AN ASYNCHRONOUS DIVIDER IMPLEMENTATION

Navaneeth Jamadagni

and

Jo Ebergen

Acknowledgements

- Oracle Labs
- DARPA
- Asynchronous Research Center
- Reviewers

Take Away

- Division Algorithm
  - Usually retires 1 quotient digit per iteration
  - Sometimes retires 2 quotient digits per iteration
- An Asynchronous Design
  - Exploits the time disparity between addition and shift operations
- Result
  - Improvement in speed

Outline

- Introduction
- Divine Division
- Hardware Design
- Results
- Conclusion

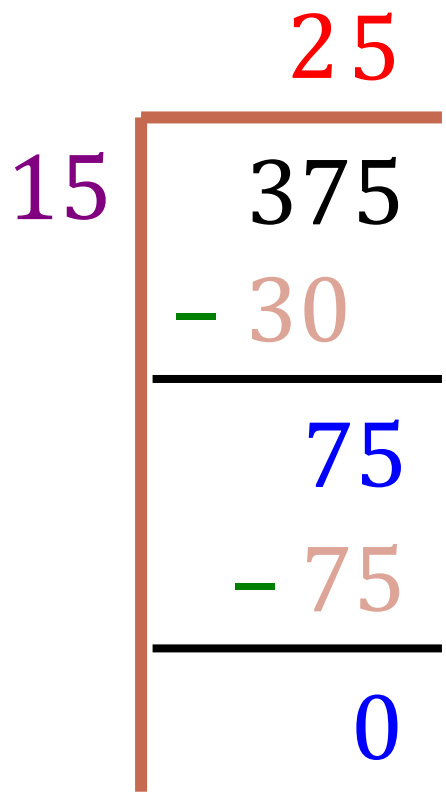
Outline

- Introduction
- Divine Division
- Hardware Design
- Results
- Conclusion

Division

$$Numerator, N = ((Quotient, Q) * (Denominator, D)) + Remainder, R$$

Long Division, Example



Long Division

- Given a Numerator and a Denominator

Iteration

1. Guess the quotient digit
   a. Ten choices, from the set {0 … 9}
2. Multiply the denominator by your guess
3. Subtract the product from the remainder to get a new *remainder*
4. Retire that quotient digit

5. Repeat steps 1 to 4 until the *remainder* is 0 or you run out of time

Two Division Methods

- Multiplicative Methods
  - Many choices (e.g., {0 …. 2^8})
  - Expensive multiplication
  - Retires many quotient bits per iteration
  - Few iterations
- Digit-Recurrence Methods
  - Very few choices (e.g., {0,1} or {-2, -1, 0, 1, 2})
  - Inexpensive multiplication
  - Retires one or two quotient bits per iteration
  - Many iterations

SRT Division

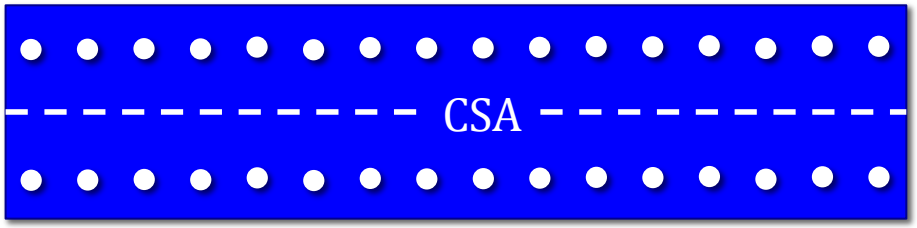
- Most common implementation in microprocessors
- Carry-Save Additions or Subtractions
  - Guess by Carry Propagate Addition
  - Guess by Table Lookup
- After Sweeney, Robertson and Tocher

SRT Division, Iteration

1. Guess : quotient digit from the set {–1, 0, 1}
2. Multiply : –1 × D, 0 × D, 1 × D

SRT Division, Iteration

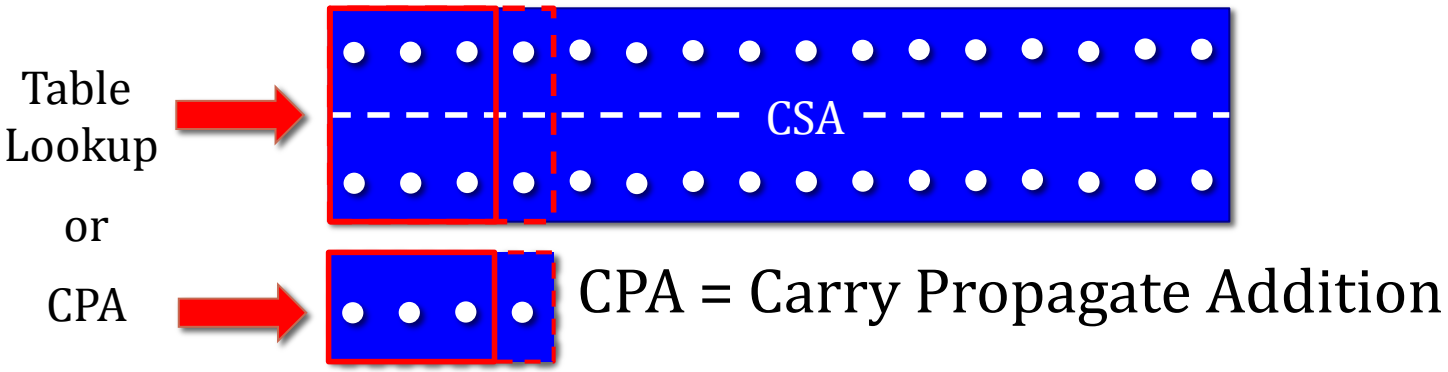
1. Guess : quotient digit from the set {–1, 0, 1}
2. Multiply : –1 × D, 0 × D, 1 × D
3. Subtract or Add:



CSA = Carry Save Addition

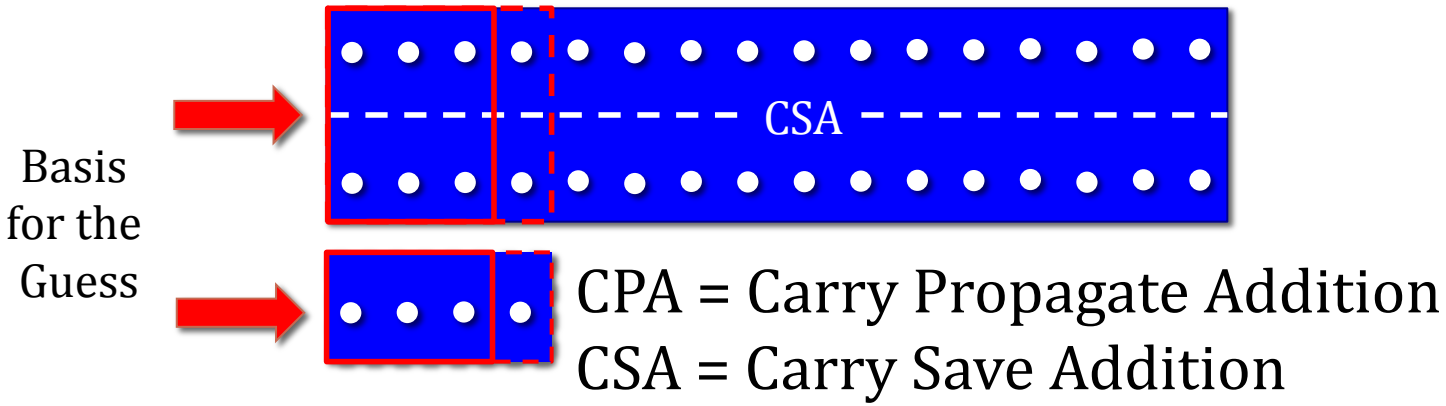
SRT Division, Iteration

1. Guess : quotient digit from the set {–1, 0, 1}
2. Multiply : –1 × D, 0 × D, 1 × D
3. Subtract or Add:



SRT Division, Iteration

1. Guess : quotient digit from the set {–1, 0, 1}
2. Multiply : –1 × D, 0 × D, 1 × D
3. Subtract or Add :



4. Retire : One quotient digit always

Outline

- Introduction
- Divine Division
- Hardware Design
- Results
- Conclusion

Divine Division

"Divine" = Discover by guess work or Intuition

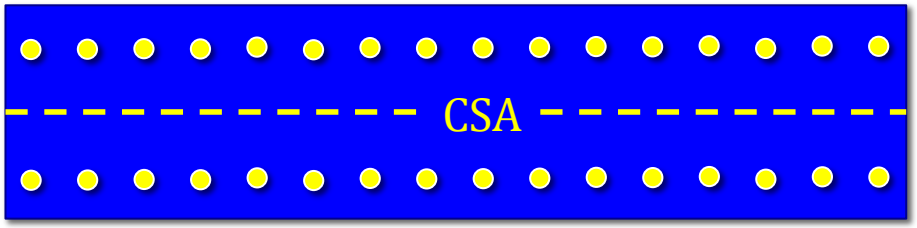
- Carry-Save Addition or Subtraction only
- Two versions in the paper: E and H
- Developed at Sun Labs, by
  Jo Ebergen, Ivan Sutherland
  and Danny Cohen

Divine Division, Iteration

1. Guess : quotient digit from the set {-2, -1, 0, 1, 2}
2. Multiply : $-2 \times D$, $-1 \times D$, $0 \times D$, $1 \times D$, $2 \times D$

