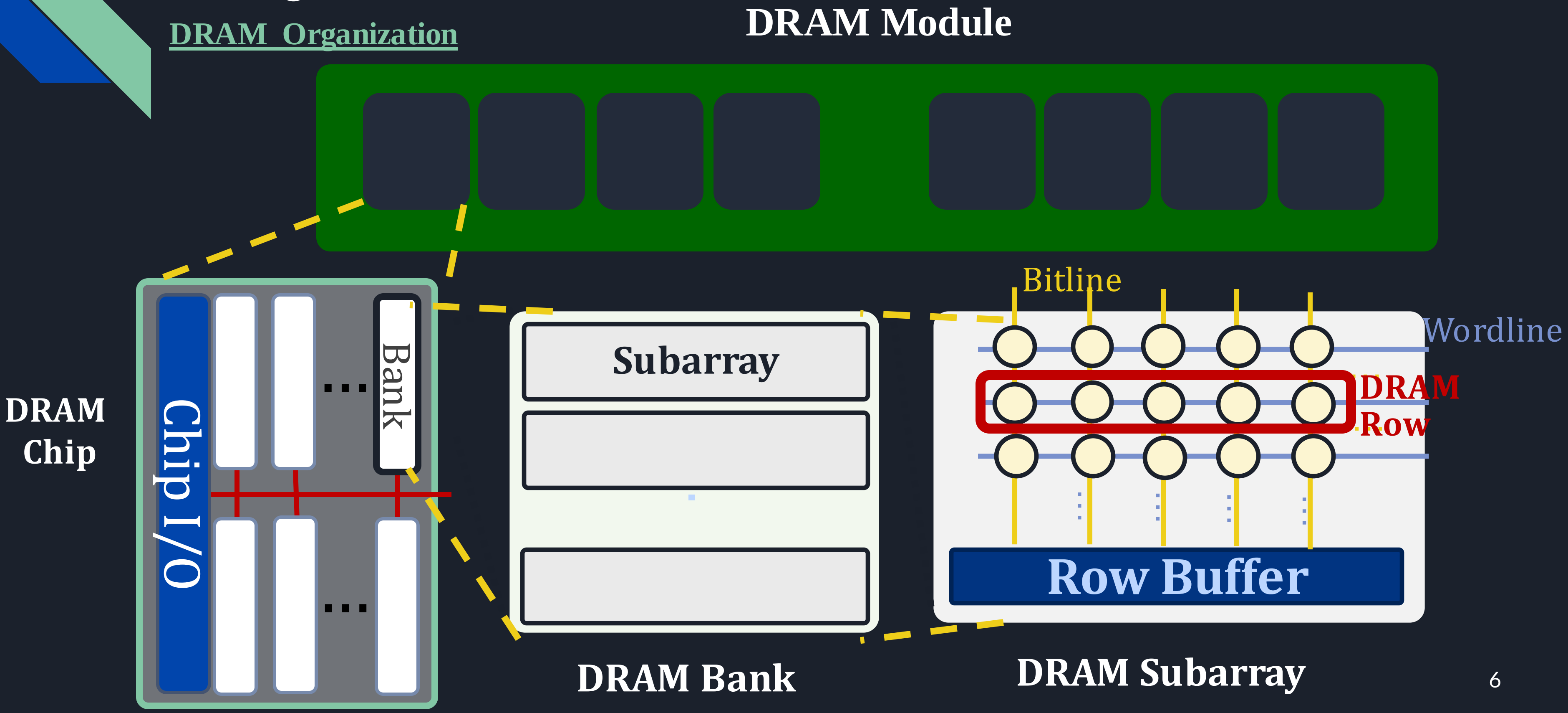
2. Developing a Full-Adder with 1, 2, 4, 8, 16 bits on actual commercial DRAM HW

3. Proof of Concept Function (Sum) with proposed FPGA-DRAM co-processing approach

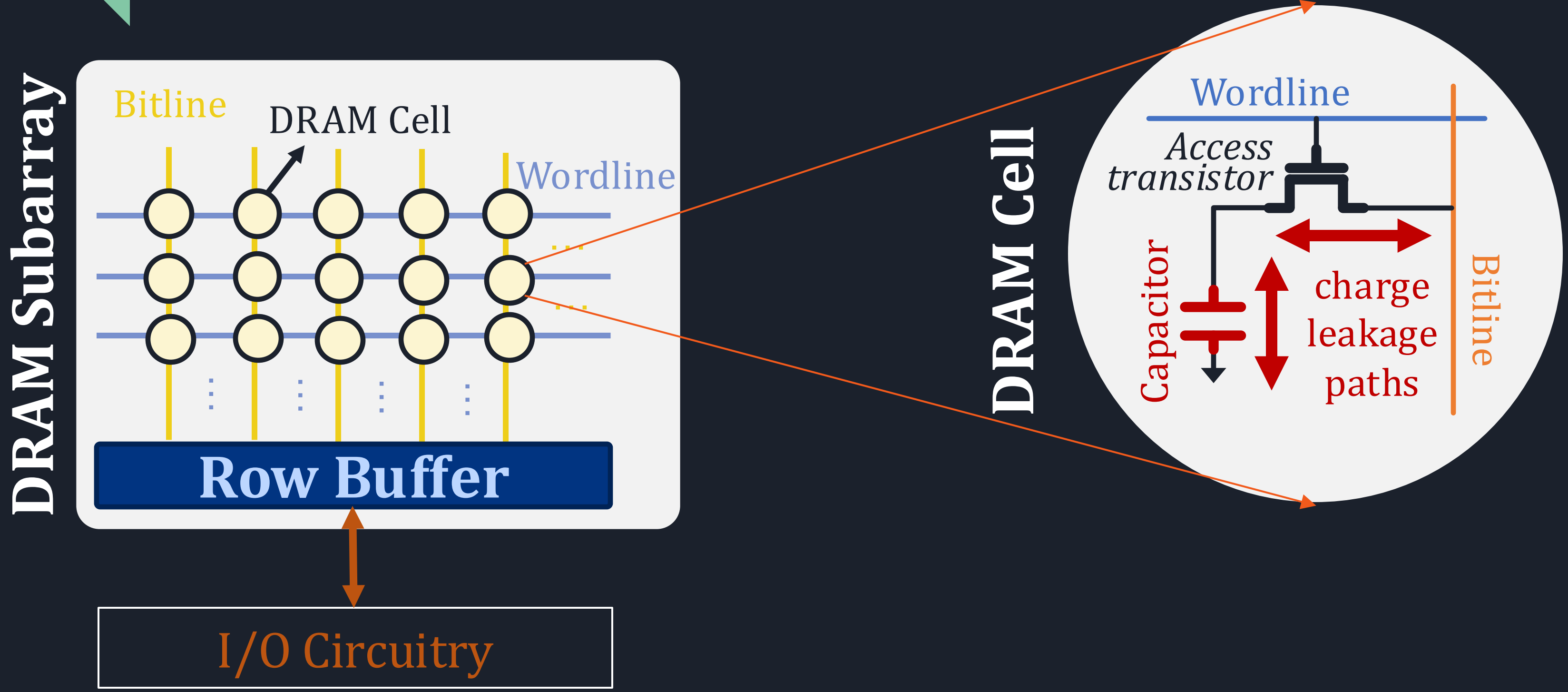
4. Evaluation of Success Rate and Acceleration factors for multiple scenarios of logical operations

