Efficient FPGA Implementation of Digit Parallel Online Arithmetic Operators

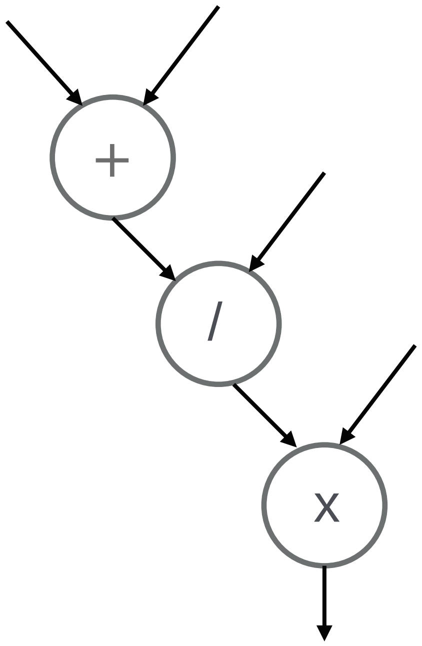
Kan Shi[1], David Boland[2] and George A. Constantinides[1]

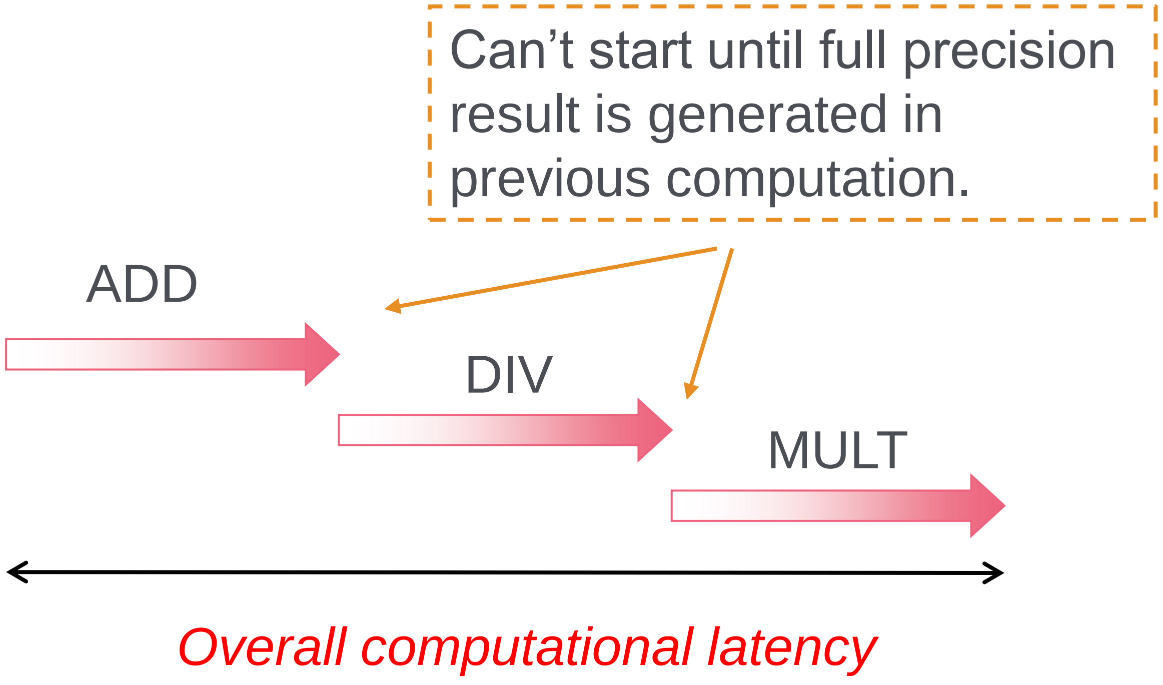
1: EEE Department, Imperial College London, UK
2: ECSE Department, Monash University, Australia

Dec 11, 2014
FPT'14   Shanghai, China

It is all about Performance

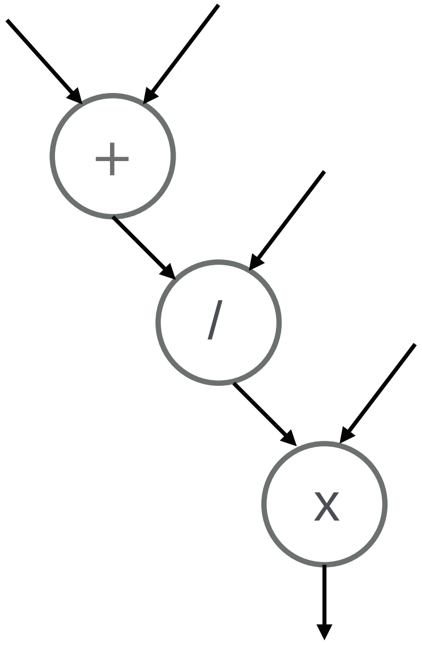
- In conventional arithmetic, operations have different **computing directions.**
- Potentially causes large latency.

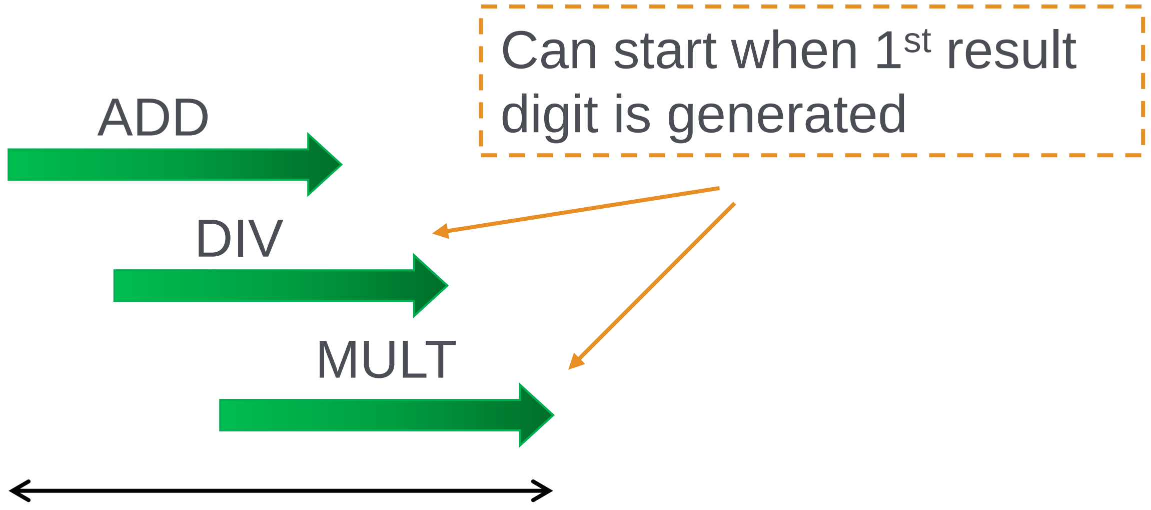



Can't start until full precision result is generated in previous computation.
ADD
DIV
MULT
Overall computational latency

It is all about Performance

- "Online Arithmetic": both inputs and outputs are processed **MSB-first.**
- Enables parallelism among multiple operations.



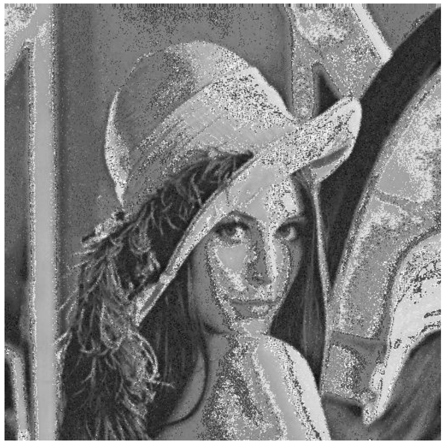


Overall computational latency: Largely reduced

Online Arithmetic is "Overclocking Friendly"

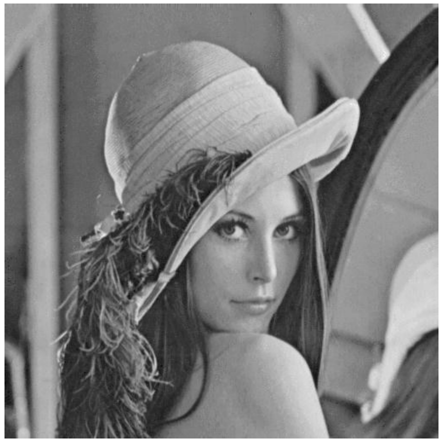
- It's beneficial to operate beyond the worst case
  - Errors due to timing violations only occur rarely.
- Online Arithmetic fails gracefully under overclocking
  - Overclocking errors only occur at LSBs.