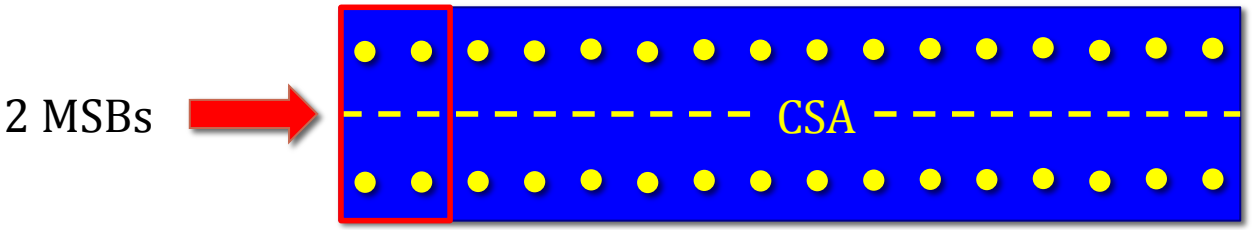
Divine Division, Iteration

1. Guess : quotient digit from the set {-2, -1, 0, 1, 2}
2. Multiply : –2 × D, -1 × D, 0 × D, 1 × D, 2 × D
3. Subtract or Add :



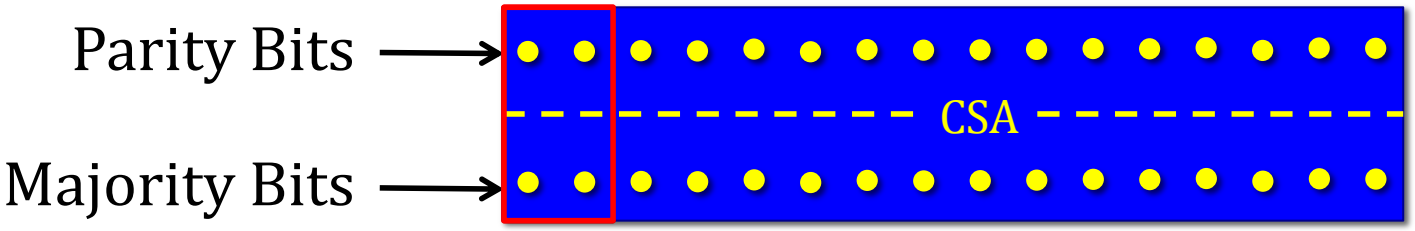
Divine Division, Iteration

1. Guess : quotient digit from the set {-2, -1, 0, 1, 2}
2. Multiply : $-2 \times D$, $-1 \times D$, $0 \times D$, $1 \times D$, $2 \times D$
3. Subtract or Add :



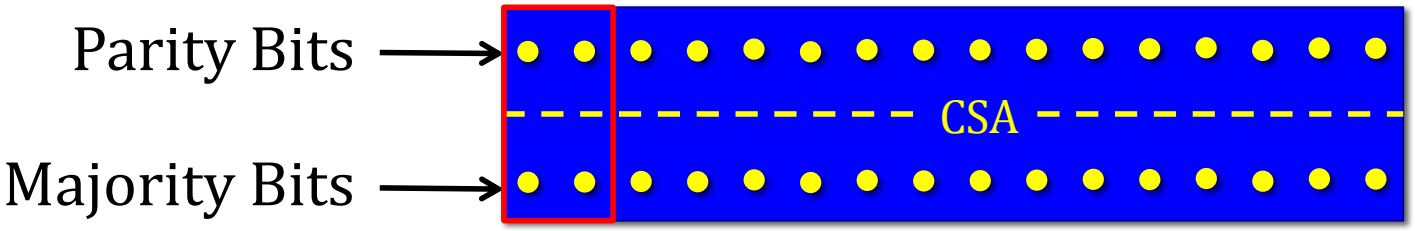
Divine Division, Iteration

1. Guess : quotient digit from the set {-2, -1, 0, 1, 2}
2. Multiply : –2 × D, -1 × D, 0 × D, 1 × D, 2 × D
3. Subtract or Add :



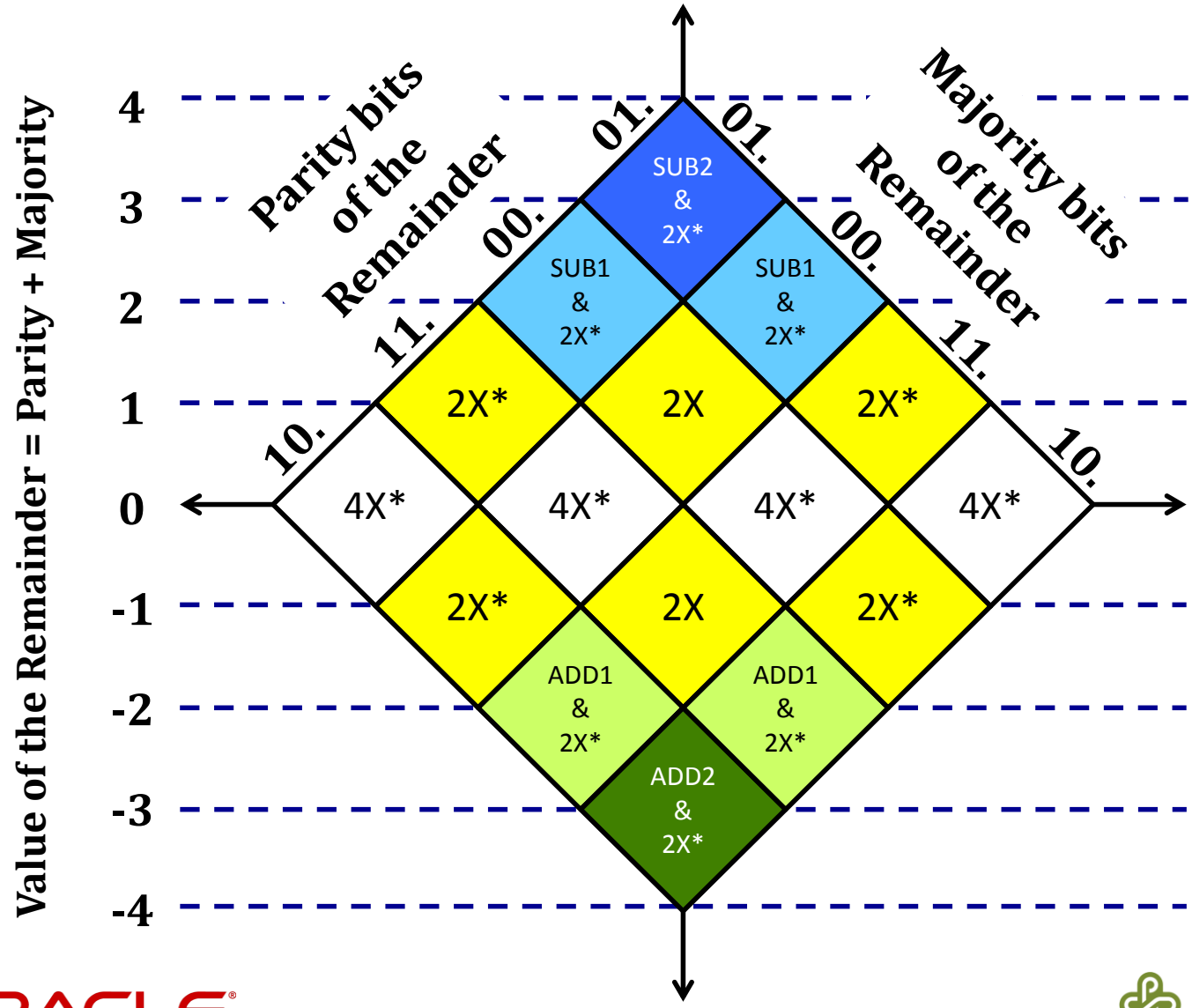
Divine Division, Iteration

1. Guess : quotient digit from the set {-2, -1, 0, 1, 2}
2. Multiply : $-2 \times D$, $-1 \times D$, $0 \times D$, $1 \times D$, $2 \times D$
3. Subtract or Add :