Background

DRAM Organization



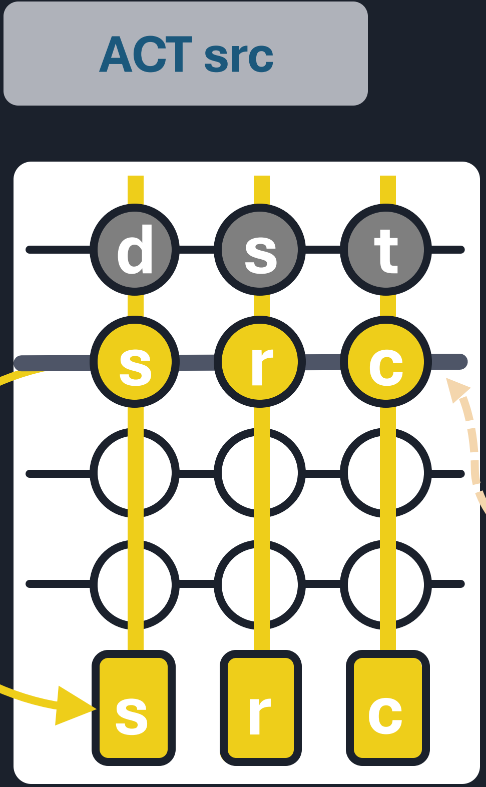
Background


DRAM Subarray
Bitline
DRAM Cell
Wordline
Row Buffer
I/O Circuitry
DRAM Cell
Wordline
Access transistor
Capacitor
charge leakage paths
Bitline

Related Work

ROW-COPY

Fetch src's content into the sense amplifiers



Enabe the src's wordline

Related Work

ROW-COPY

No time violation

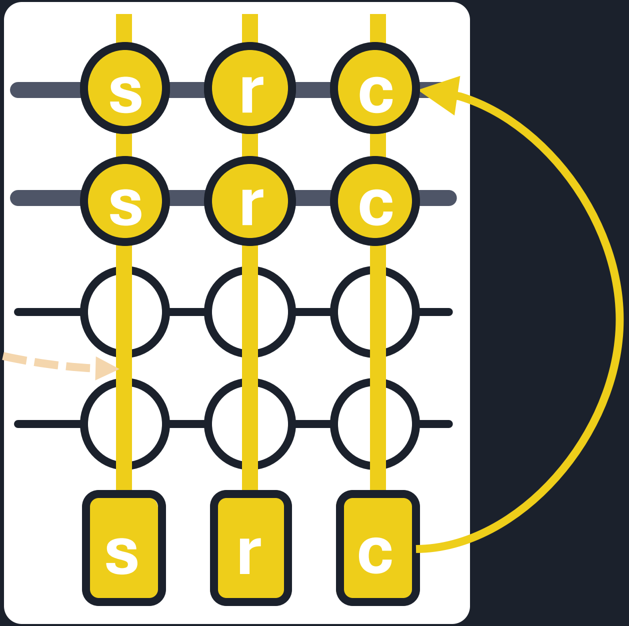
Time Violation

ACT src

PRE

ACT dst

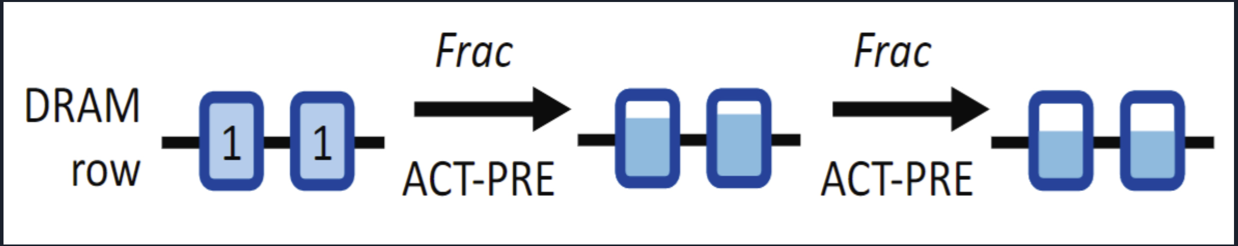
Bitlines are still at src voltage

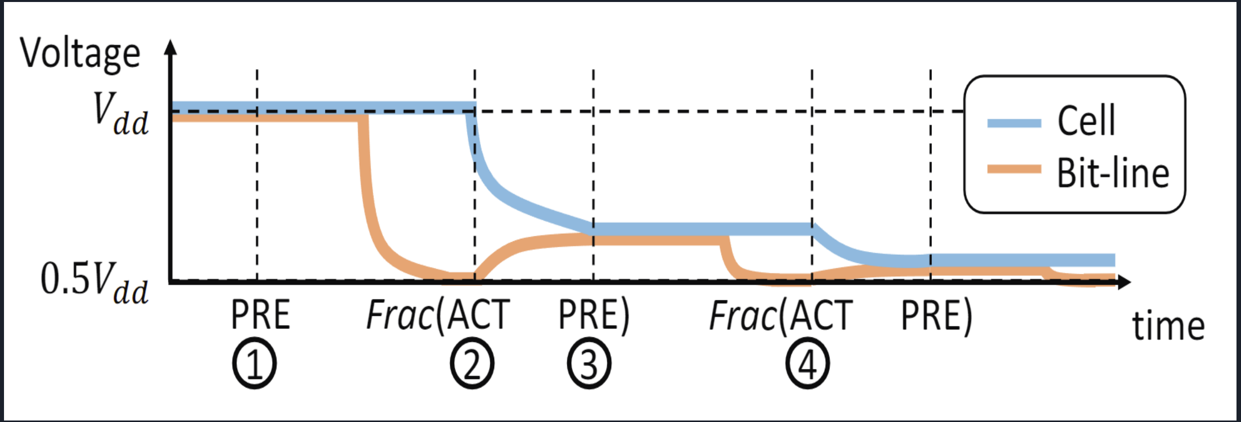


src's content is copied to dst

Related Work

FRAC Operation





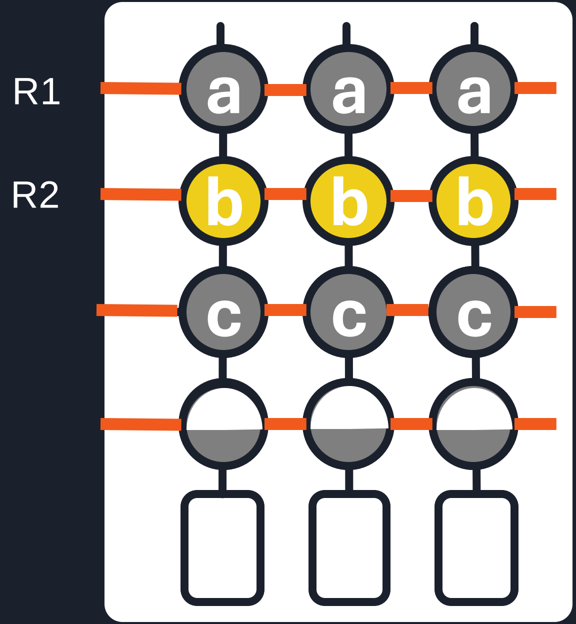
Related Work

LOGICAL AND/OR

Time violation

Time violation

Frac Op. | ACT R1 | PRE | ACT R2



Related Work

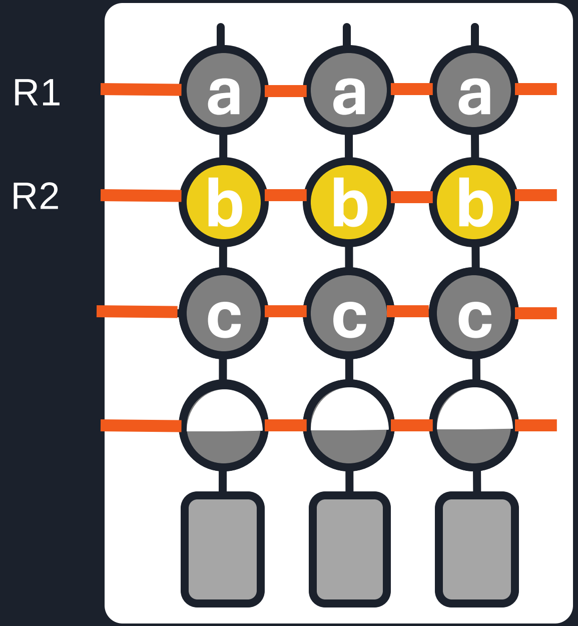
LOGICAL AND/OR

Time violation

Time violation

Frac Op. | ACT R1 | PRE | ACT R2

MAJ3 (a,b,c)