Example: Image Filter (DAC'14), 25% Speed-up



Traditional Arithmetic

SNR=9dB

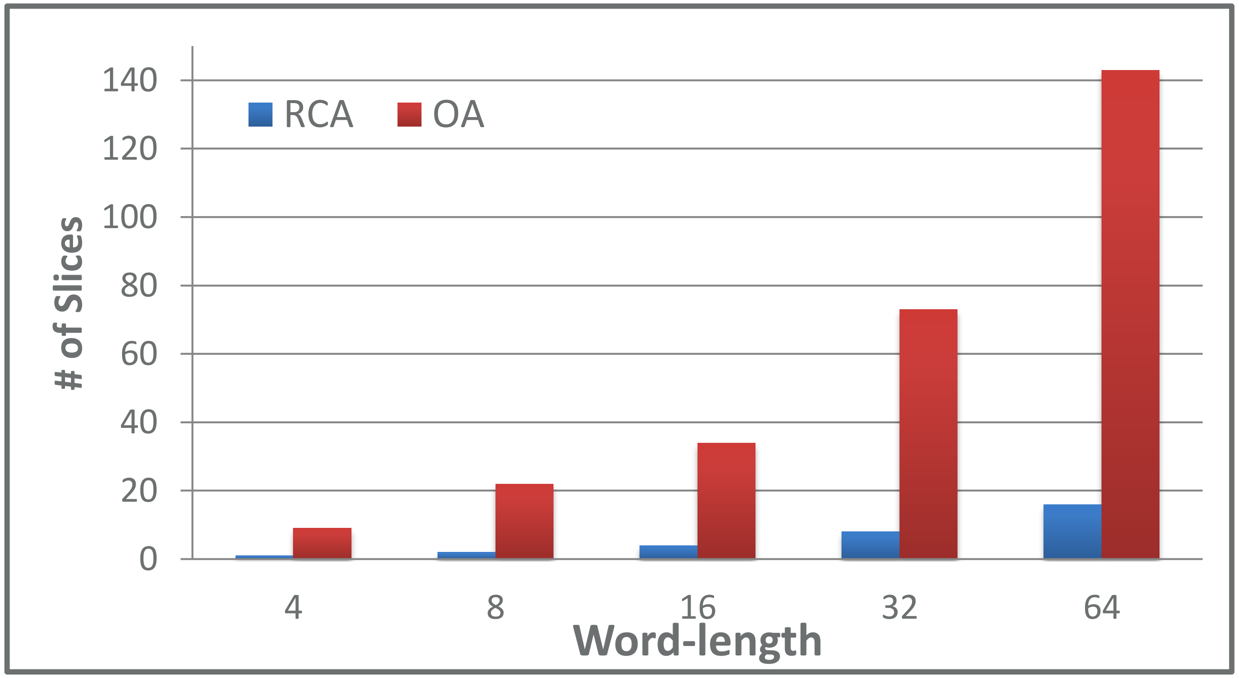


Online Arithmetic

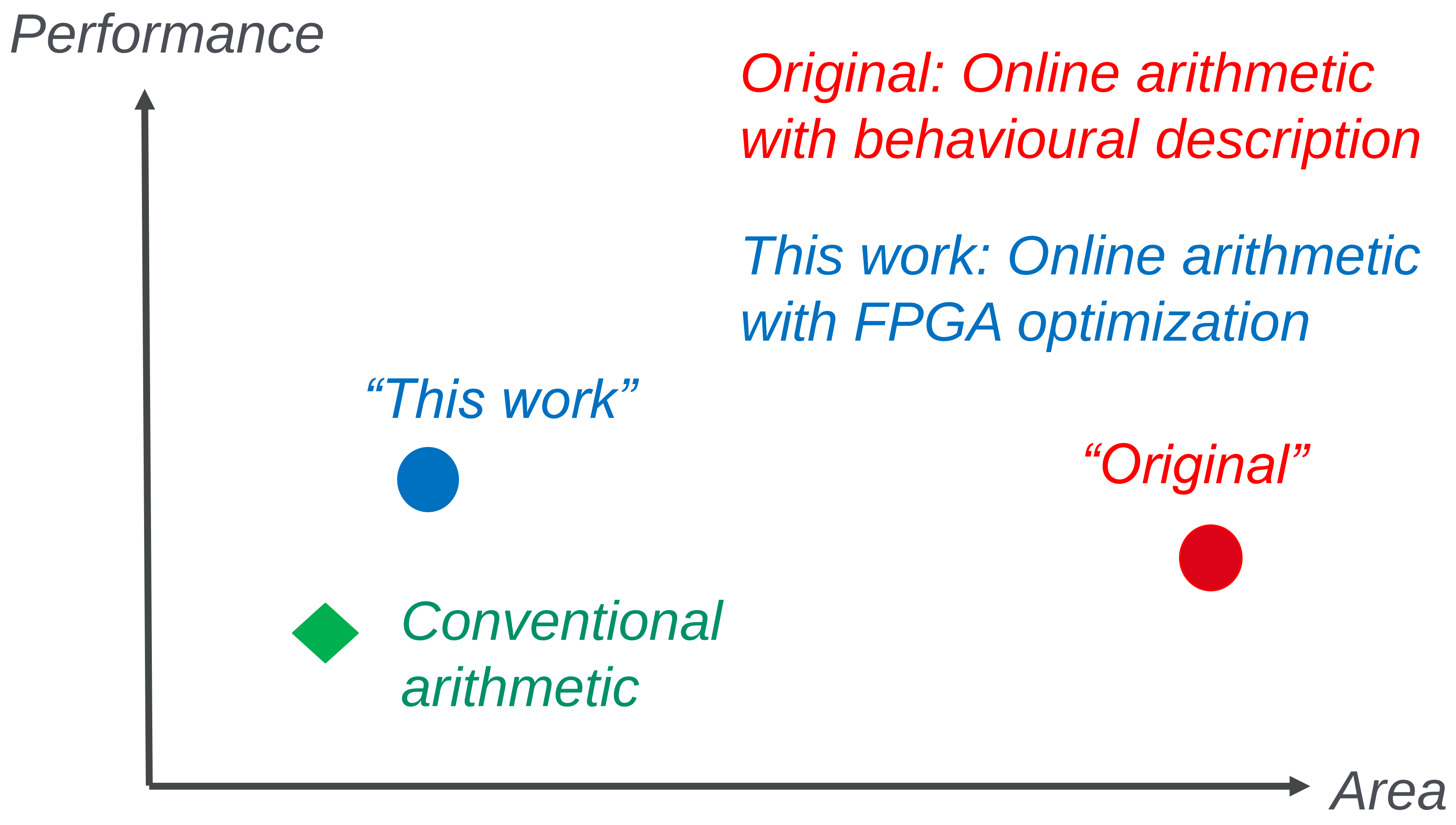
SNR=36.2dB

However…

- Online arithmetic operators cost large area
  - As it employs a redundant number system
- Example: Online adder is 3x ~ 11x larger than RCA



Our Vision


Performance
Original: Online arithmetic with behavioural description
This work: Online arithmetic with FPGA optimization
"This work"
"Original"
Conventional arithmetic
Area

Our Contributions

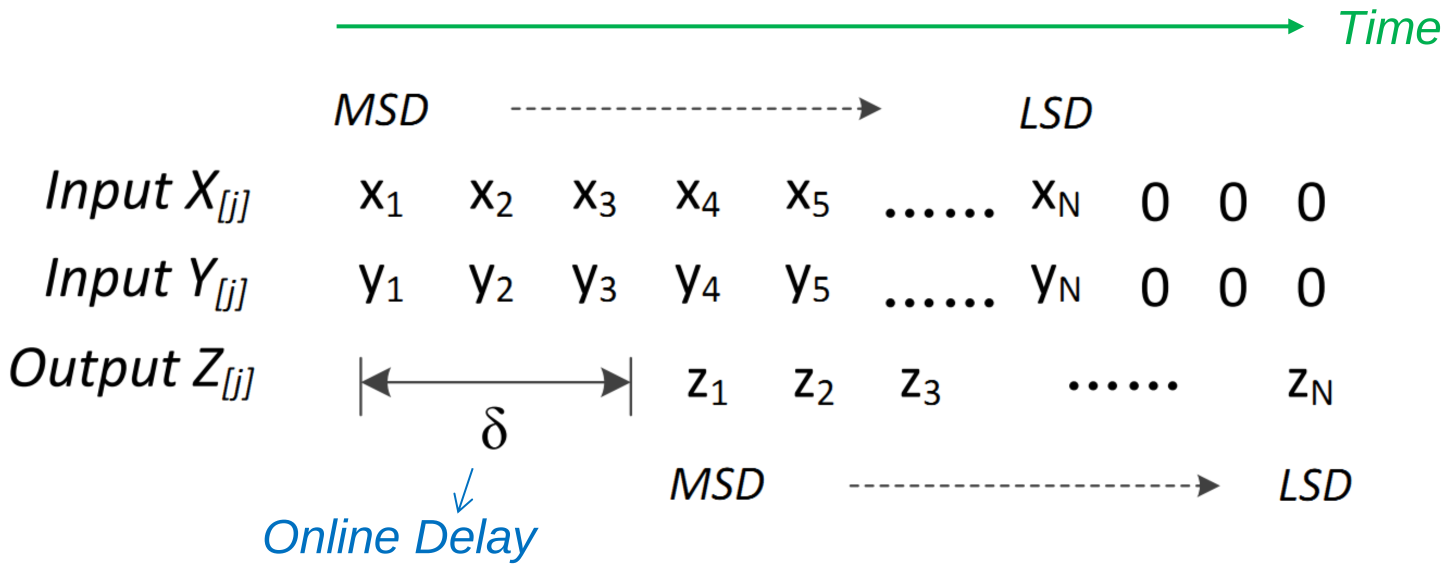
- Efficient methods to map online adders and multipliers to modern FPGAs;
- A method to implement correctly-rounded online multipliers for a chosen output precision;
- Demonstration of performance improvements in terms of area and frequency.

Outline

- Motivation
- **Background: Online arithmetic**
  - **Key features**
- Digit-parallel online adder on FPGAs
- Digit-parallel online multiplier on FPGAs
- Conclusion

Online Arithmetic: Background

- What is it?
  - Online Arithmetic performs computation in an MSD-first manner.
  - It is initially designed for digit-serial operation.


Time
MSD
LSD
Input $X_{[j]}$
x_1 x_2 x_3 x_4 x_5 …… x_N 0 0 0
Input $Y_{[j]}$
y_1 y_2 y_3 y_4 y_5 …… y_N 0 0 0
Output $Z_{[j]}$
z_1 z_2 z_3 …… z_N
δ
Online Delay
MSD
LSD