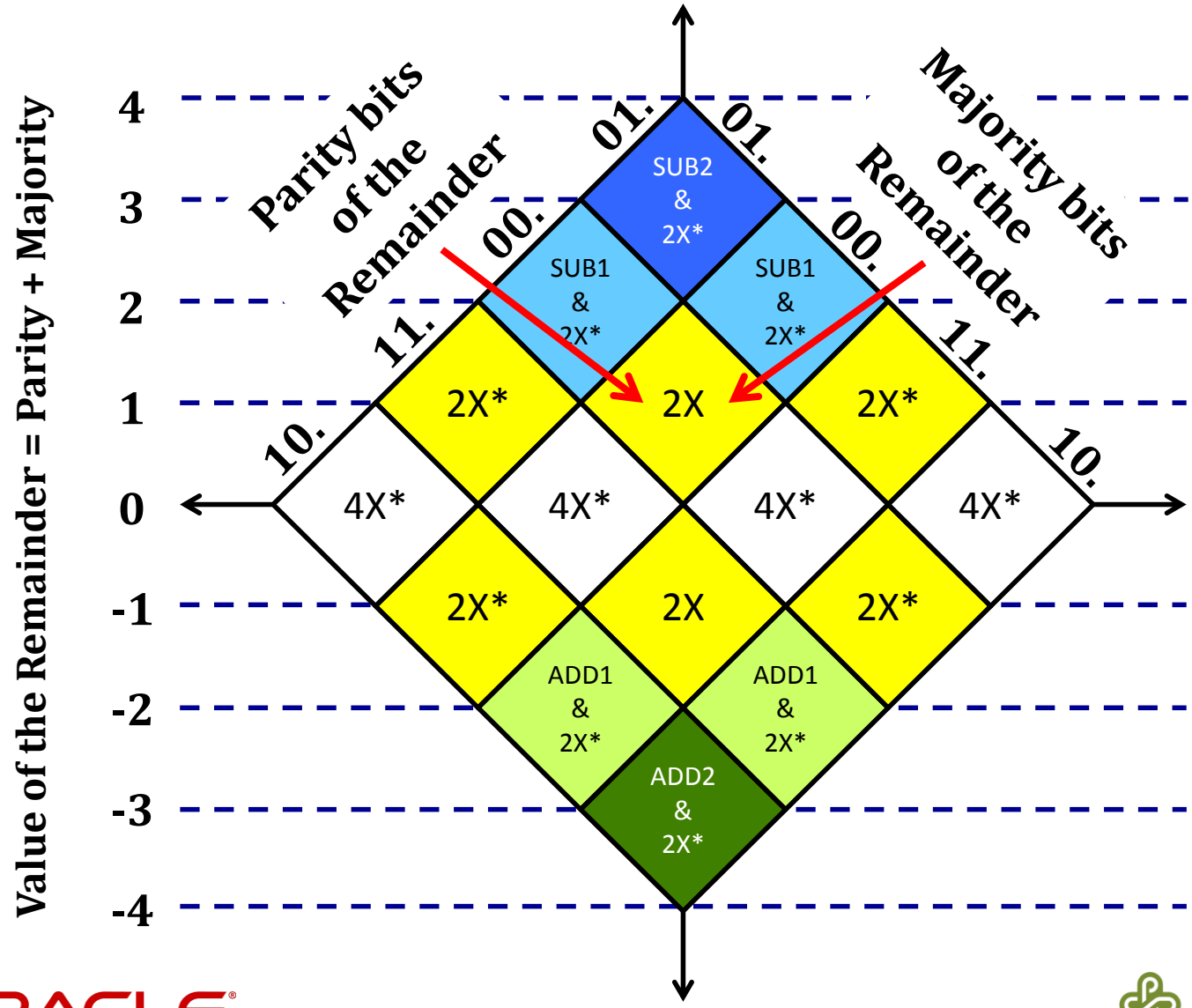


1. Retire : One quotient digit usually,
   Two quotient digits sometimes
2. One more iteration than SRT for equal accuracy