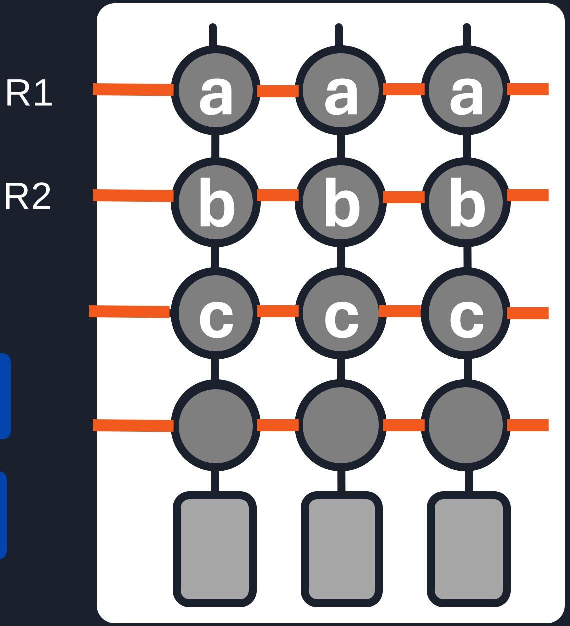
Related Work

LOGICAL AND/OR

Frac Op. → ACT R1 → PRE → ACT R2

Time violation (ACT R1 – PRE)

Time violation (PRE – ACT R2)



MAJ (A, B, C)

MAJ (A, B, 0) = AND (A, B)

MAJ (A, B, 1) = OR (A, B)

Proposed Solutions

Limitation	Corresponding Solution
Bit transfer (i.e. shifting, propagation of carry out) between columns is prohibited.	Column-major architectures, where all numbers and their bits are stored in the same bit-line.
NOT operation cannot be executed in the same sub-array.	Store the operand and the complementary one and (if necessary) calculate the complementary result in each step
Logical operations for the same full-adder cannot be conducted in parallel.	Use bit-serial arithmetic and exploit the group of rows as much as possible.
The number of operands is restricted by the number of rows in the sub-array.	Utilize co-processing in memory and in the FPGA.
The results of each operation overwrites all the rows in the group.	Execute each operation in different groups of rows.

Methodology & Design Space Exploration for improving Accuracy and Speed

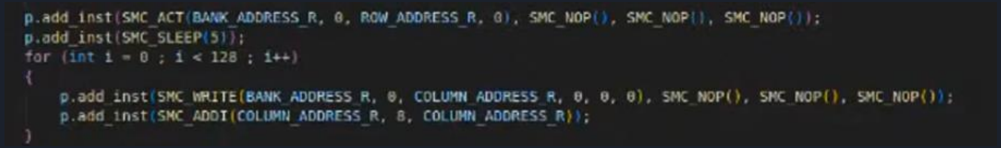
- ❖ Estimate the timing parameters to violate on a simulation level.
- ❖ Identifying the boundaries of the sub-array on actual hardware level
- ❖ Exploring Simultaneously Opening-Rows
- ❖ Activate different groups of rows to achieve more accurate results.
- ❖ Testing timing parameters between commands.
- ❖ Evaluate the in-Memory Full-Adder

DSE parameters: Banks, sub-arrays, group of rows, bit-lines, bit width, numbers per column, total numbers per sum, DRAM command timing, operands for logical operations.

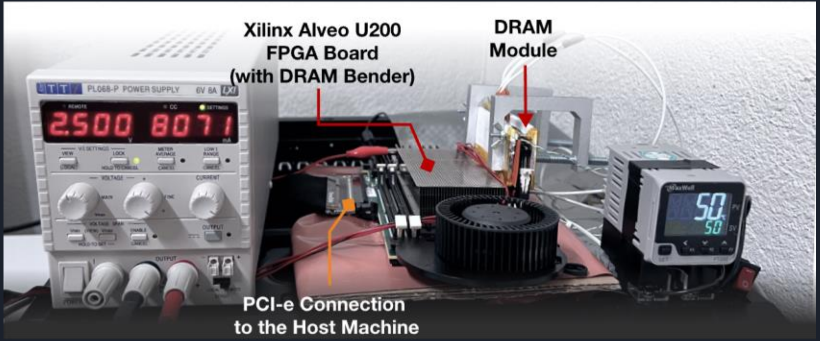
Setup

Experiment Setup

- ❖ Xilinx Alveo U200 FPGA Board (200MHz).
- ❖ Evaluation was conducted on DDR4 memories.
- ❖ DRAM Bender HW → infrastructure with fine-grained control over DRAM.
- ❖ DRAM Bender SW →Framework facilitating memory access flexibility.
    - ❖ Add commands into a list (Control flow, Arithmetic, Memory) with high-level programming languages (C++) and then schedule the DDR commands into the DRAM.



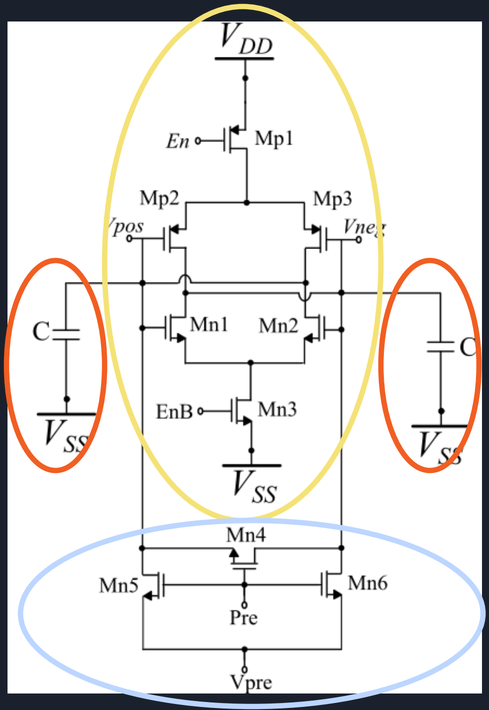
```
p.add_inst(SMC_ACT(BANK_ADDRESS_R, 0, ROW_ADDRESS_R, 0), SMC_NOP(), SMC_NOP(), SMC_NOP());
p.add_inst(SMC_SLEEP(5));
for (int i = 0 ; i < 128 ; i++)
{
    p.add_inst(SMC_WRITE(BANK_ADDRESS_R, 0, COLUMN_ADDRESS_R, 0, 0, 0), SMC_NOP(), SMC_NOP(), SMC_NOP());
    p.add_inst(SMC_ADDI(COLUMN_ADDRESS_R, 8, COLUMN_ADDRESS_R));
}
```