Online Arithmetic: Background

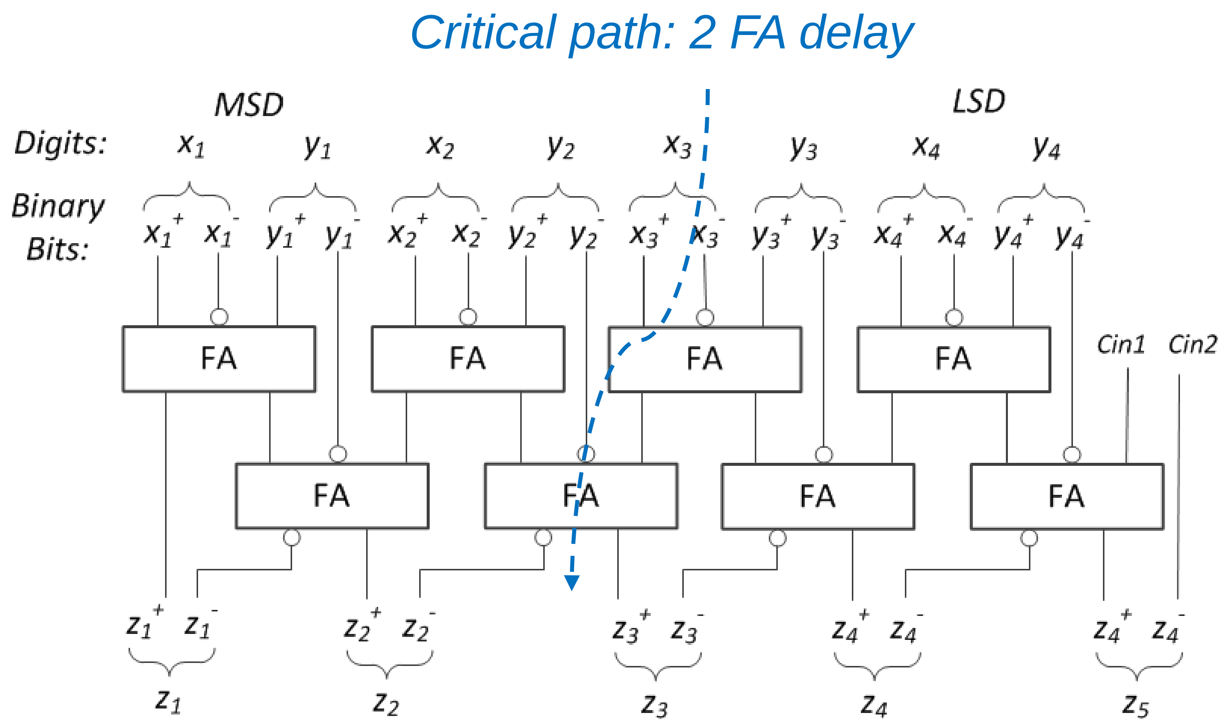
- What is it?
  - Online Arithmetic performs computation in an **MSD-first manner**.
  - It is initially designed for digit-serial operation.
- How is this possible?
  - Requires a flexibility in computing outputs only based on partial information of inputs.
  - Redundancy in the number representation.

	Binary Standard	**Binary Redundant**
Digit Set	{0,1}	{-1,0,1}
Example	0.011	0.1(-1)1, 0.10(-1), 0.011, …

Imperial College London

Online Arithmetic: Adder

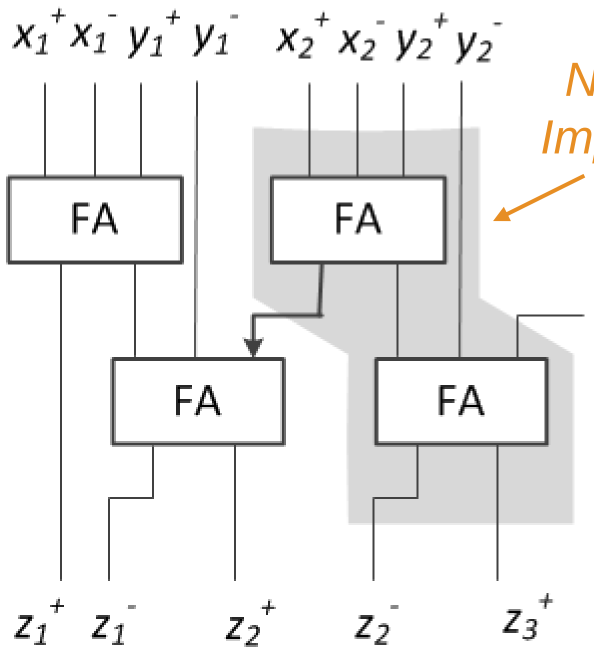
- Features:
  - Binary, digit parallel
  - Fast: critical path is irrelevant to operand precision


Critical path: 2 FA delay
MSD
LSD
Digits:
x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4
Binary Bits:
x_1^+ x_1^- y_1^+ y_1^- x_2^+ x_2^- y_2^+ y_2^- x_3^+ x_3^- y_3^+ y_3^- x_4^+ x_4^- y_4^+ y_4^-
FA
Cin1
Cin2
z_1^+ z_1^- z_2^+ z_2^- z_3^+ z_3^- z_4^+ z_4^- z_4^+ z_4^-
z_1 z_2 z_3 z_4 z_5

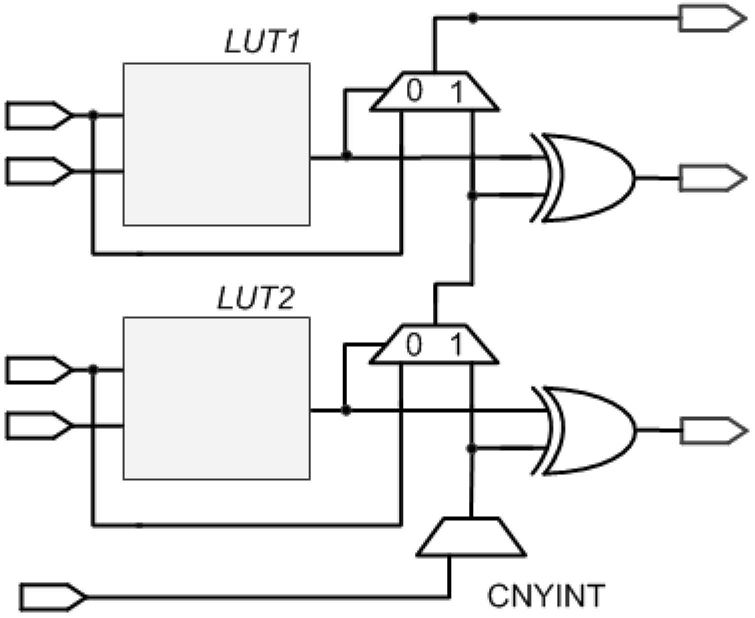
Online Adder on FPGAs

- Previous work described an implementation method
    - Targeting FPGAs with 4-input LUTs and 2 LUTs per Slice

Online Adder (Partial)


x_1^+ x_1^- y_1^+ y_1^- x_2^+ x_2^- y_2^+ y_2^-
FA
FA
FA
FA
z_1^+ z_1^- z_2^+ z_2^- z_3^+
Normal ASIC Implementation

Slice Structure of Spartan FPGA

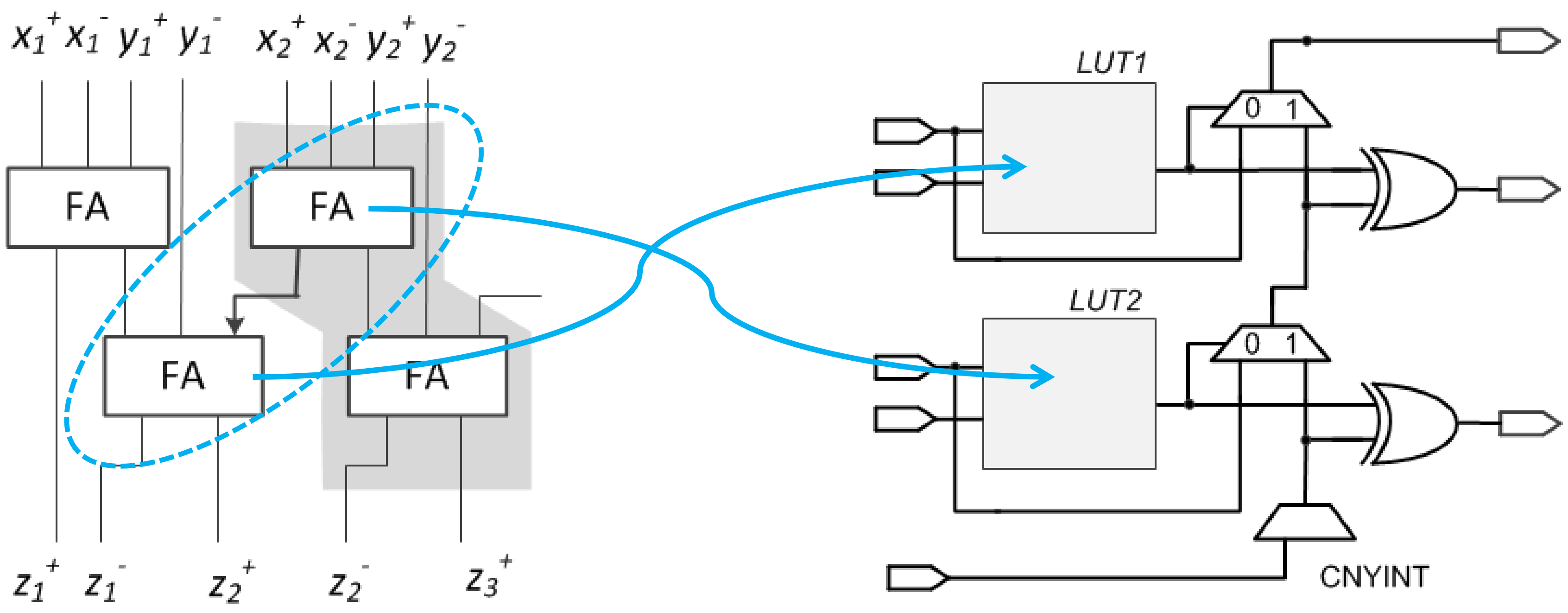


Online Adder on FPGAs

- Previous work described an implementation method
  - Targeting FPGAs with 4-input LUTs and 2 LUTs per Slice
  - Take advantage of fast carry logic in FPGAs