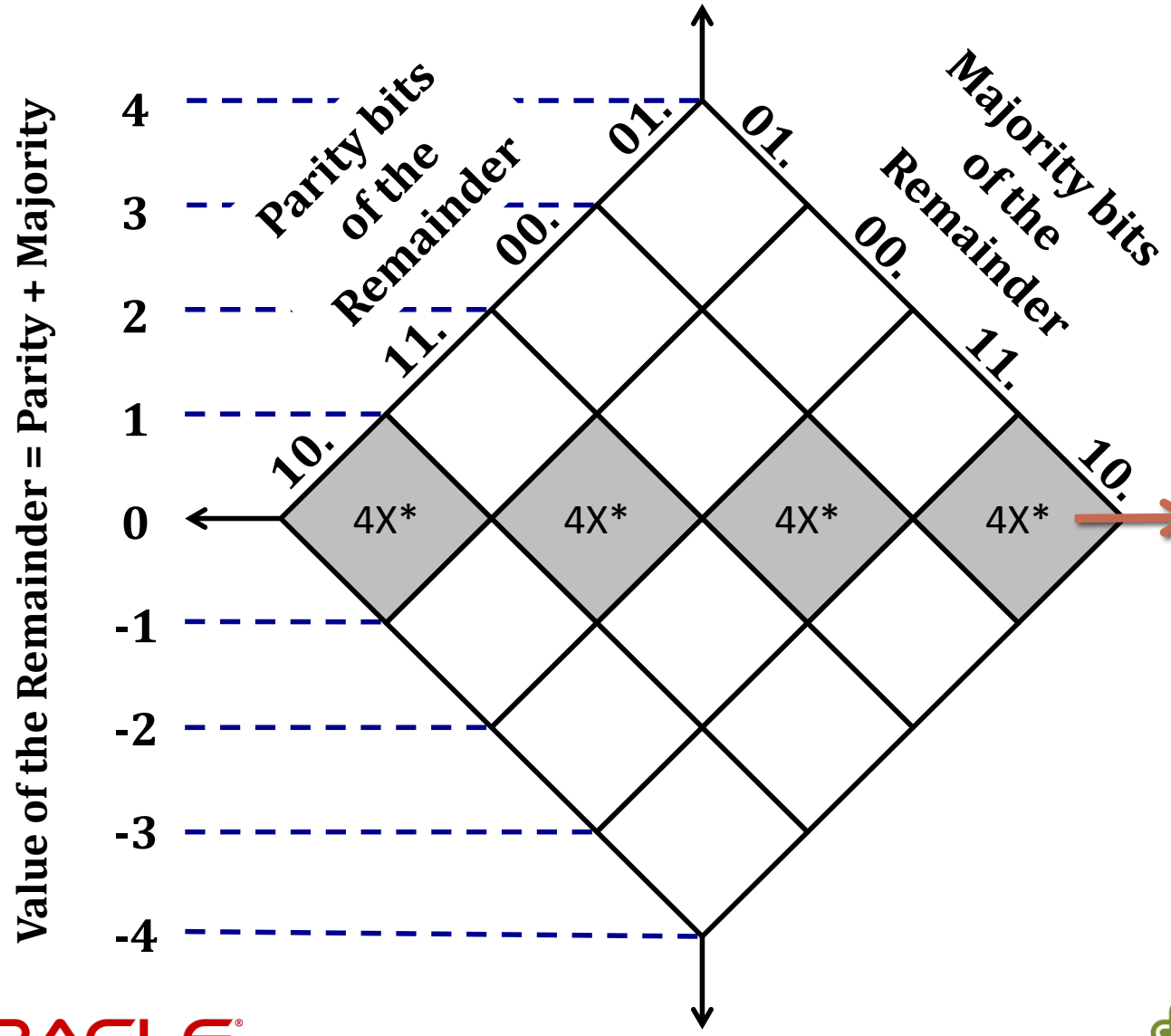
Divine Division Choices



Divine Division Choices



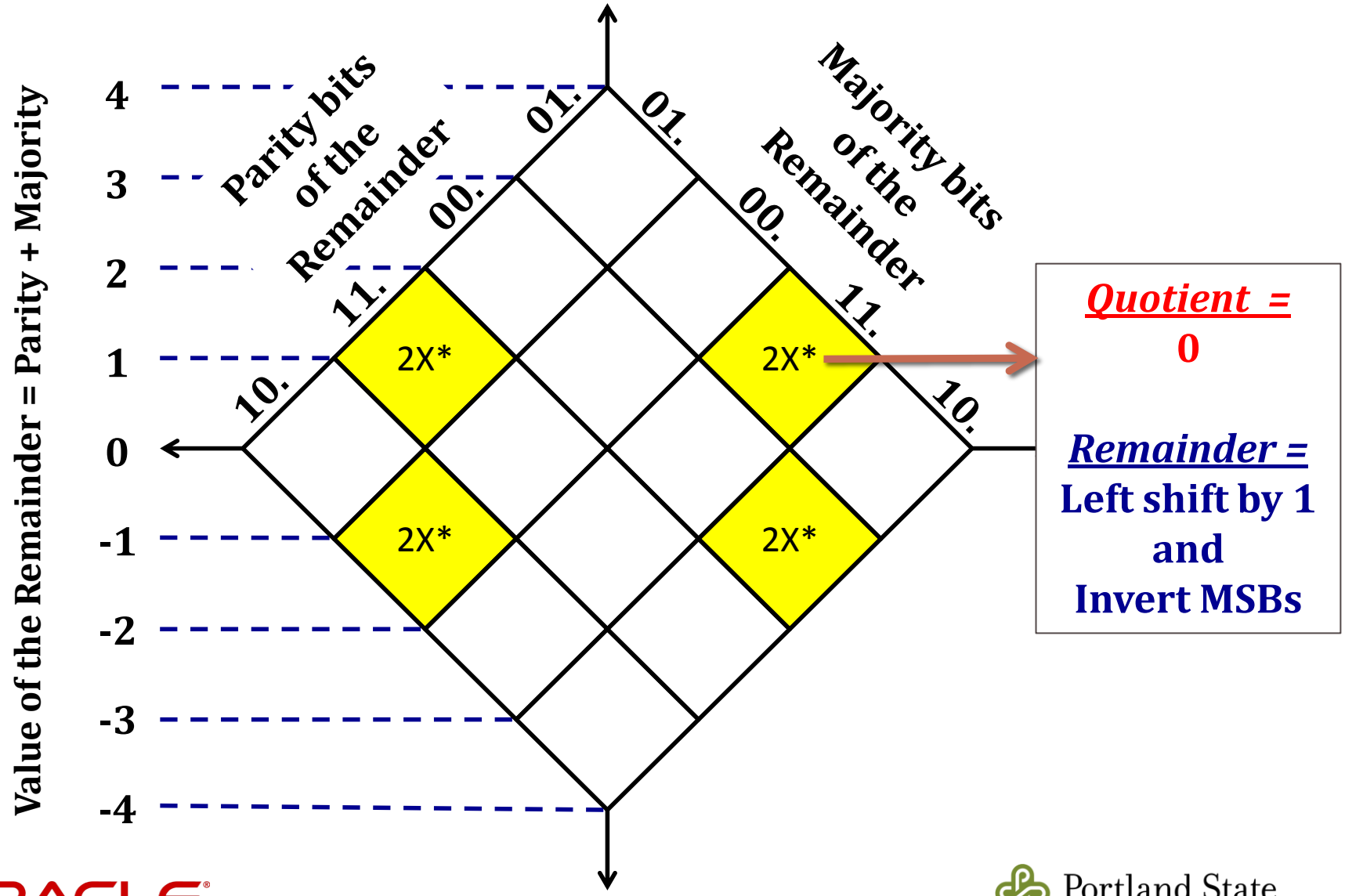
Divine Division Choice: 4X*



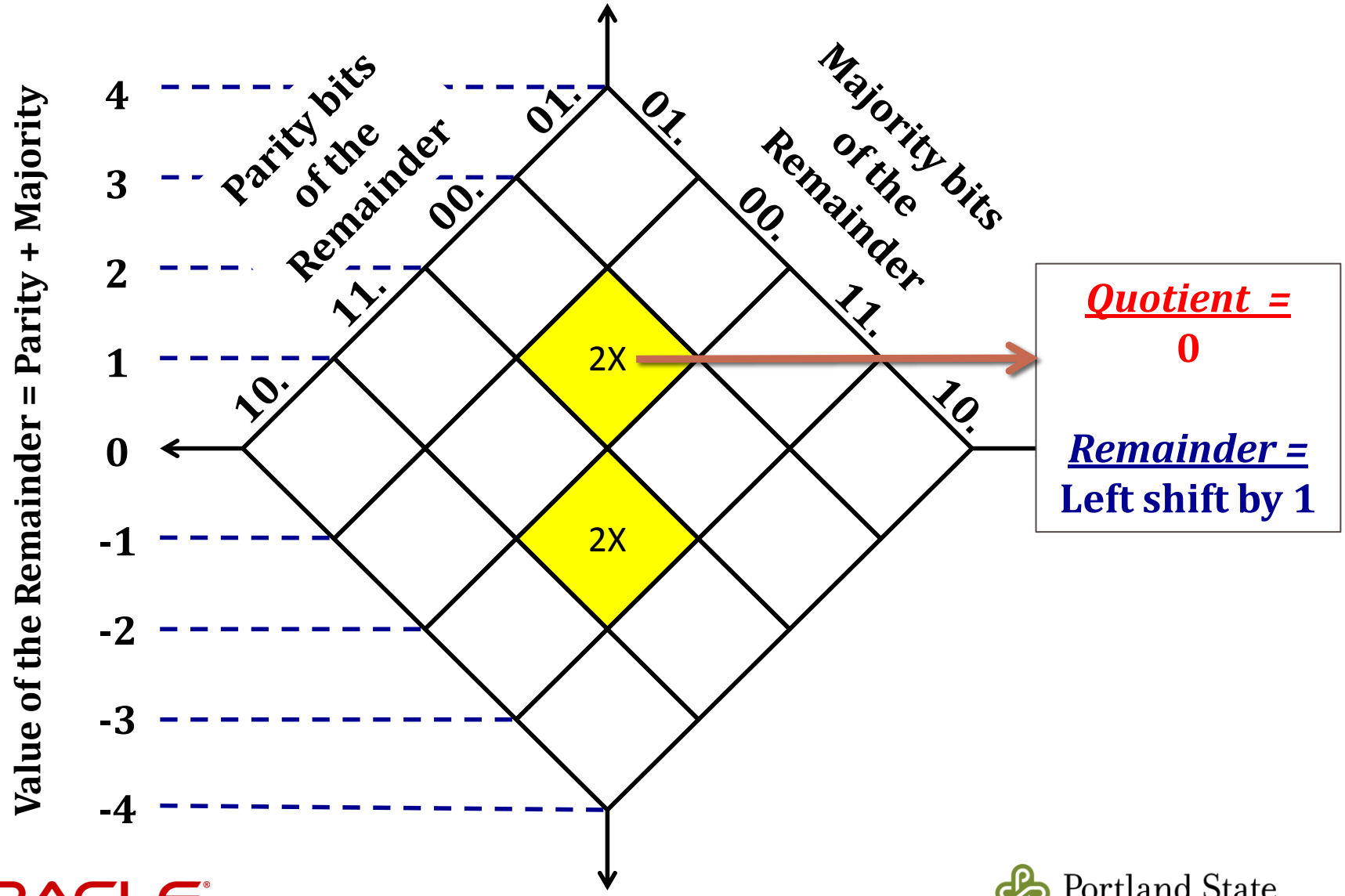
Quotient =
0 and 0

Remainder =
Left shift by 2 and Invert MSBs

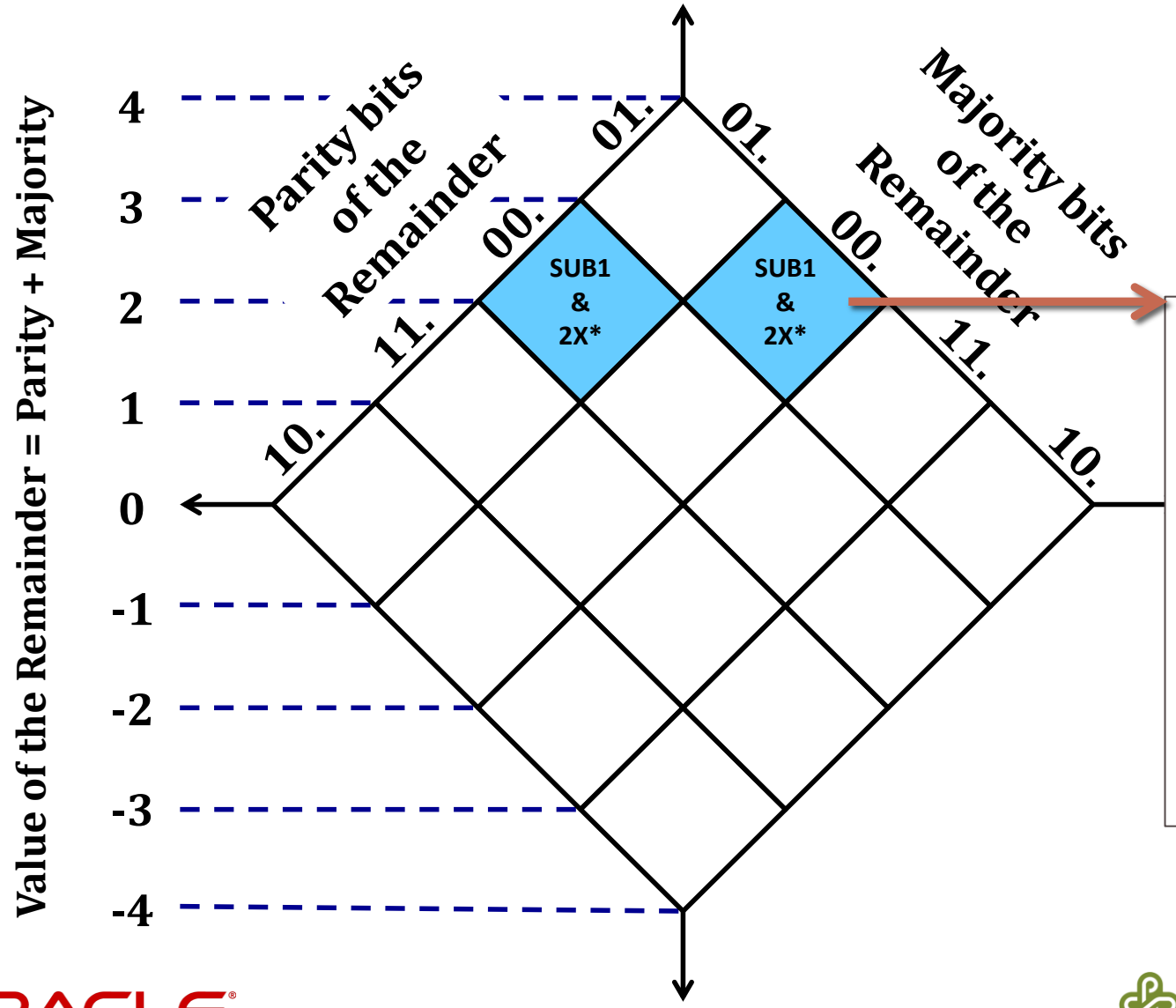
Divine Division Choice: 2X*



Divine Division Choice: 2X



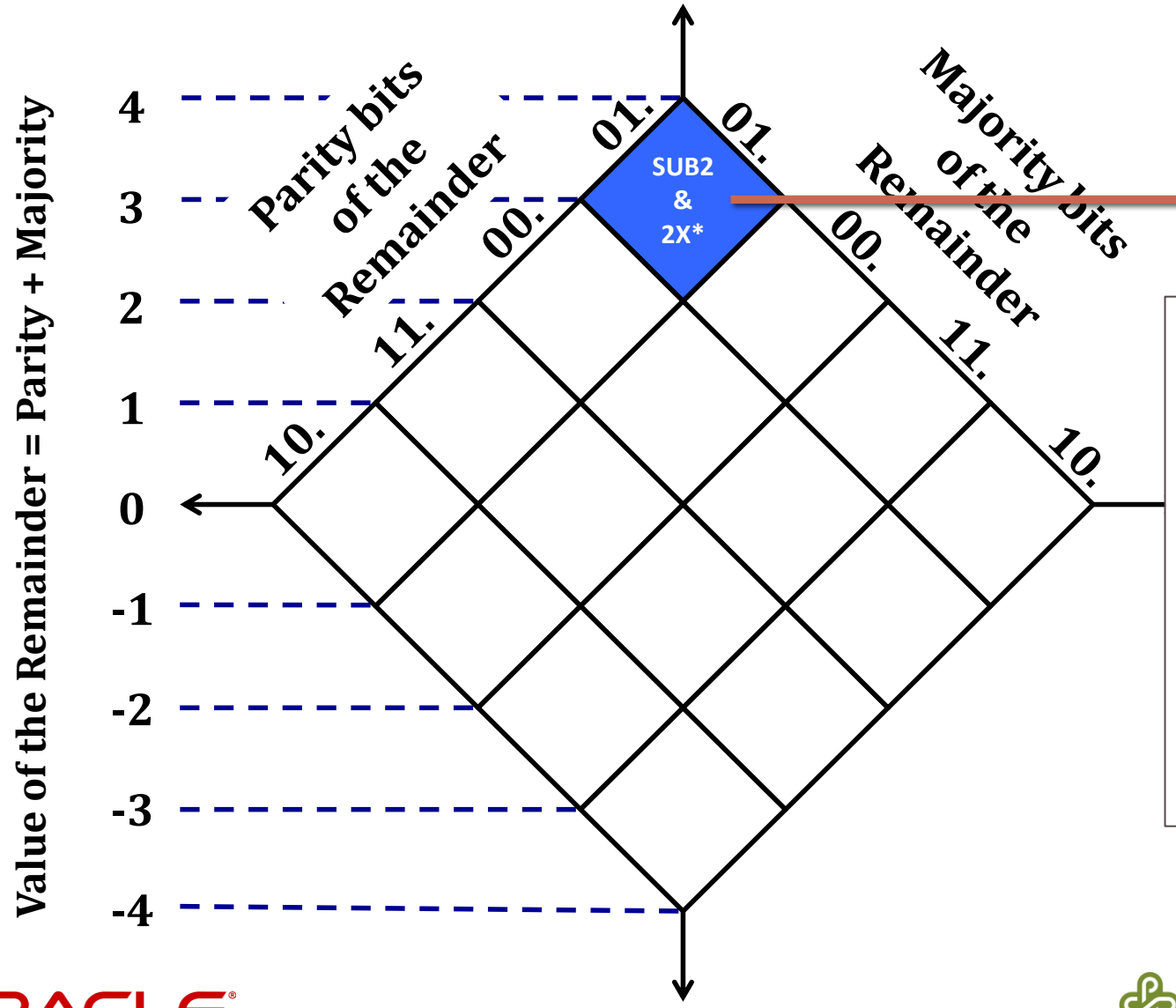
Divine Division Choice: SUB1 & 2X*



Quotient =
1