A. Olgun"DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

Experiments & Results

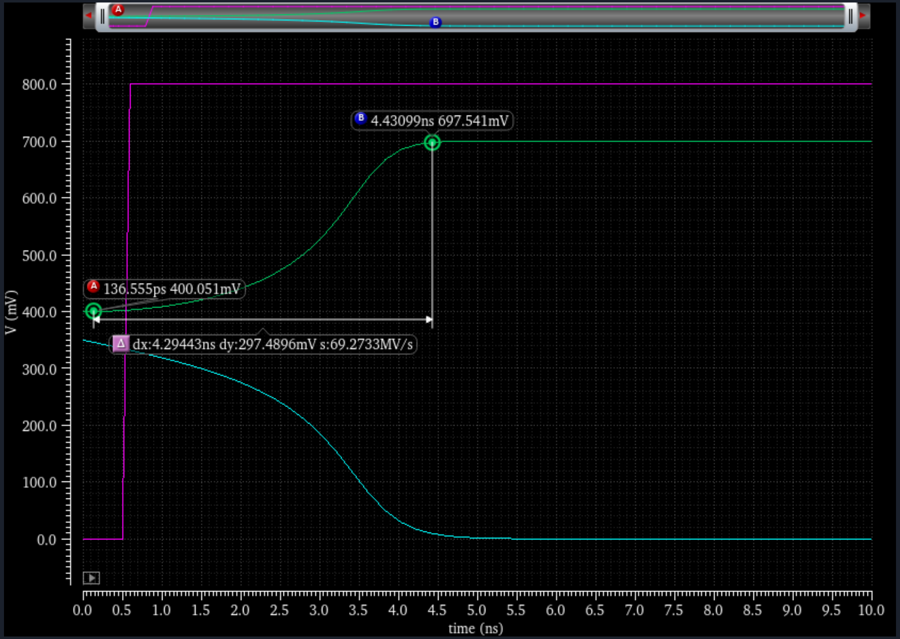
Estimate the timing parameters to violate on a simulation level.



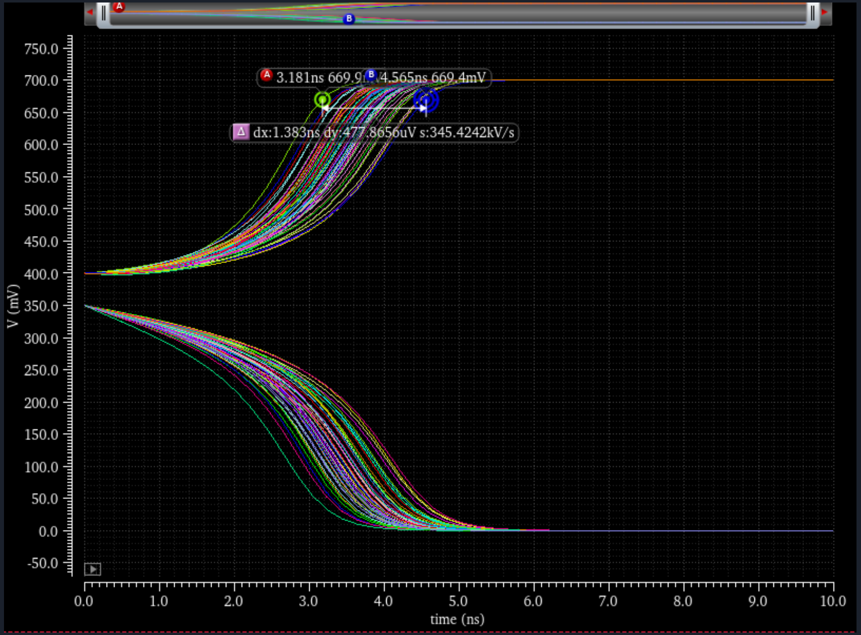
- Capacitors equal to the parasitic capacitance of bit-line.
- The inverters have a positive feedback. With small deviations the output is driven to Vdd or ground.
- Precharge Circuit. The Voltage that is precharged must be without noise.

Experiments & Results

Estimation of the Timing Parameters



Monte Carlo Analysis

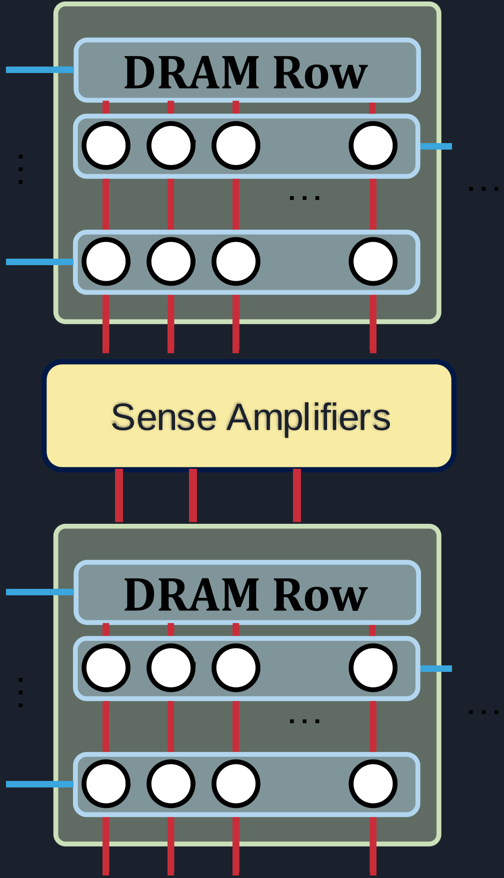


Experiments & Results

Identifying the boundaries of the sub-array

We implement the Row-Copy

1. Zeroing the whole memory
2. Write all ones in a random row R1
3. Issue a row-copy command Act(R1)-Pre-Act(R2)
4. Read the data in row R2
5. Loop for rows R2

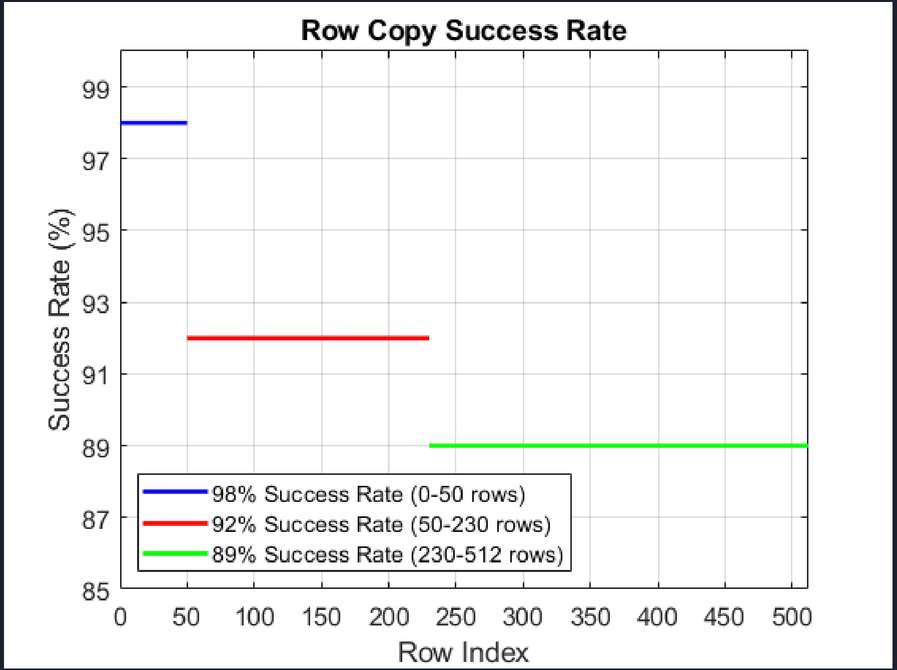


Experiments & Results

Results

512 rows in each sub-array

32 sub-arrays in each bank



Success Rate in Row Copy depends:

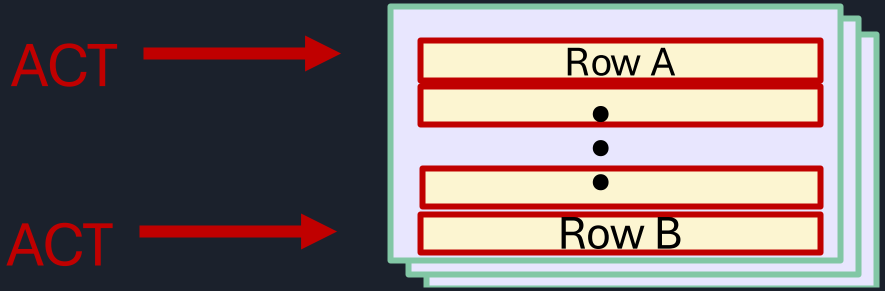
- Data Stored in Cells
- The Row Index

Experiments & Results

Exploring Simultaneously Opening-Rows

Activating two rows in **quick succession** can **simultaneously** activate **many rows in a subarray**

ACT Row A → PRE → ACT Row B → WR



If rows are activated, WR command overwrites all of the activated rows' content

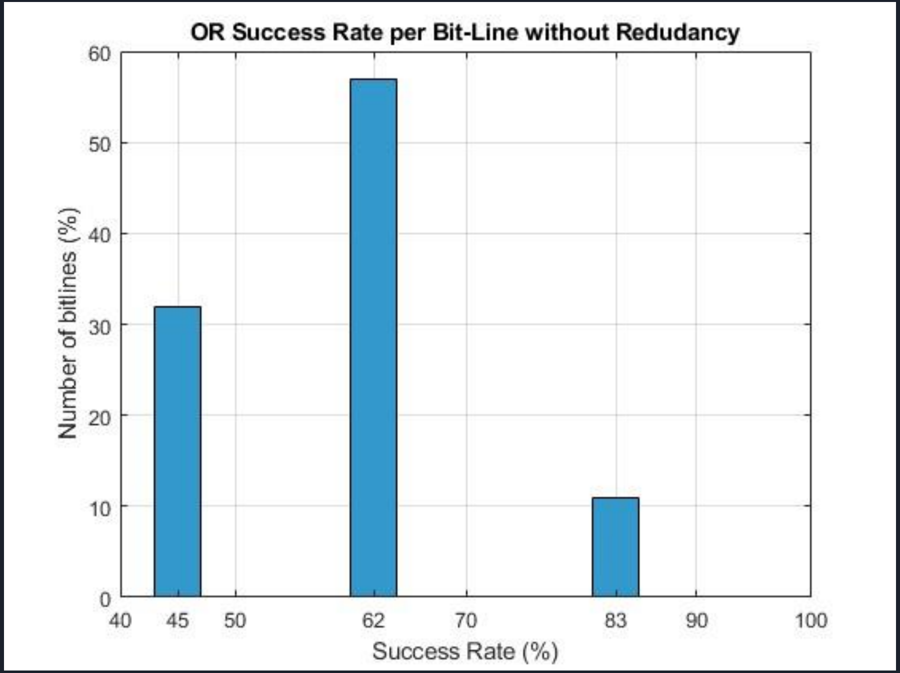
Results: 2, 4, 8, 16 rows can be activated simultaneously

Experiments & Results

Next: study single AND/OR operations for the given DRAM chip

1. DSE for optimal timing
   - Idle cycles between ACT-PRE-ACT equal 0
   - Randomness of bit values lead to worse results

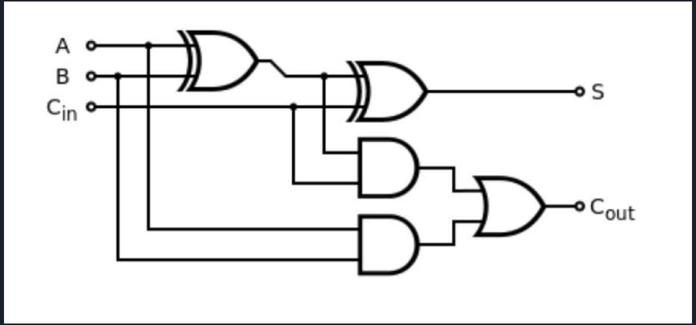
2. Evaluation of Success Rate
   - A subset of columns (~6000) have higher success rate without redundancy.



The results are almost the same for AND/OR

Experiments & Results

Next: 1-bit Full Adder



Step 1: Store operands in the same Bit-Line

Step 2: Store the complementary values

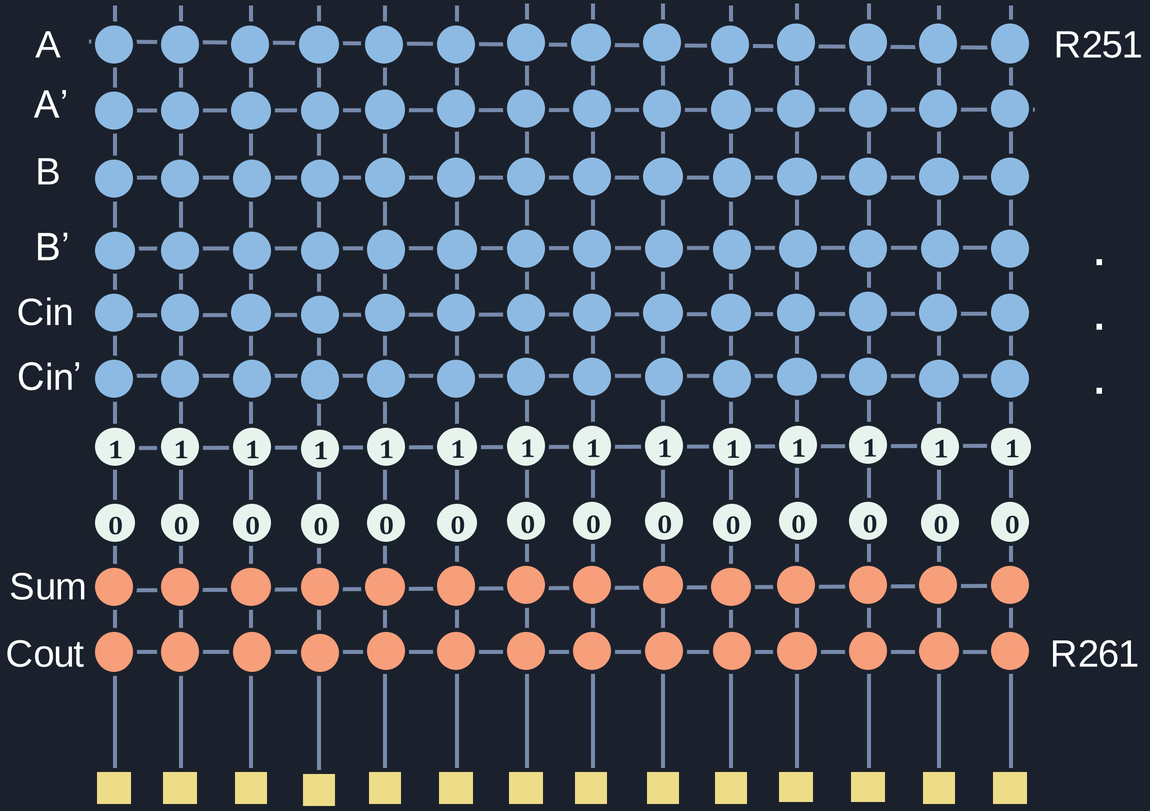
Step 3: Choose how many groups of rows will be used