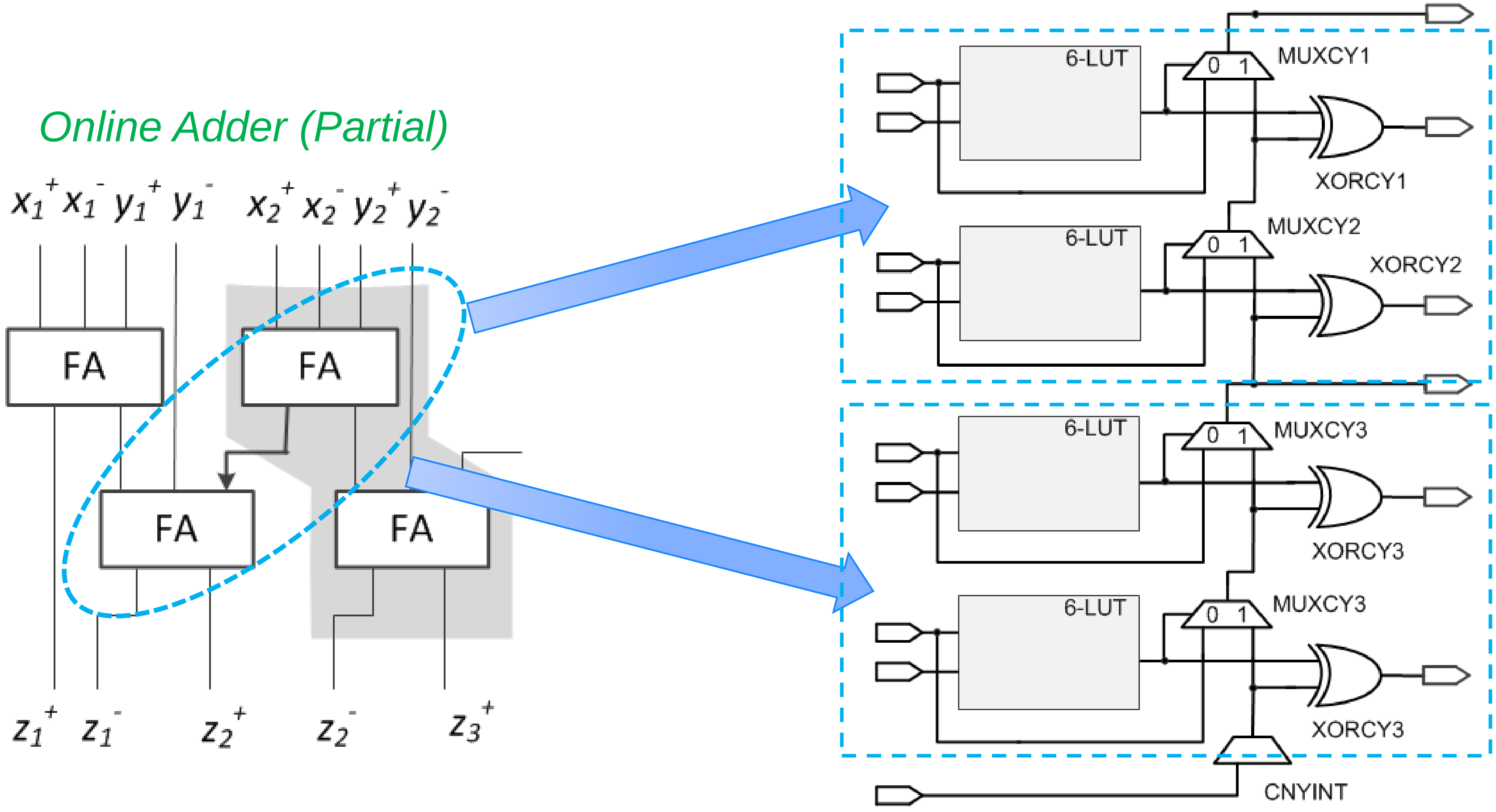
Online Adder (Partial)

Slice Structure of Spartan FPGA


x_1^+ x_1^- y_1^+ y_1^- x_2^+ x_2^- y_2^+ y_2^-
FA
FA
FA
FA
z_1^+ z_1^- z_2^+ z_2^- z_3^+
LUT1
LUT2
0 1
0 1
CNYINT

Online Adder on FPGAs

- However, can't be directly applied to modern FPGAs
  - With 6-input LUTs and 4 LUTs per Slice

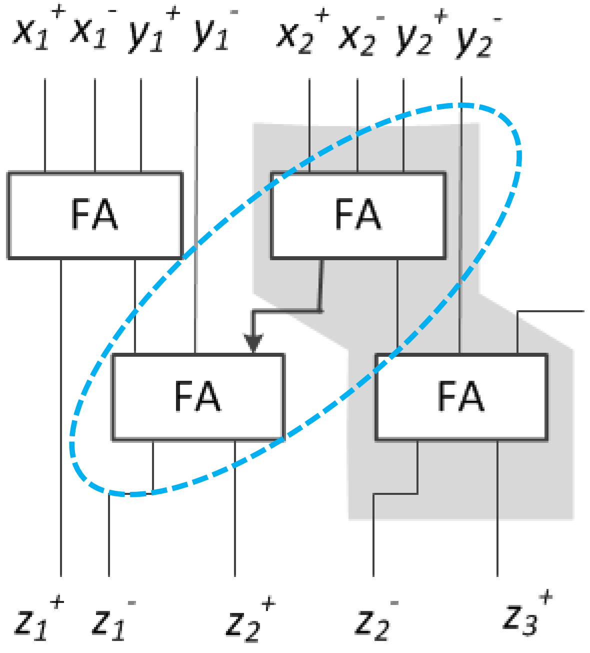

Online Adder (Partial)
x_1^+ x_1^- y_1^+ y_1^- x_2^+ x_2^- y_2^+ y_2^-
FA
FA
FA
FA
z_1^+ z_1^- z_2^+ z_2^- z_3^+
6-LUT
MUXCY1
XORCY1
MUXCY2
XORCY2
MUXCY3
XORCY3
MUXCY3
XORCY3
0 1
CNYINT

Imperial College London

Online Adder on FPGAs

- However, can't be directly applied to modern FPGAs
  - With 6-input LUTs and 4 LUTs per Slice

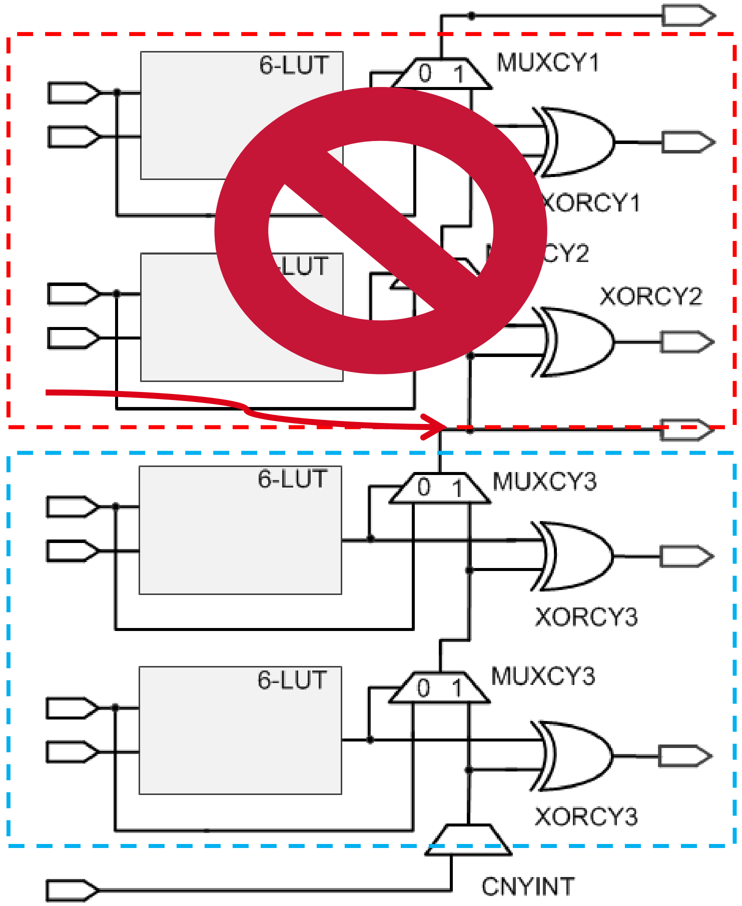
Online Adder (Partial)



x_1^+ x_1^- y_1^+ y_1^- x_2^+ x_2^- y_2^+ y_2^-
FA
FA
FA
FA
z_1^+ z_1^- z_2^+ z_2^- z_3^+