Remainder =
SUB 1 × D
&
Left shift by 1
and
Invert MSBs

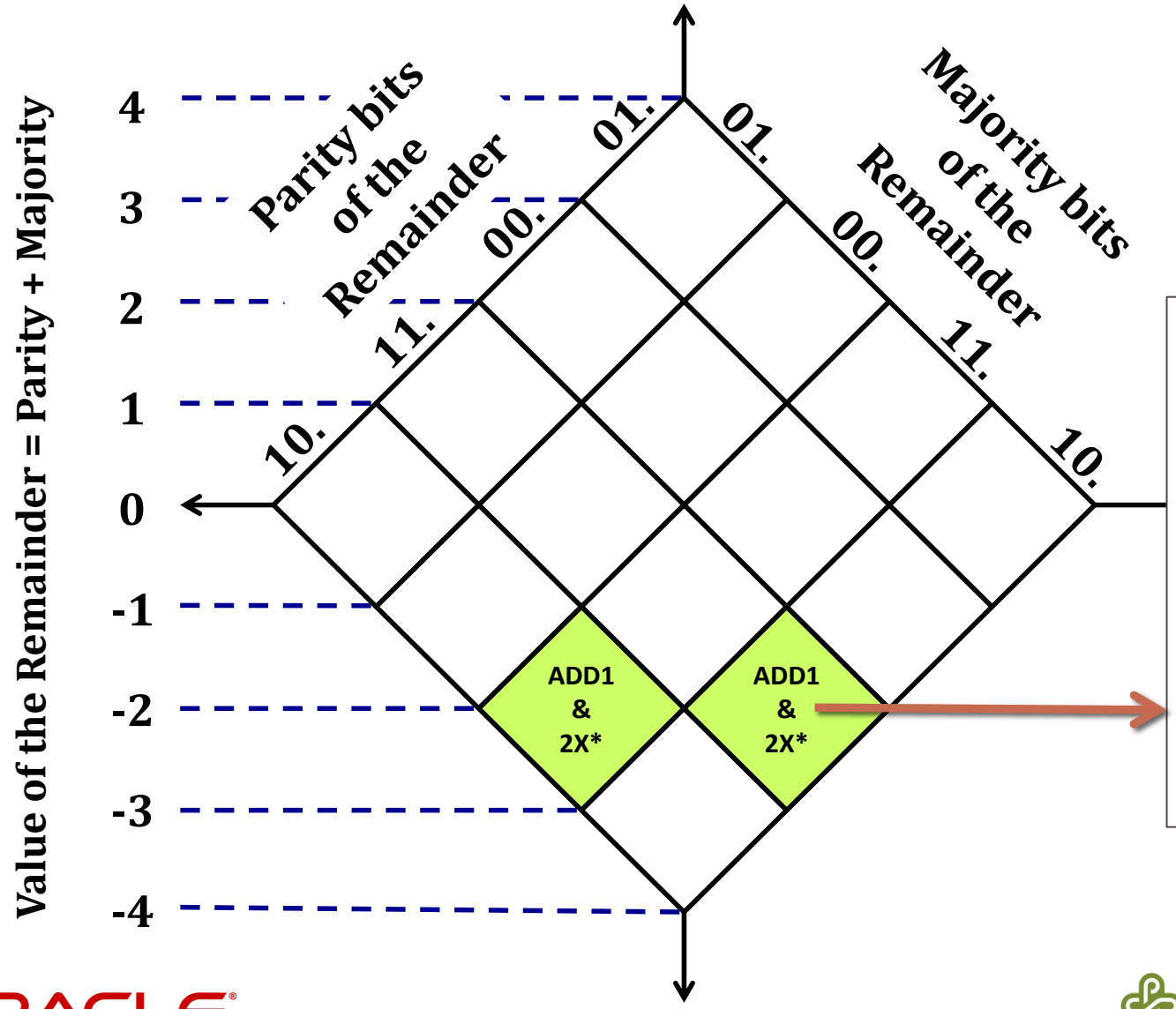
Divine Division Choice: SUB2 & 2X*



Quotient =
2

Remainder =
SUB 2 × D
&
Left shift by 1
and
Invert MSBs

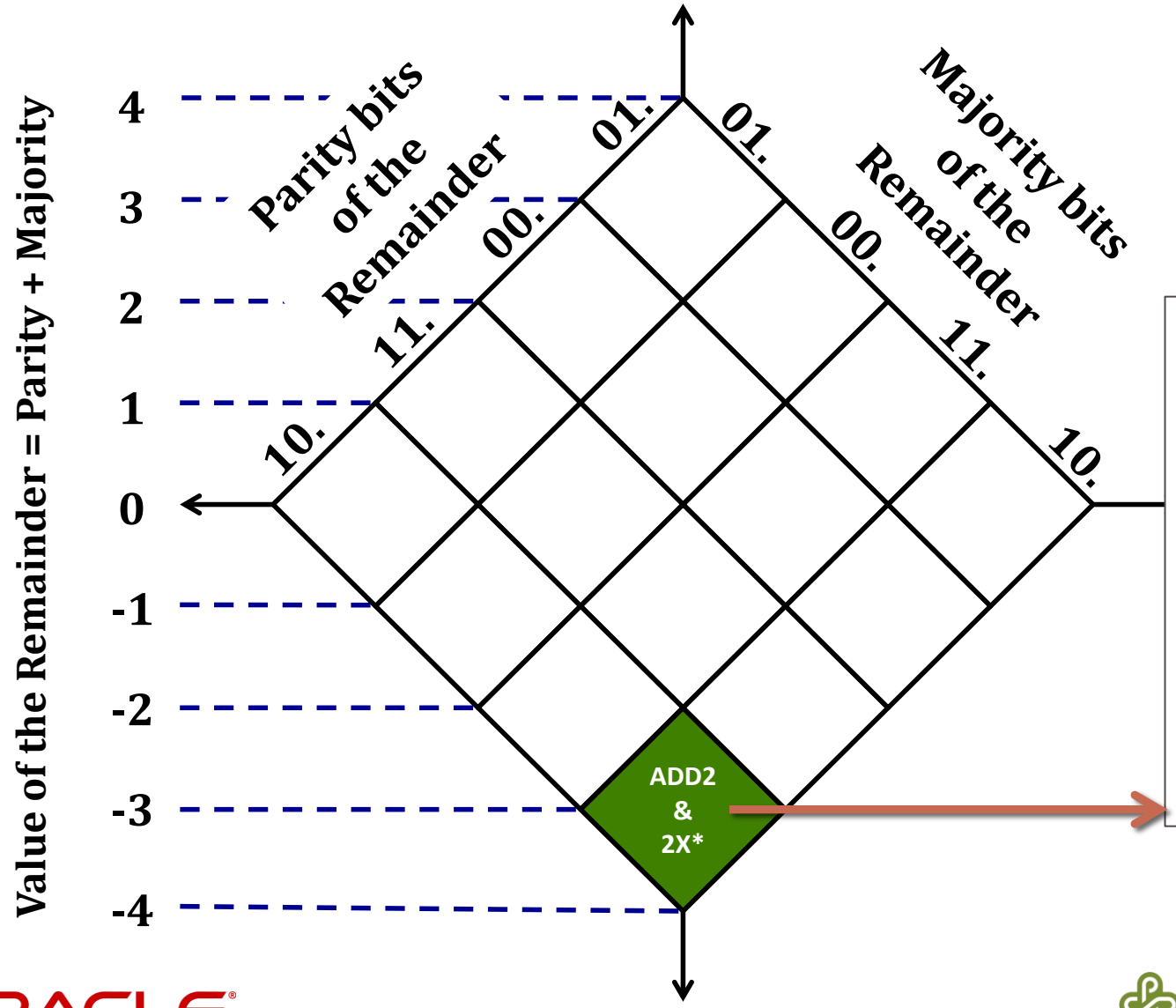
Divine Division Choice: ADD1 & 2X*



Quotient =
-1

Remainder =
ADD 1 × D
&
Left shift by 1
and
Invert MSBs

Divine Division Choice: ADD2 & 2X*

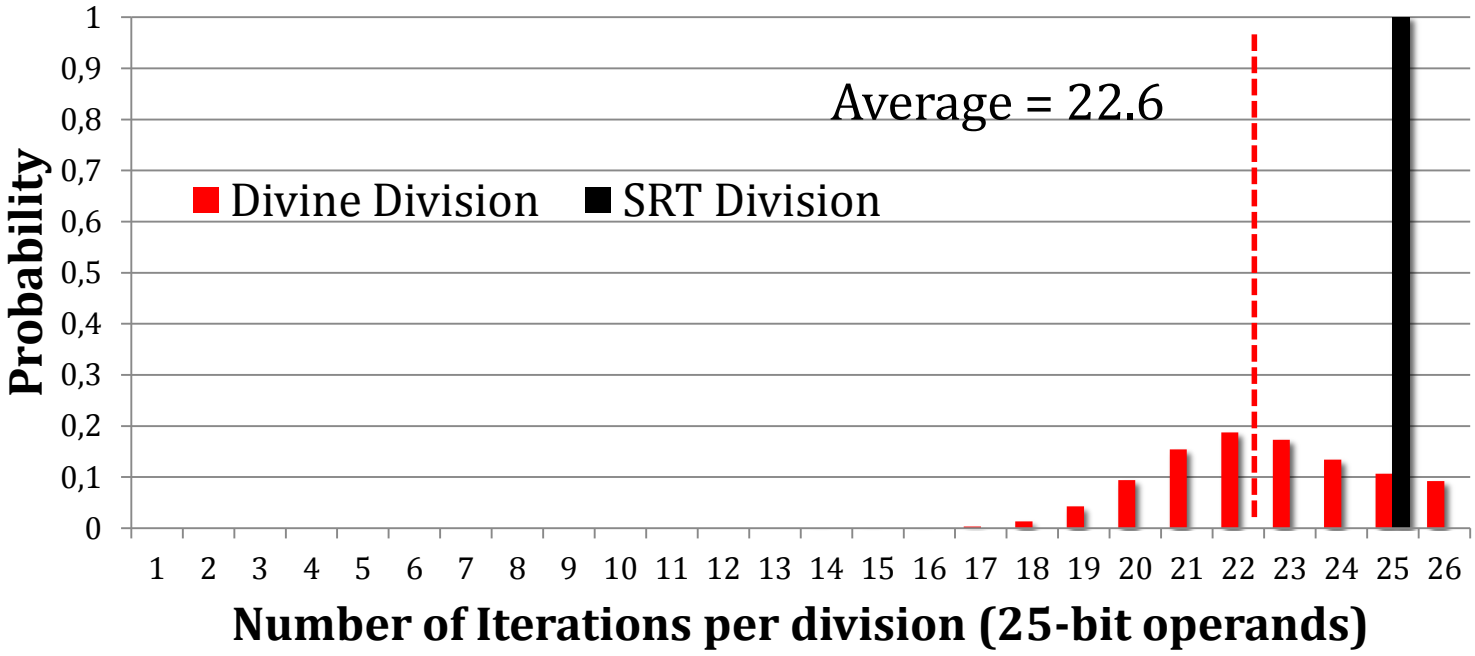


Quotient =
-2

Remainder =