Step 4: Execute the logical operations in these groups

Step 5: Store the result in another row

Experiments & Results

1-bit Full Adder

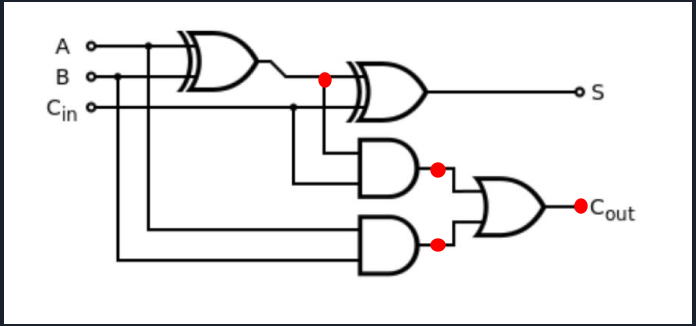


Experiments & Results

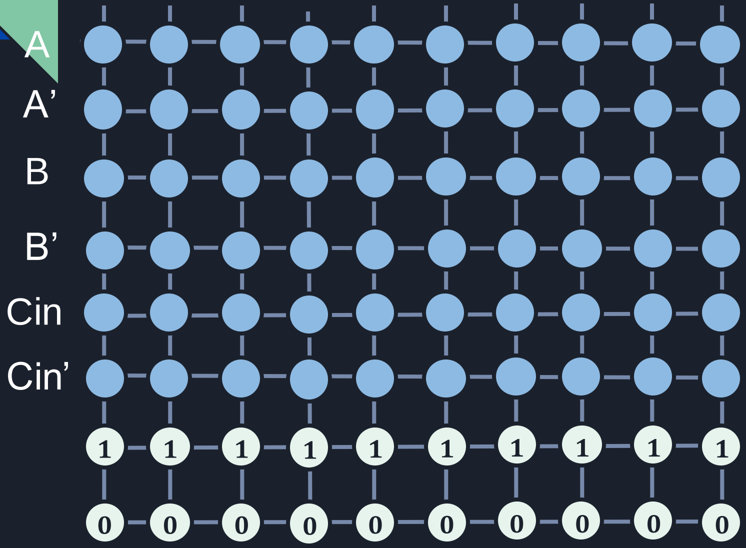
1-bit Full Adder

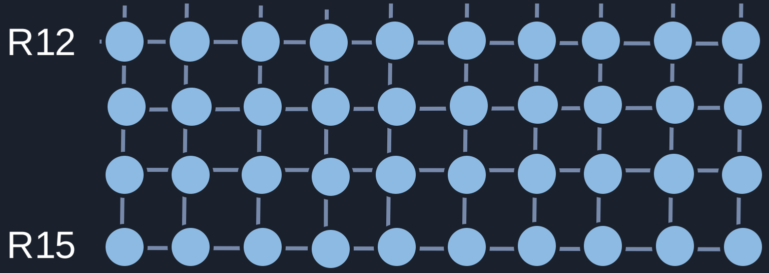
Cout will be used as input in the next bits