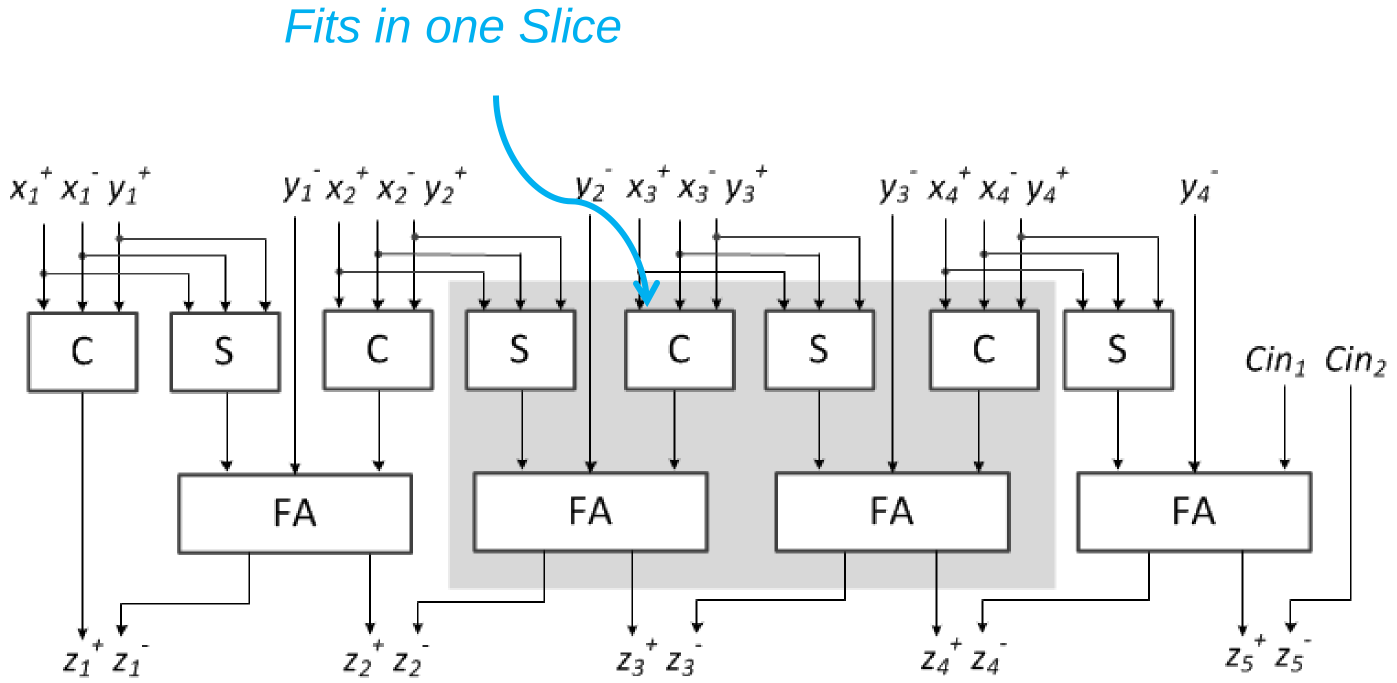
Can't initialize carry in here

Can initialize carry in here



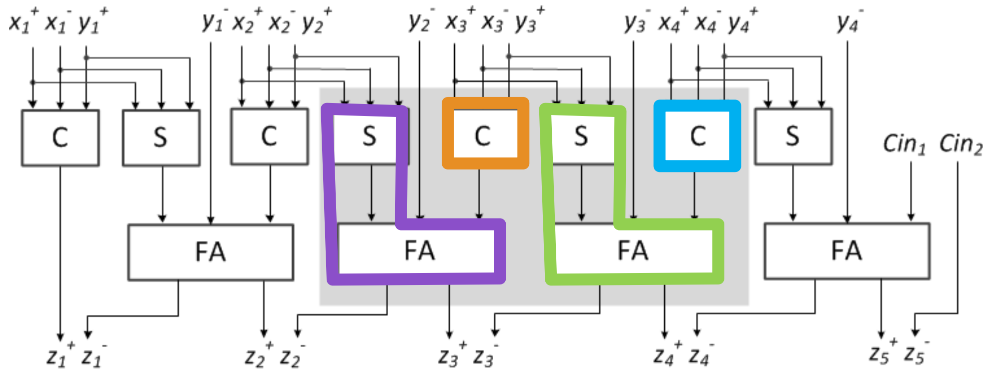
Online Adder on FPGAs: Proposed

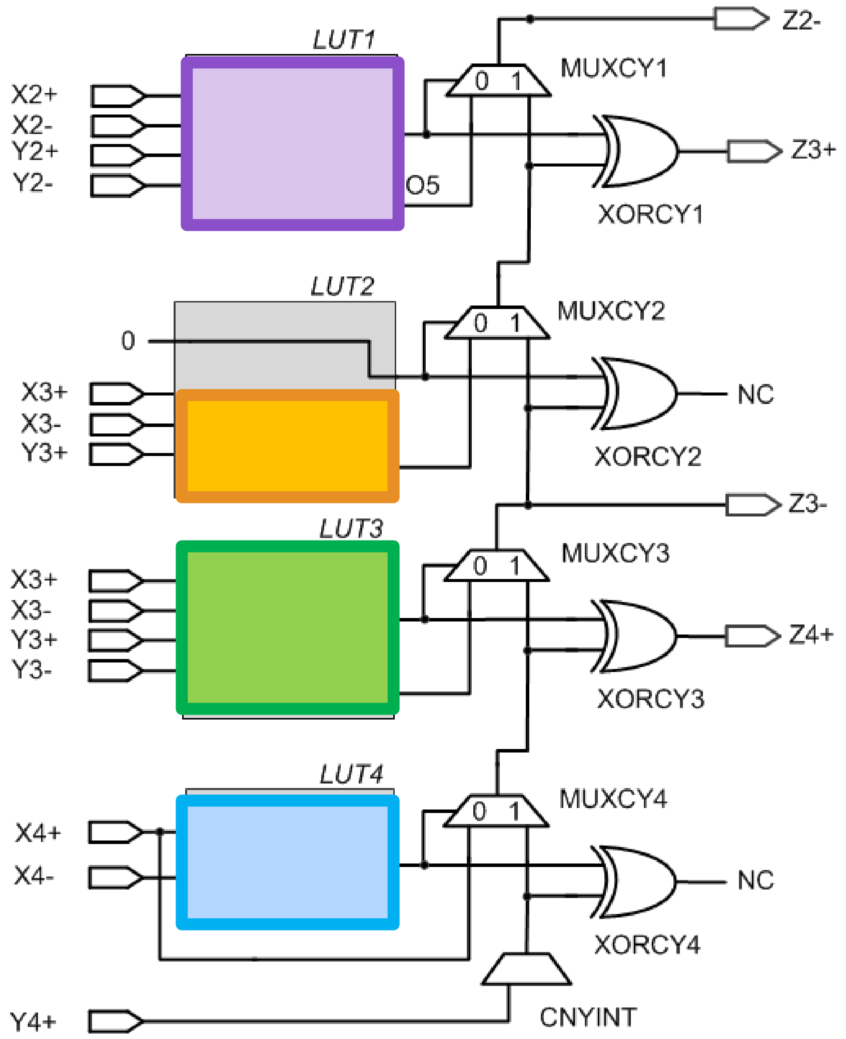
- We propose an equivalent transformation of OA
  - One FA is divided to generate Sum and Carry separately
- Example: 4-digit online adder



Online Adder on FPGAs: Proposed

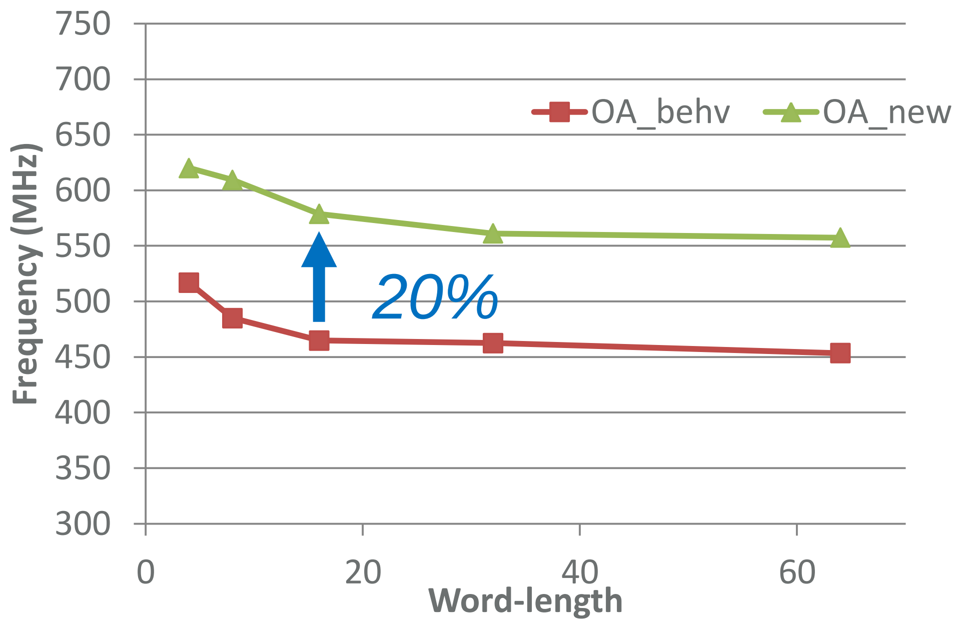
- We propose an equivalent transformation of OA
- FPGA Implementation





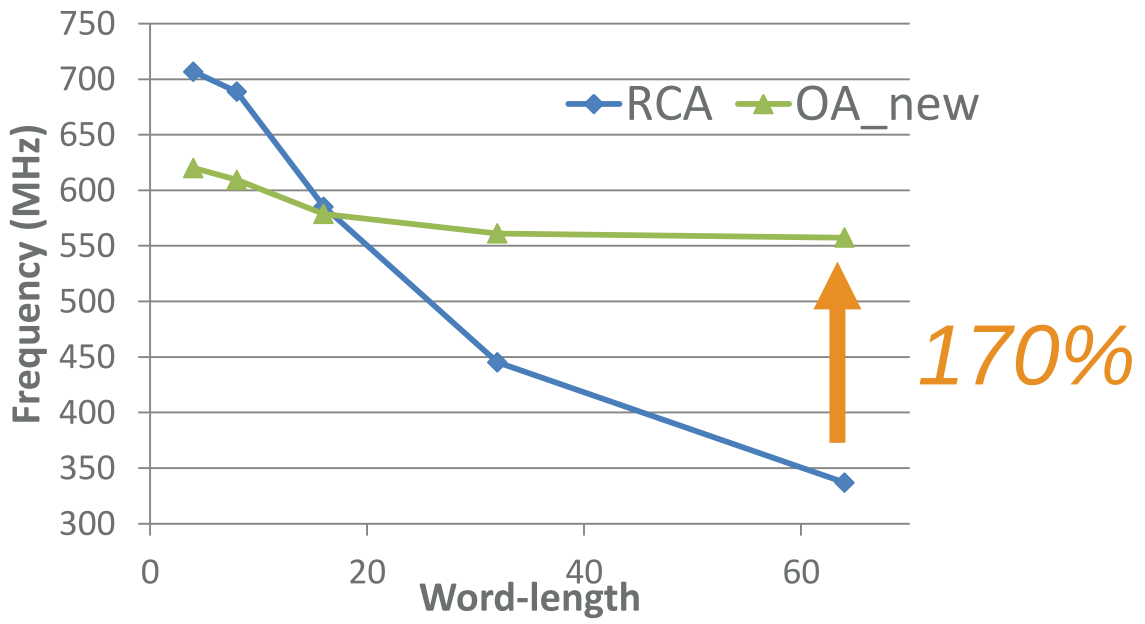
Online Adder on FPGAs: Performance

- Rated frequency comparison
    - OA with direct implementation (OA_behv)
    - Proposed OA (OA_new)