ADD 2 × D
&
Left shift by 1
and
Invert MSBs

Number of Iterations per Division

- One million pairs of uniform-random 25-bit input operands



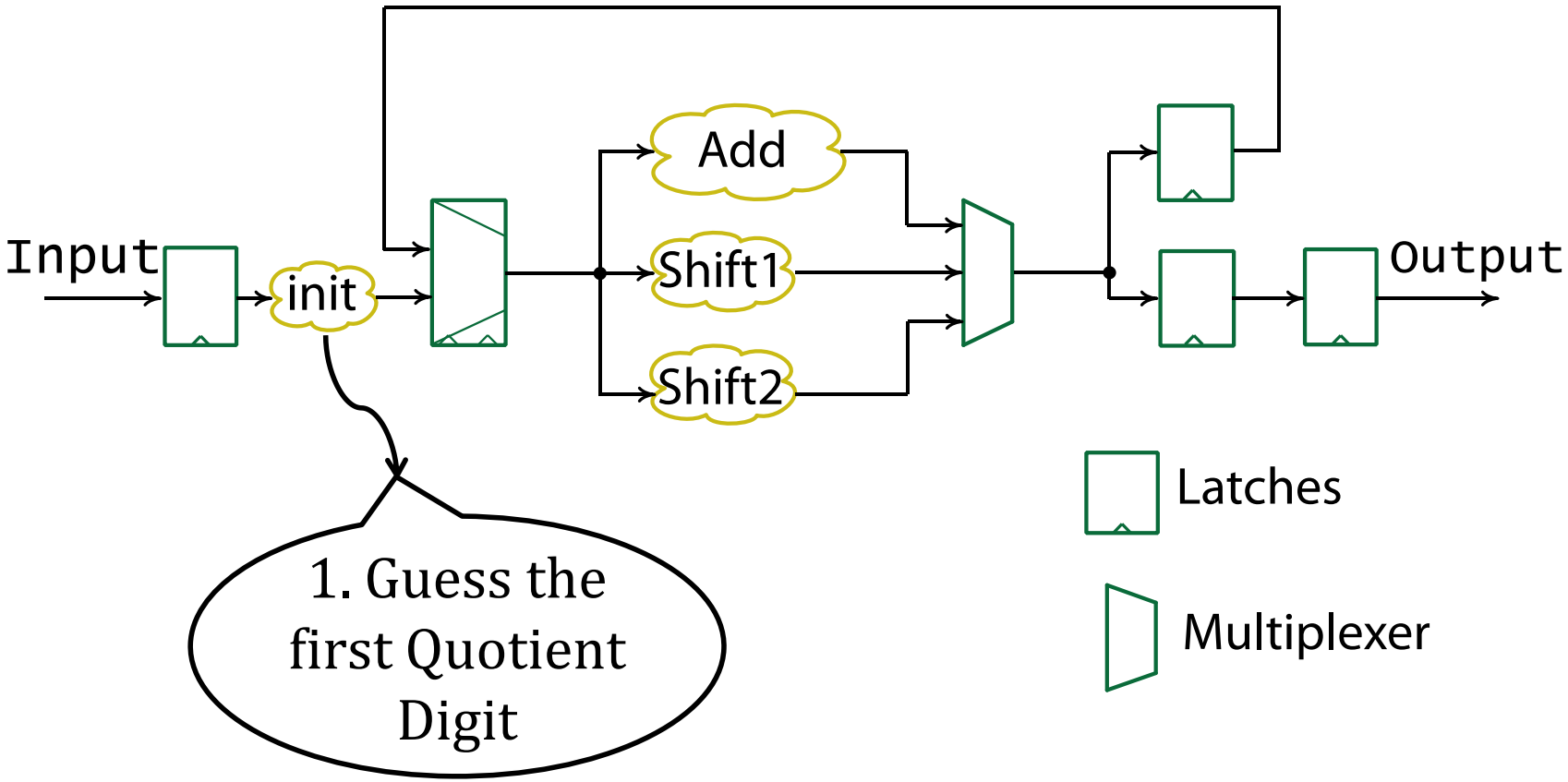
Outline

- Introduction
- Divine Division
- Hardware Design
- Results
- Conclusion

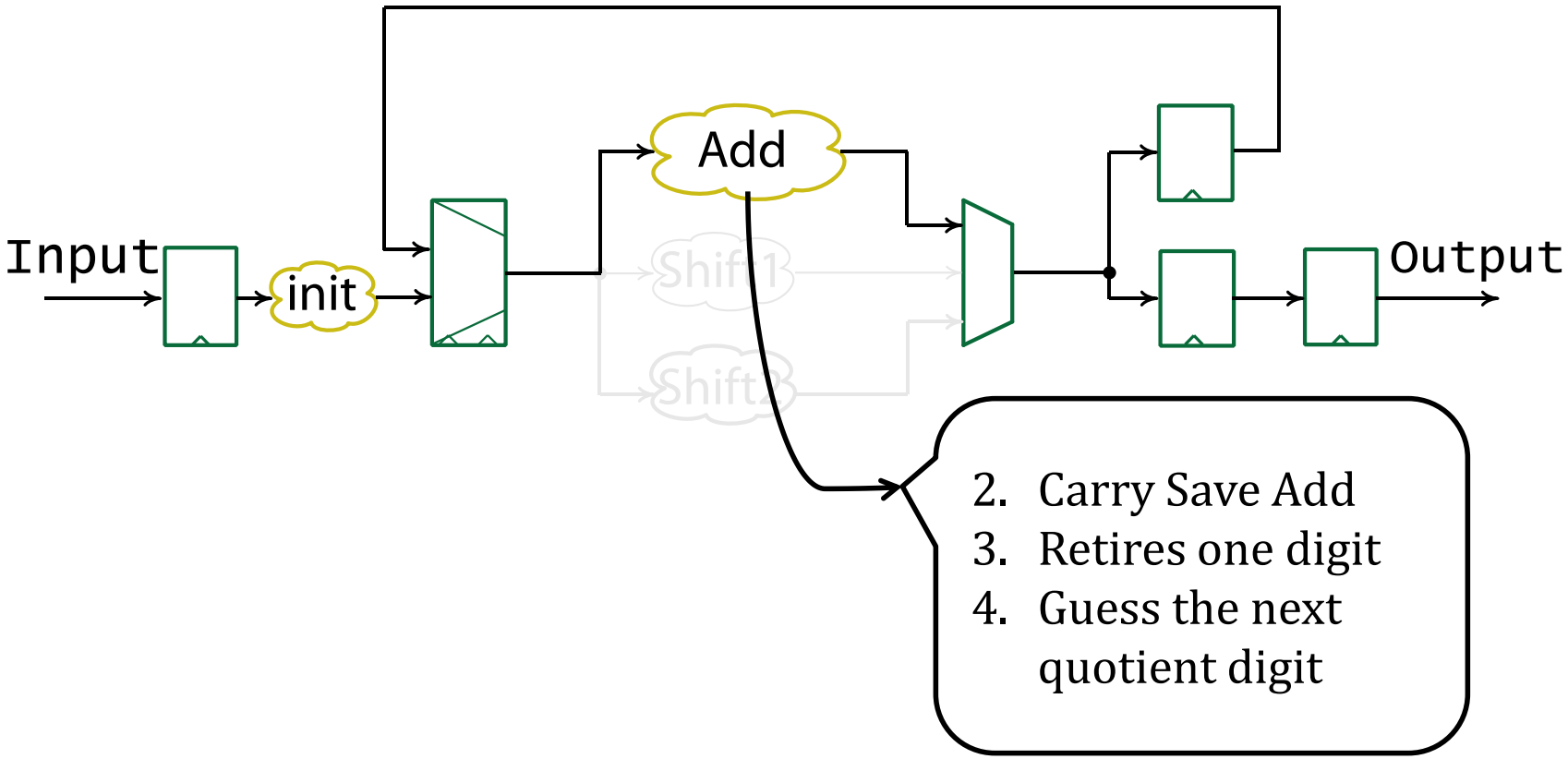
Asynchronous Divine Divider

- Control Path
  - Uses GasP modules
  - Generates the control signals for the registers
  - Delay matched to data path