We need 15 groups of 4 rows in total

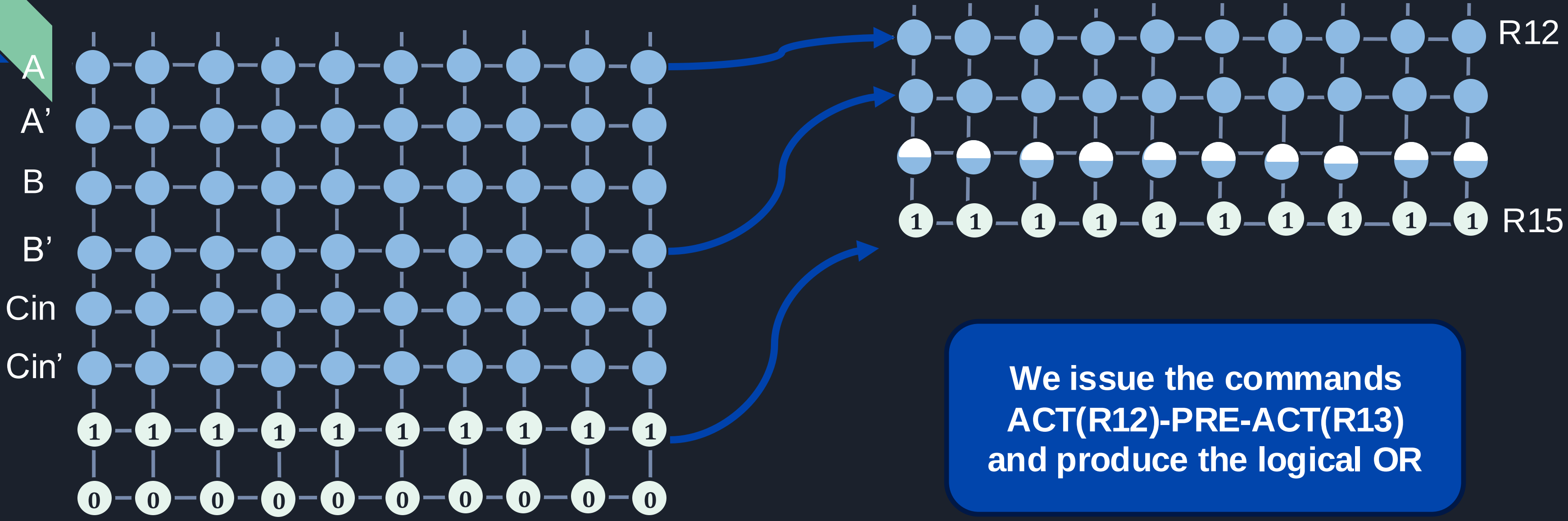


Experiments & Results



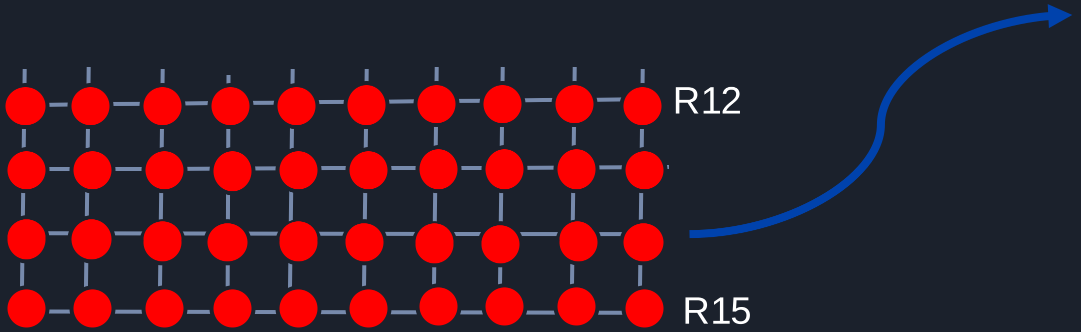


Experiments & Results



We issue the commands ACT(R12)-PRE-ACT(R13) and produce the logical OR

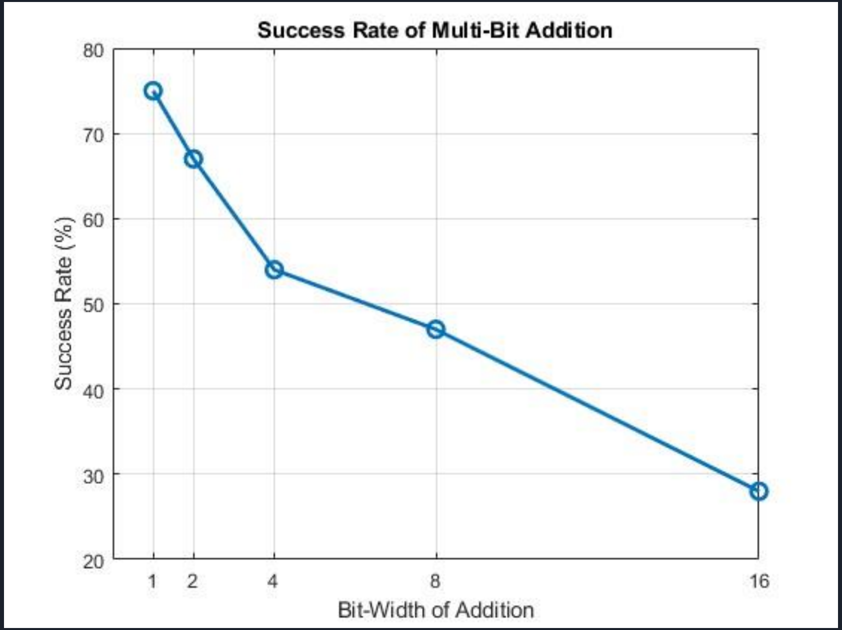
Experiments & Results

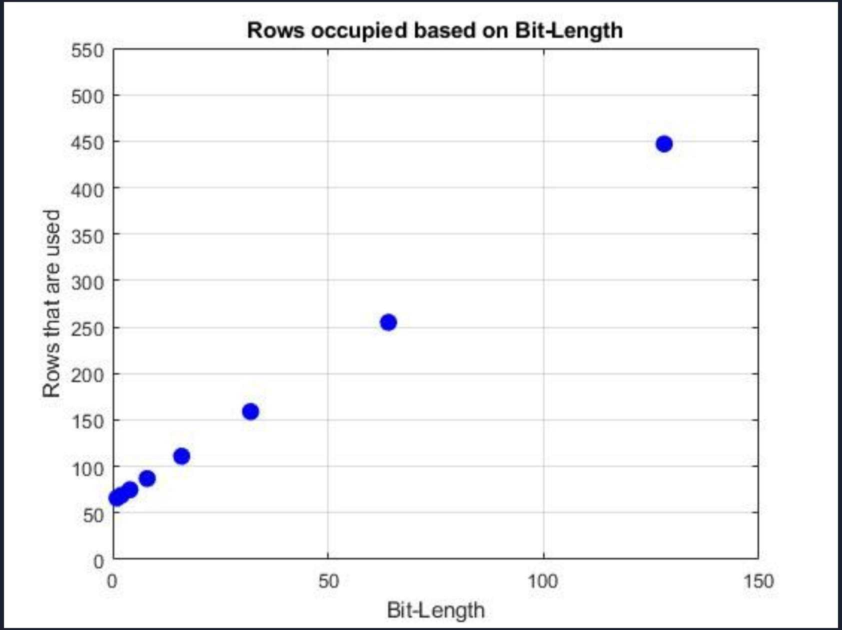


If we have more than 1 bit in each operand, we add the Cout of the previous stage in the same way we add Cin

Experiments & Results

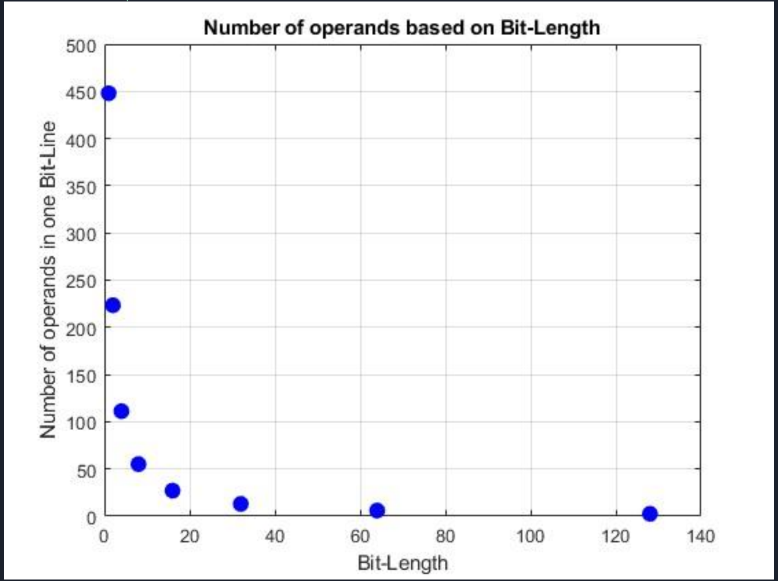
Next: Addition of more than 1 bit-length operands

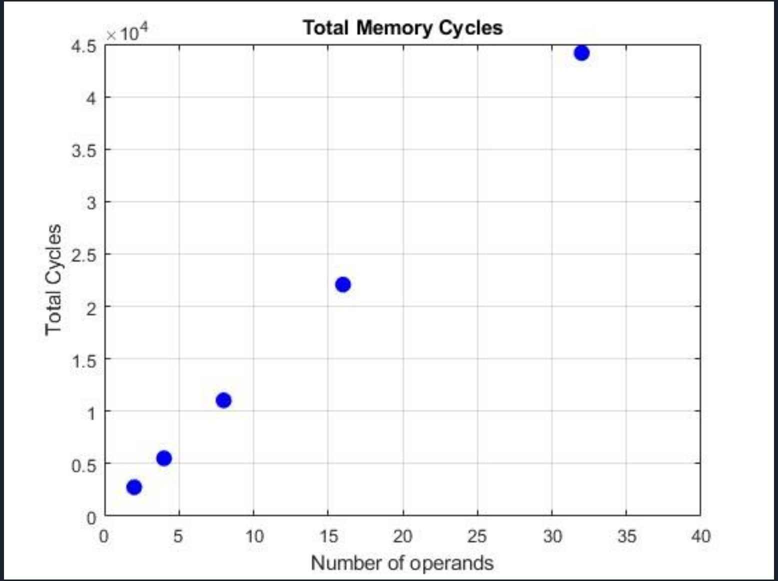




Experiments & Results

Next: Sum of more than two operands





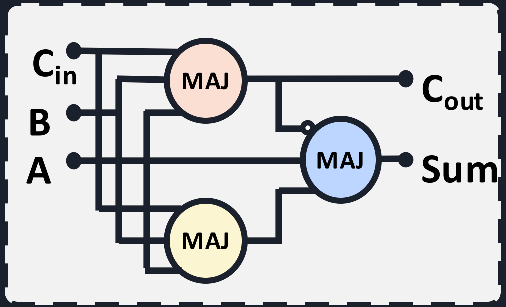
We need 45000 cycles to add 32 8-bit operands!

Do we exploit all the 65000 bit-lines?

Comments

The presented full-adder was designed in a general way using AND/OR logical gates based on Boolean algebra.