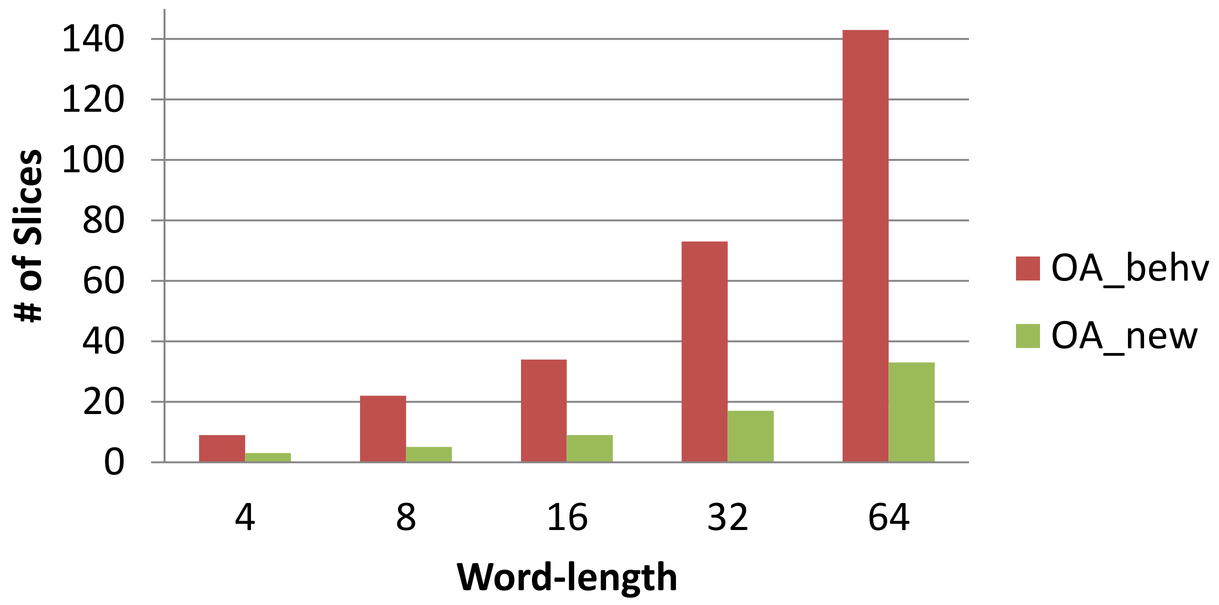
Online Adder on FPGAs: Performance

- Rated frequency comparison
  - OA with direct implementation (OA_behv)
  - Proposed OA (OA_new)
  - RCA



Online Adder on FPGAs: Performance

- Area comparison (Slice)

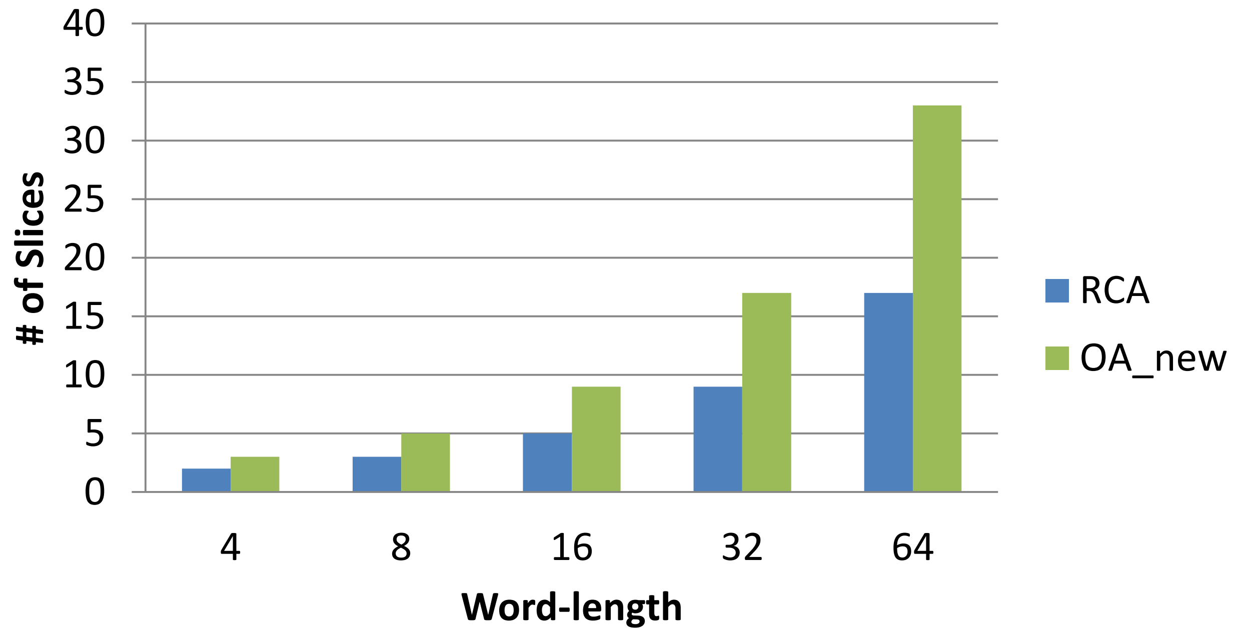


Area savings over direct implementation

WL	4	8	16	32	64
Slices	66.7%	77.3%	73.5%	76.7%	76.9%

Online Adder on FPGAs: Performance

- Area comparison: OA vs RCA



Area overhead over RCA: less than 2x

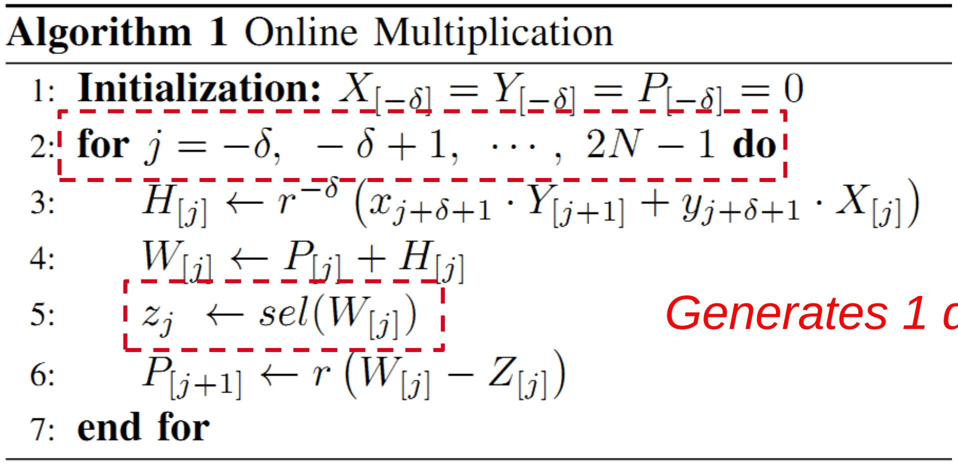
WL	4	8	16	32	64
Slices	1.50x	1.67x	1.80x	1.89x	1.94x

Outline

- Motivation
- Background: Online arithmetic
- Digit-parallel online adder on FPGAs
- **Digit-parallel online multiplier on FPGAs**
    - **Optimization of original algorithm for full precision results**
    - **Structure optimization for half precision results**
    - **Performance analysis: area and frequency**
- Conclusion

Online Multiplier: Digit Parallel Algorithm

- Recurrence algorithm:



Algorithm 1 Online Multiplication

```
1: Initialization: $X_{[-\delta]} = Y_{[-\delta]} = P_{[-\delta]} = 0$
2: for $j = -\delta, \; -\delta + 1, \; \cdots, \; 2N - 1$ do
3:     $H_{[j]} \leftarrow r^{-\delta}\left(x_{j+\delta+1} \cdot Y_{[j+1]} + y_{j+\delta+1} \cdot X_{[j]}\right)$
4:     $W_{[j]} \leftarrow P_{[j]} + H_{[j]}$
5:     $z_j \leftarrow sel(W_{[j]})$
6:     $P_{[j+1]} \leftarrow r\left(W_{[j]} - Z_{[j]}\right)$
7: end for
```

Generates 1 digit per iteration

- More details in the paper
- Take 1 digit input per iteration
- Modify to take 1 partial product per iteration

Online Multiplier: Digit Parallel Algorithm

- Recurrence algorithm:

Algorithm 1 Online Multiplication

```
1: Initialization: $X_{[-\delta]} = Y_{[-\delta]} = P_{[-\delta]} = 0$
2: for $j = -\delta, \ -\delta + 1, \ \cdots, \ 2N - 1$ do
3:     $H_{[j]} \leftarrow r^{-\delta}\left(x_{j+\delta+1} \cdot Y_{[j+1]} + y_{j+\delta+1} \cdot X_{[j]}\right)$
4:     $W_{[j]} \leftarrow P_{[j]} + H_{[j]}$
5:     $z_j \ \leftarrow sel(W_{[j]})$
6:     $P_{[j+1]} \leftarrow r\left(W_{[j]} - Z_{[j]}\right)$
7: end for
```