    - Shift steps are faster than addition steps
  - Asynchronous loop counter
- Data Path
  - Registers and computational blocks (e.g., CSA)
  - Single rail bundled data

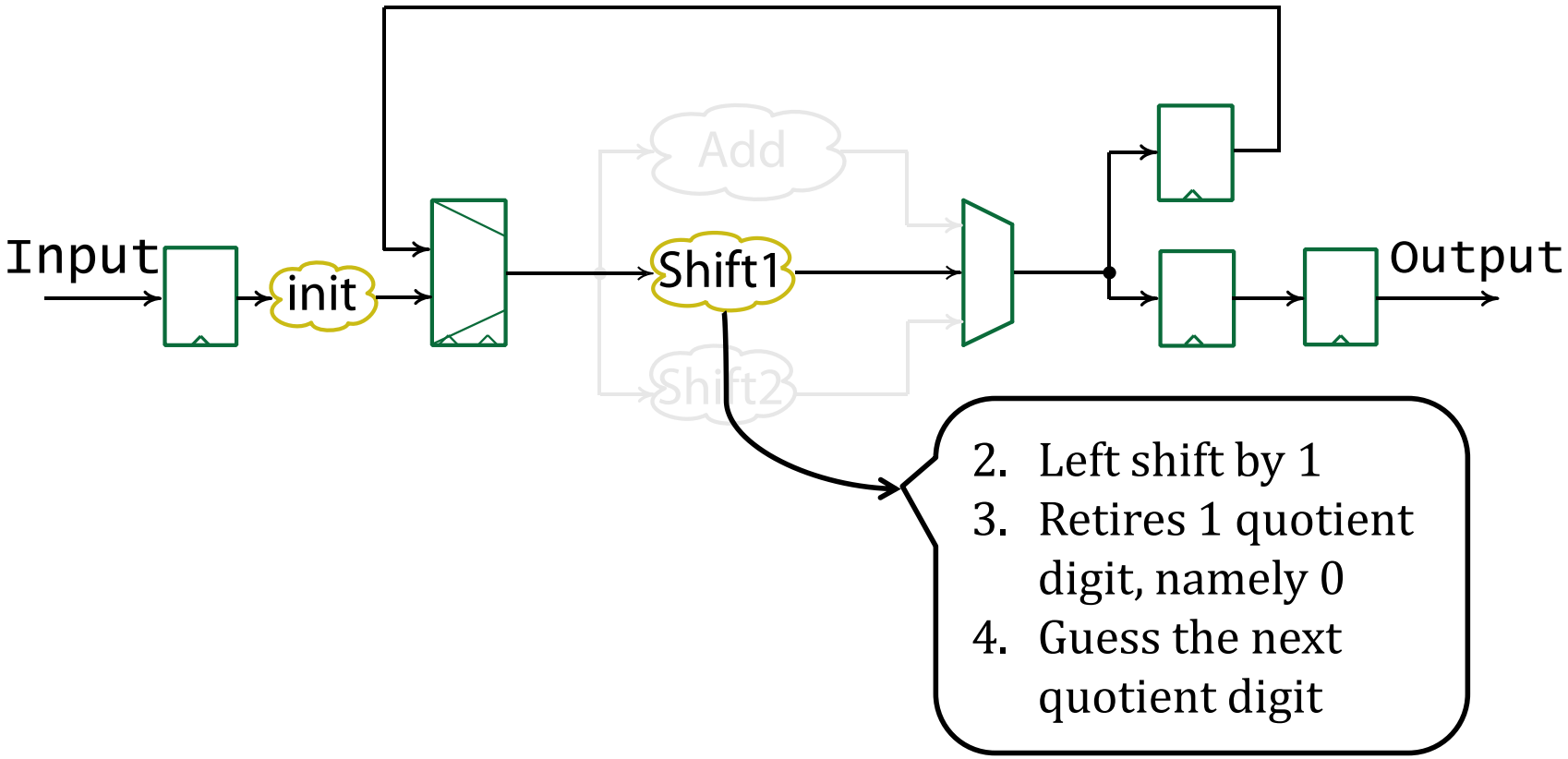
Data Path


Input
init
Add
Shift1
Shift2
Output
Latches
Multiplexer
1. Guess the first Quotient Digit

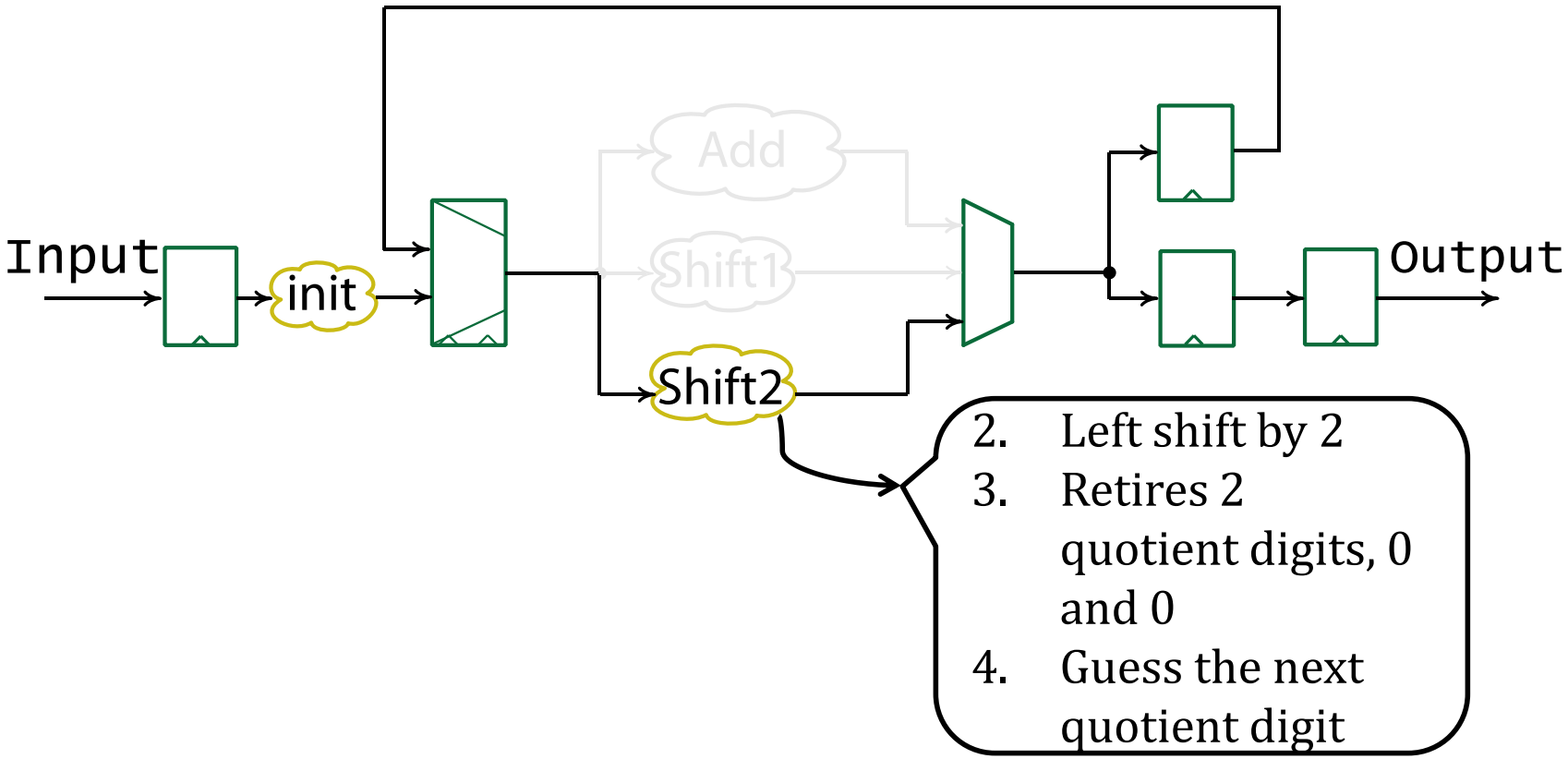
Data Path


Input
init
Add
Shift 1
Shift 2
Output
2. Carry Save Add
3. Retires one digit
4. Guess the next quotient digit