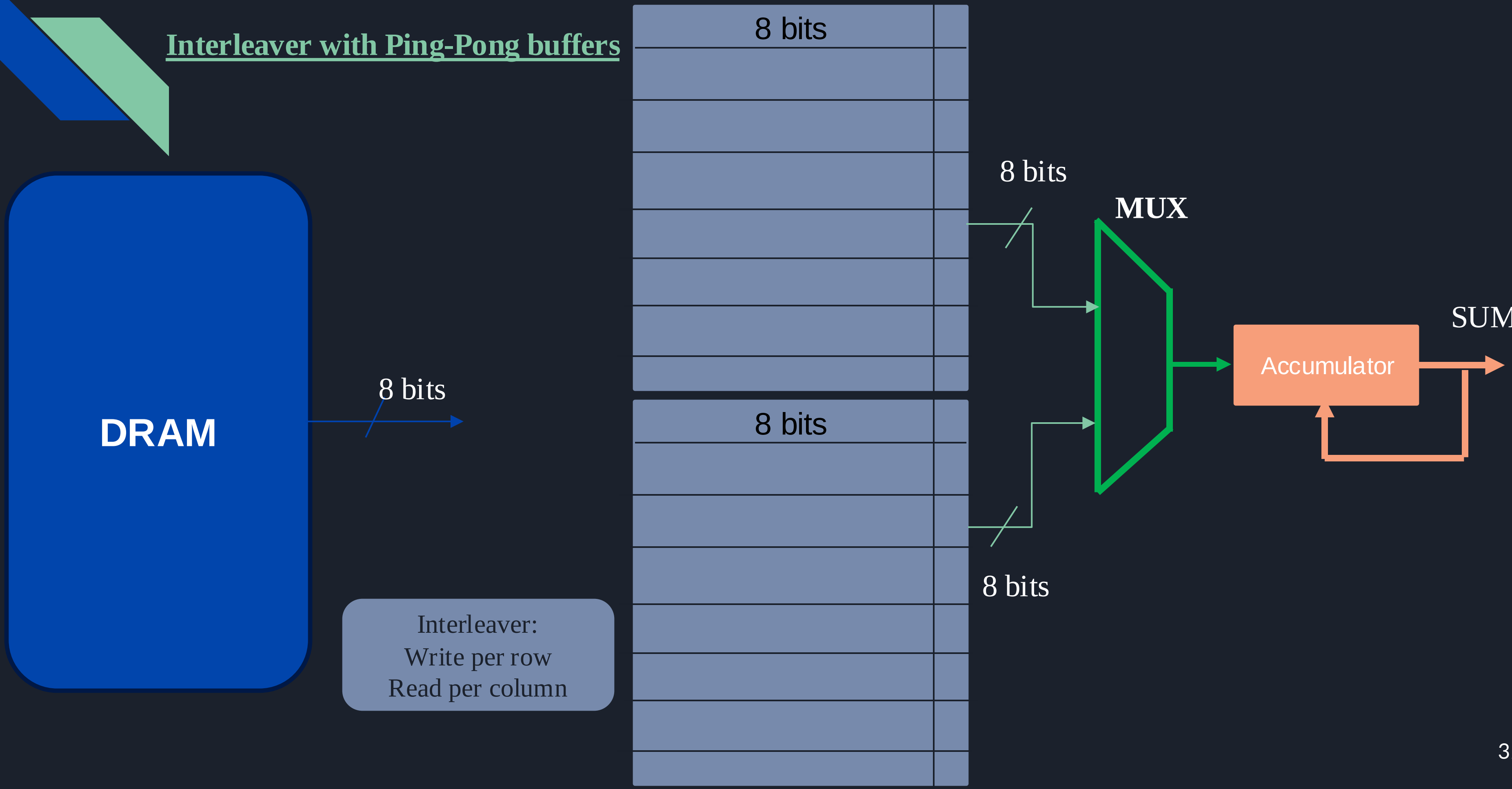
We can achieve better results by using Majority logic which must be specific for every function which is ~ 4x times faster



DRAM-FPGA Co-Processing

Interleaver with Ping-Pong buffers

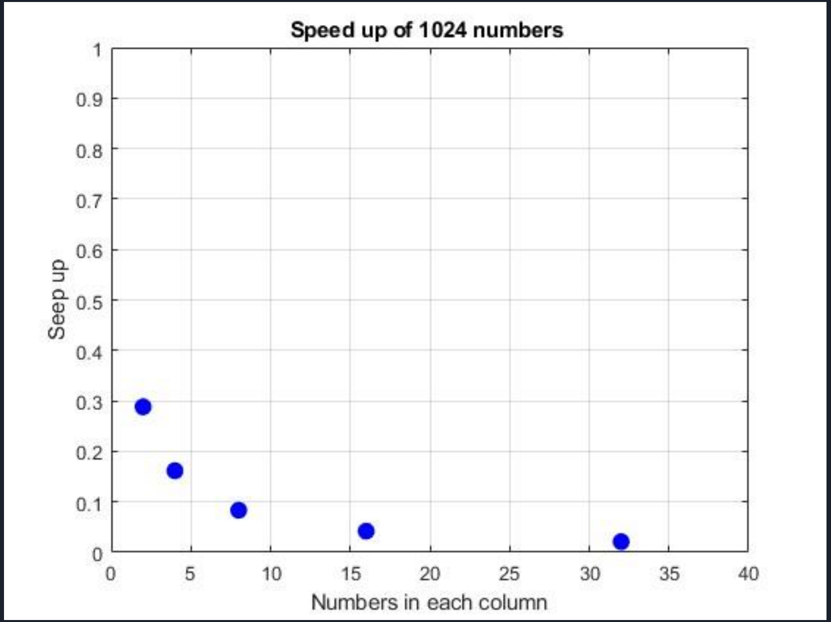
Ping Pong Pipelined Buffers / Interleaver

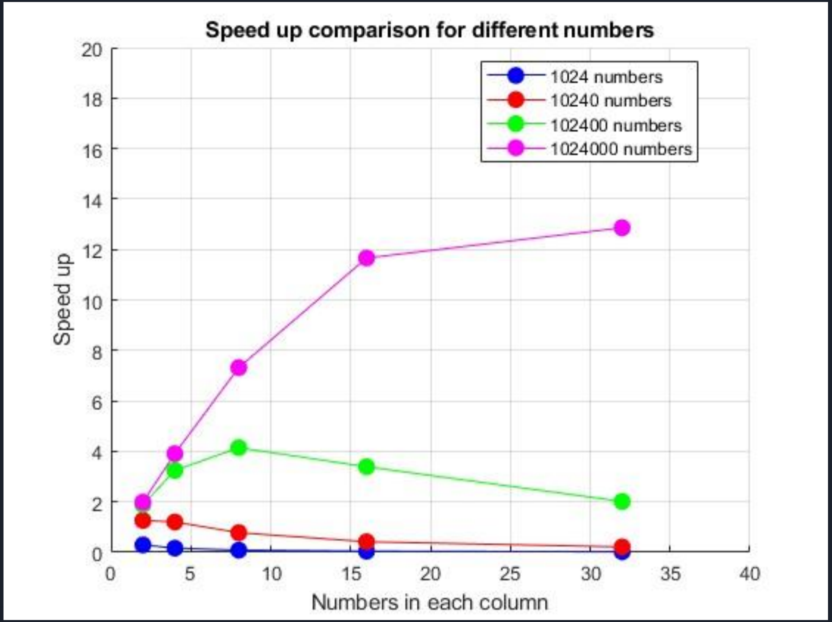


Interleaver:
Write per row
Read per column

DRAM-FPGA Co-Design

Accelaration of the sum-operation

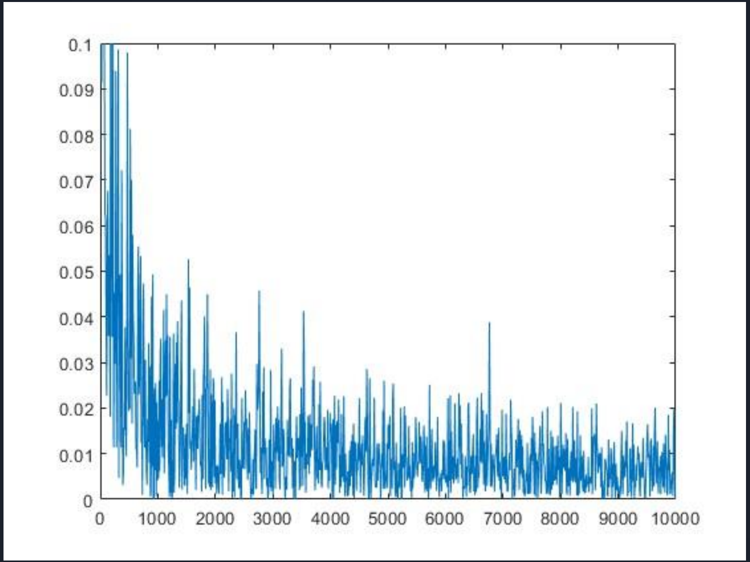




Conclusions & Future Work

1. For small number of operands, we have no acceleration

2. For larger number of operands, we have speed-up but with errors



Conclusions & Future Work

1. Explore the multiplication operator

2. Estimation of the Power Efficiency

3. DSP applications (thousands of Convolution Kernel multiplications via thousands of columns operating in parallel, such as Sobel, approximate Gauss)

4. ML applications where there is bit error tolerance