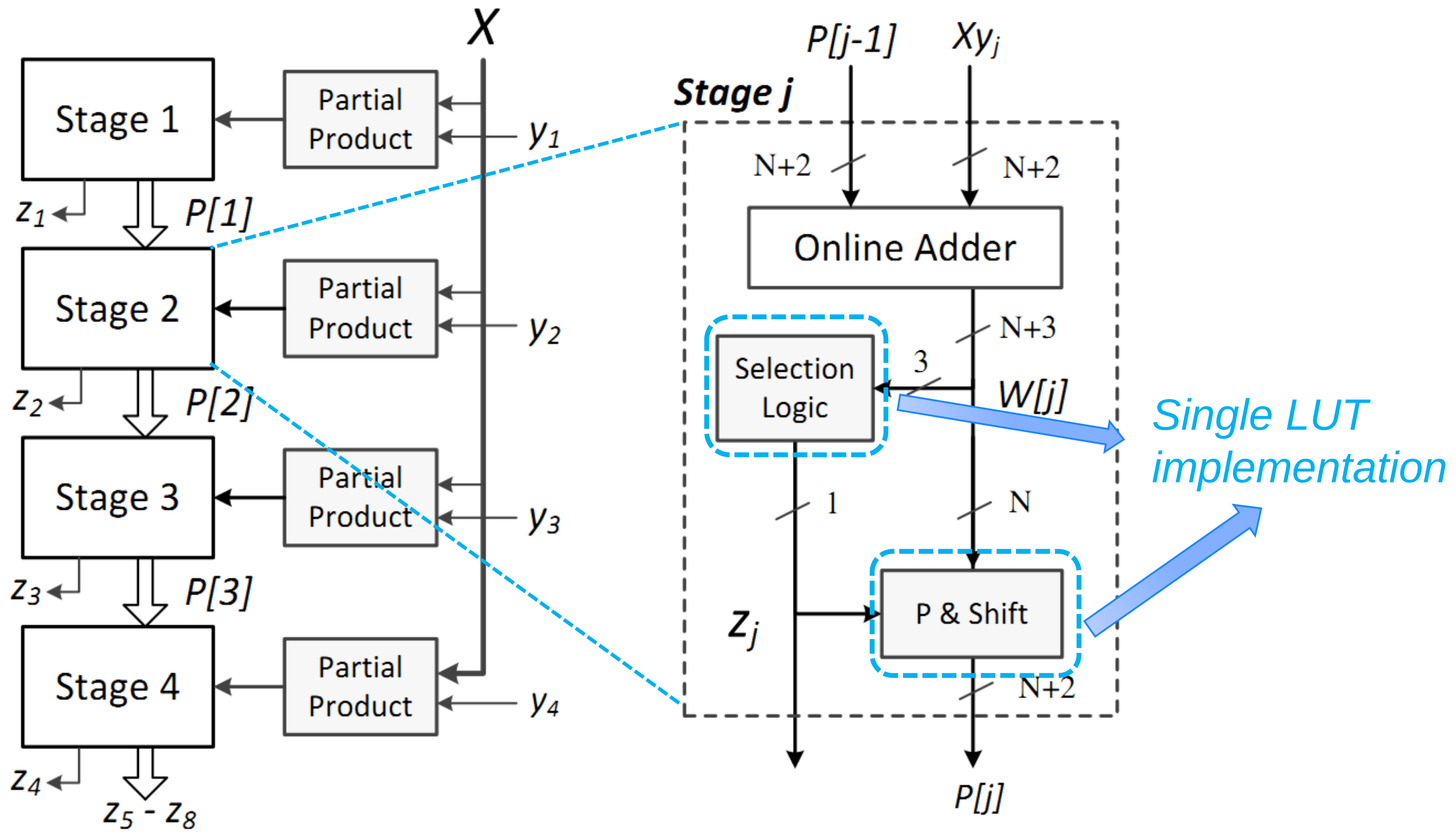
- Proposed algorithm

Algorithm 2 Digit Parallel Online Multiplication

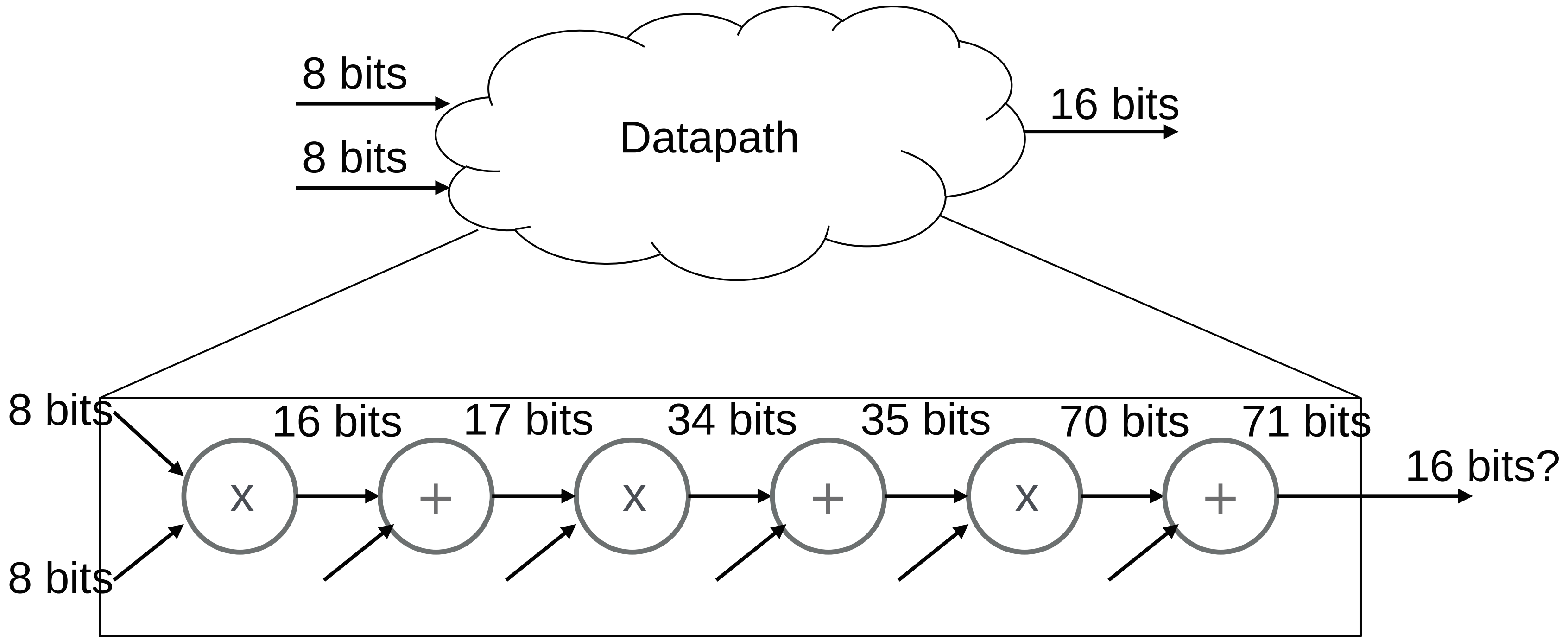
```
1: Initialization: $P_{[0]} = 0$
2: for $j = 1, \ 2, \ \cdots, \ N$ do
3:     $Xy_j \leftarrow X \cdot y_j$        Partial product
4:     $W_{[j]} \leftarrow P_{[j-1]} + Xy_j$
5:     $z_j \ \leftarrow sel(W_{[j]})$
6:     $P_{[j+1]} \leftarrow r\left(W_{[j]} - Z_{[j]}\right)$
7: end for
8: $Z_{[N+1:2N]} \leftarrow frac(W_{[N]})$
```