Data Path


Input
init
Add
Shift1
Shift2
Output
2. Left shift by 1
3. Retires 1 quotient digit, namely 0
4. Guess the next quotient digit

Data Path


Input
init
Add
Shift1
Shift2
Output
2. Left shift by 2
3. Retires 2 quotient digits, 0 and 0
4. Guess the next quotient digit

Outline

- Introduction
- Divine Division
- Hardware Design
- Results
- Conclusion

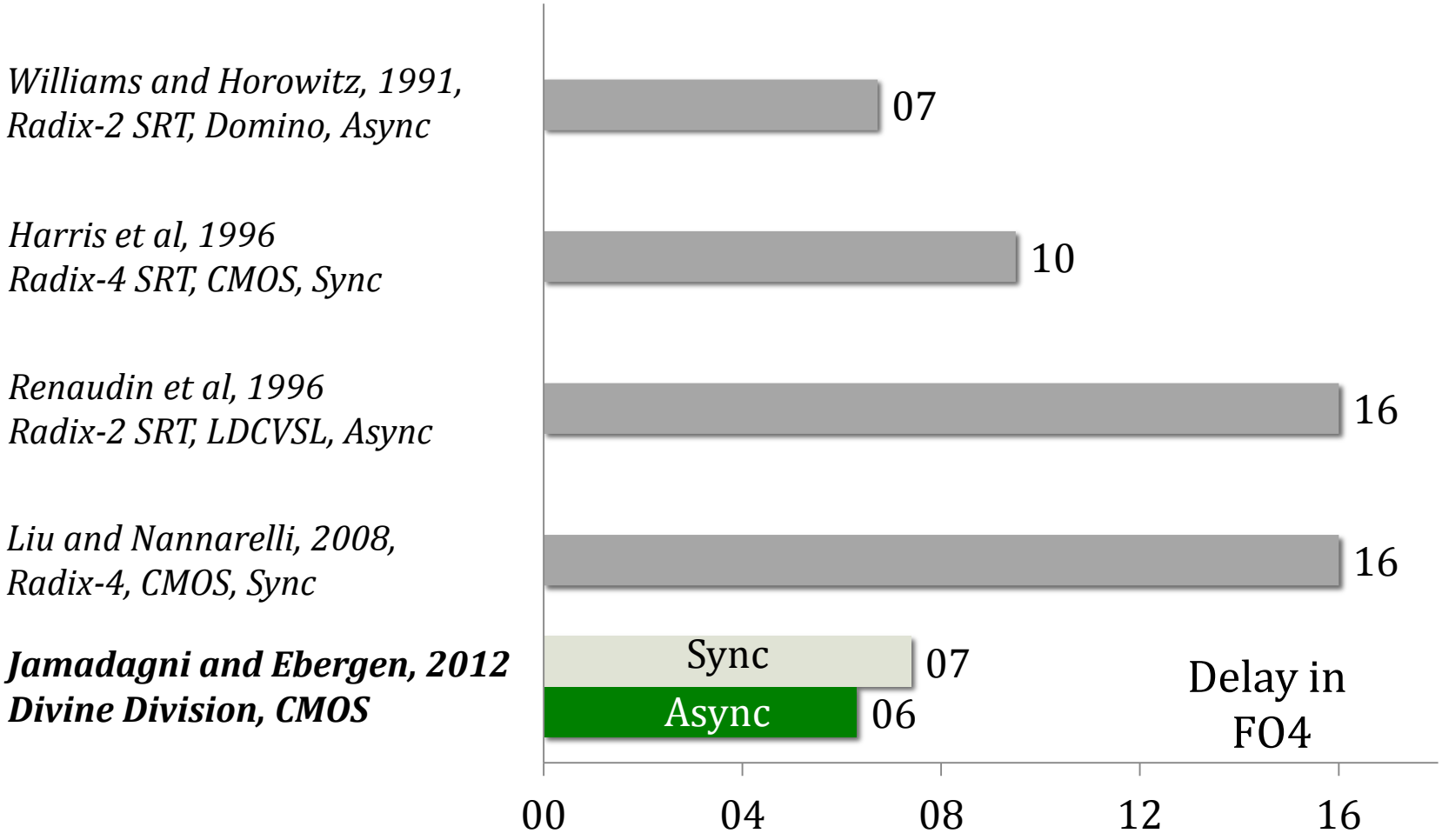
Simulation

- SPICE Simulation in TSMC 90nm
  - Partial layout with estimated wire lengths
- 50 pairs of random test vectors
  - Iteration statistics similar to 10^6 random test vectors
- Delay measurements are normalized to FO4 delays
  - Comparisons 1FO4 ≈ 25ps in 90nm process
  - Shift Path = 6 FO4, Add Path = 8.5 FO4
- Energy data estimated for Data and Control Paths
  - Comparison normalized to 1V Vdd

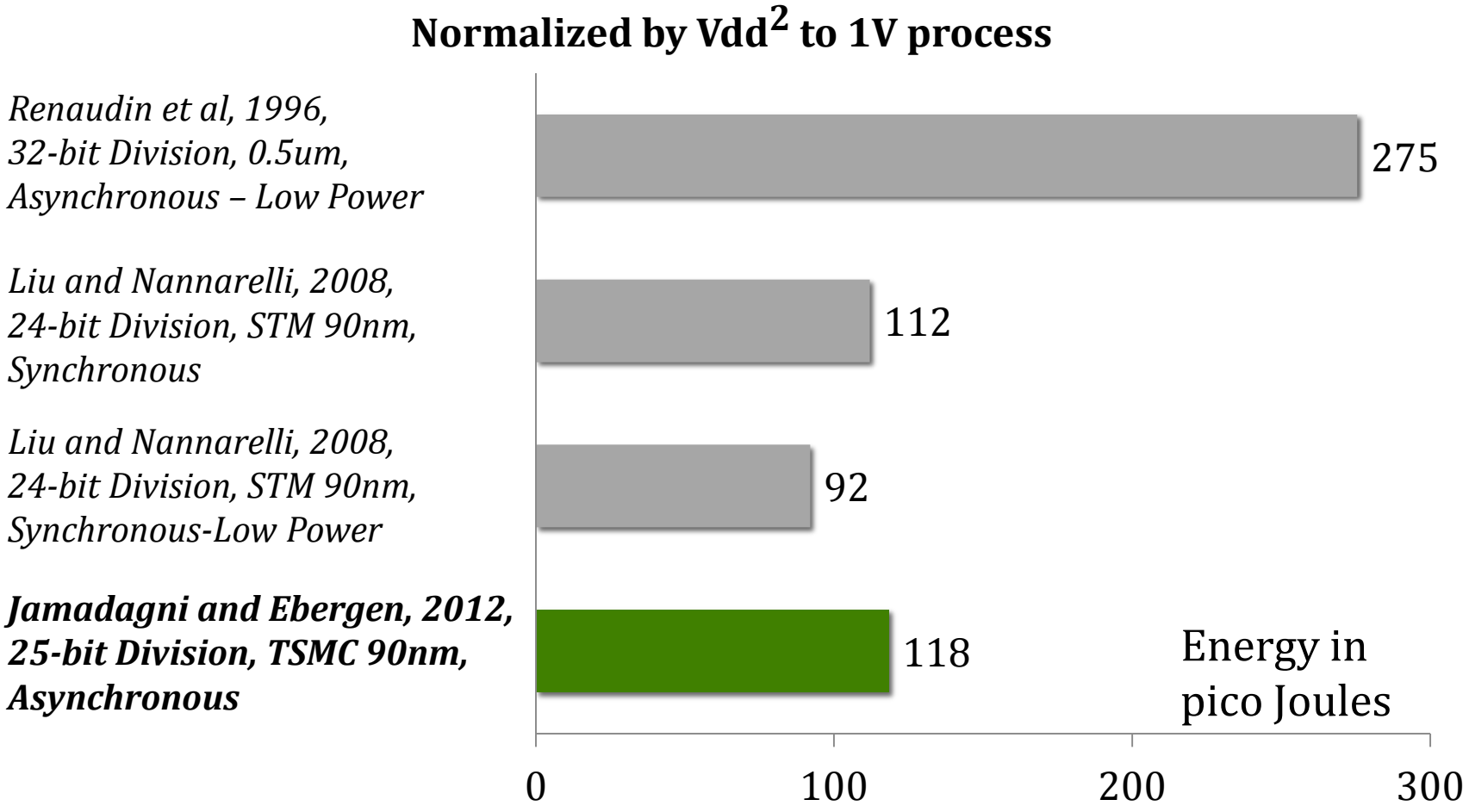
Average Delay

- From SPICE
  - Shift Path = 6 FO4
  - Add Path = 8.5 FO4
- Asynchronous Design
  - Sometimes retires two bits and Shift steps are quicker
  - Average delay per bit is 6.3 FO4
- Synchronous Design
  - Sometimes retires two bits
  - Average delay per bit is 7.4 FO4

Delay per Quotient bit Normalized



Energy per Division



Outline

- Introduction
- Divine Division
- Hardware Design
- Results
- Conclusion

Summary and Conclusion

- An Asynchronous design
  - Exploits the average case behavior of the Divine Division algorithm
  - Exploits the disparity in data path delays
- Future Work
  - Add computation in the feedback path
    - Insert another data path
    - Mitigate the effect of sequencing overhead
  - Reduce power consumption
    - Controlling the data inputs to the adder

Questions?

Thank You ☺