Online Multiplier: Structure

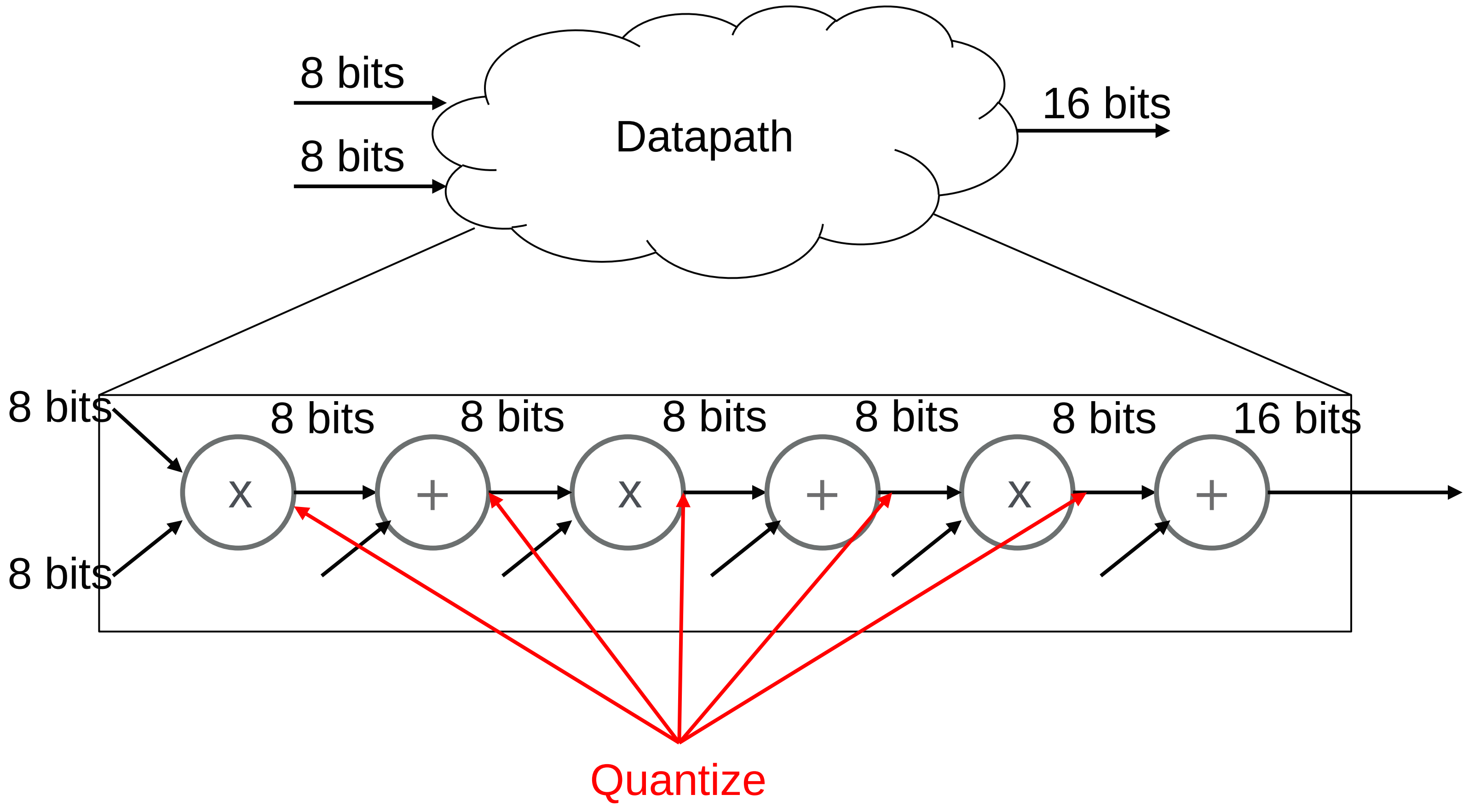
- Example: 4-digit online multiplier



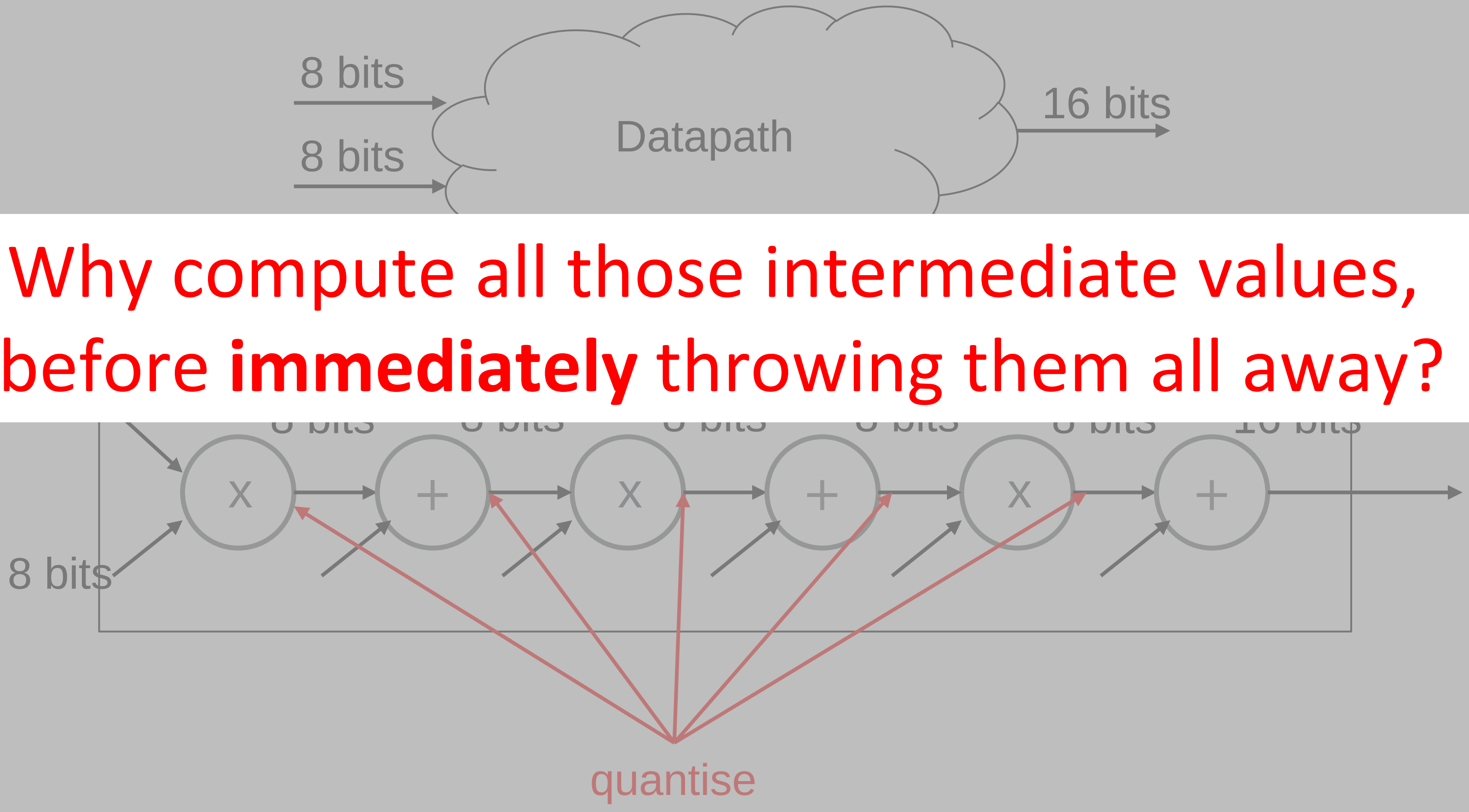
Recovering the area



Recovering the area



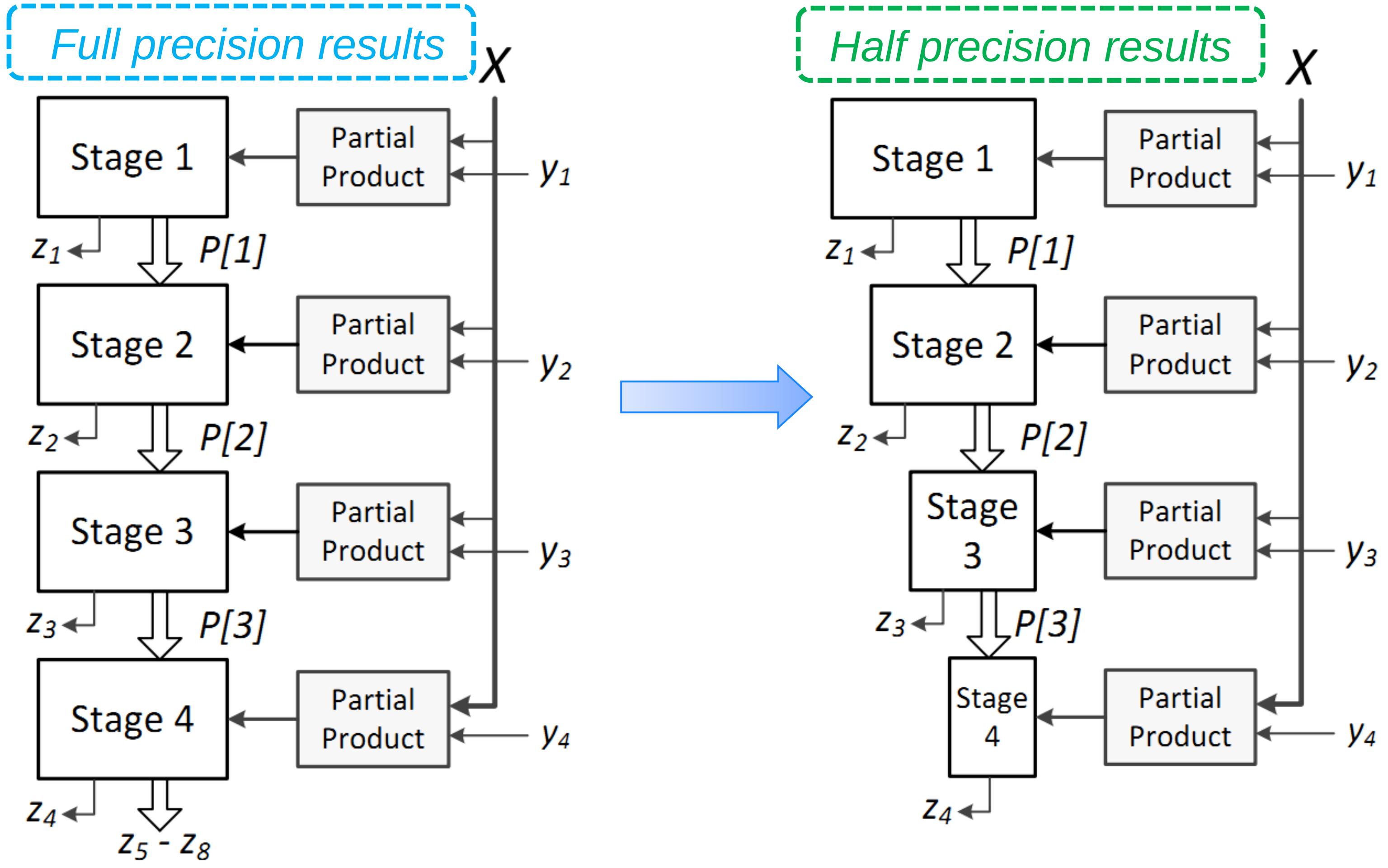
Recovering the area



Why compute all those intermediate values, before **immediately** throwing them all away?

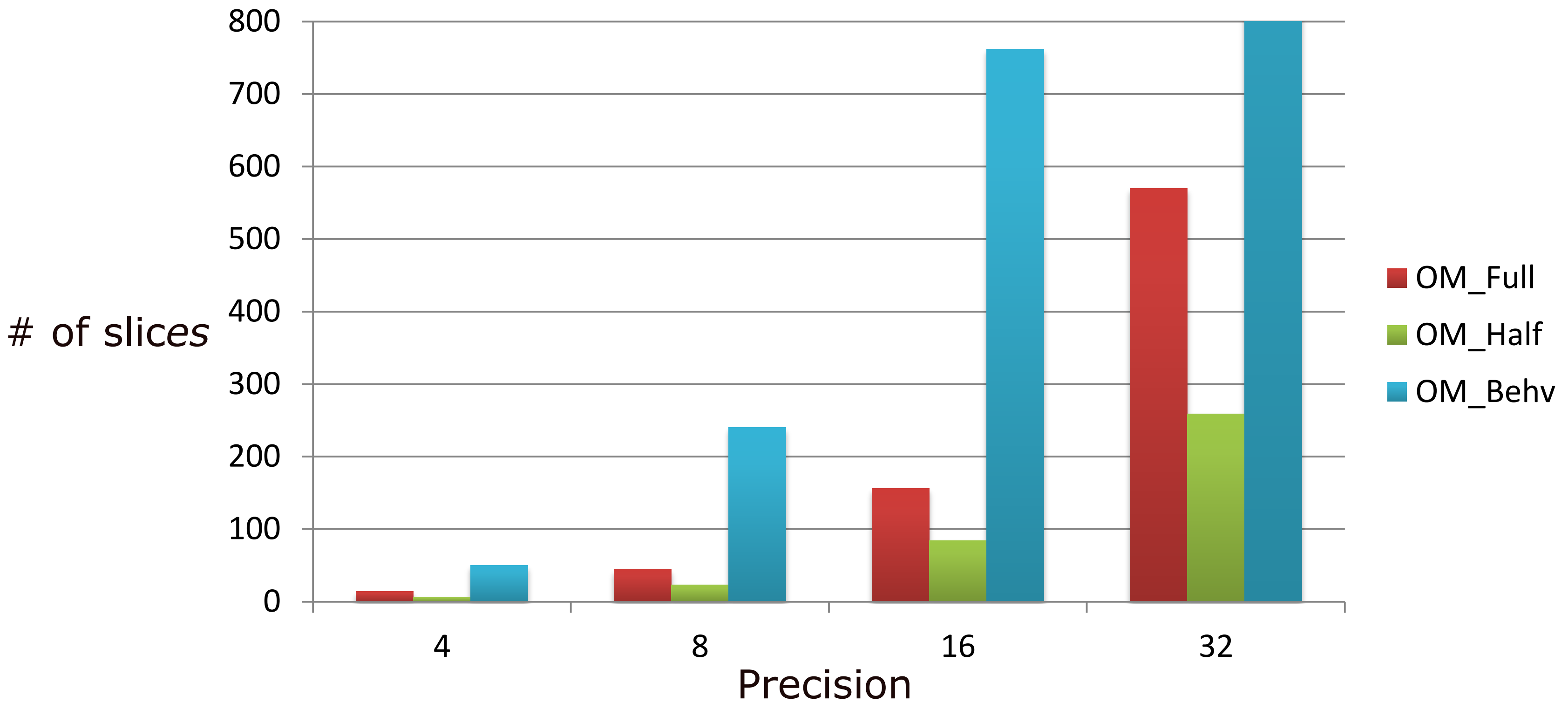
Online Multiplier: Half Precision Results

- Example: 4-digit online multiplier



Imperial College
London

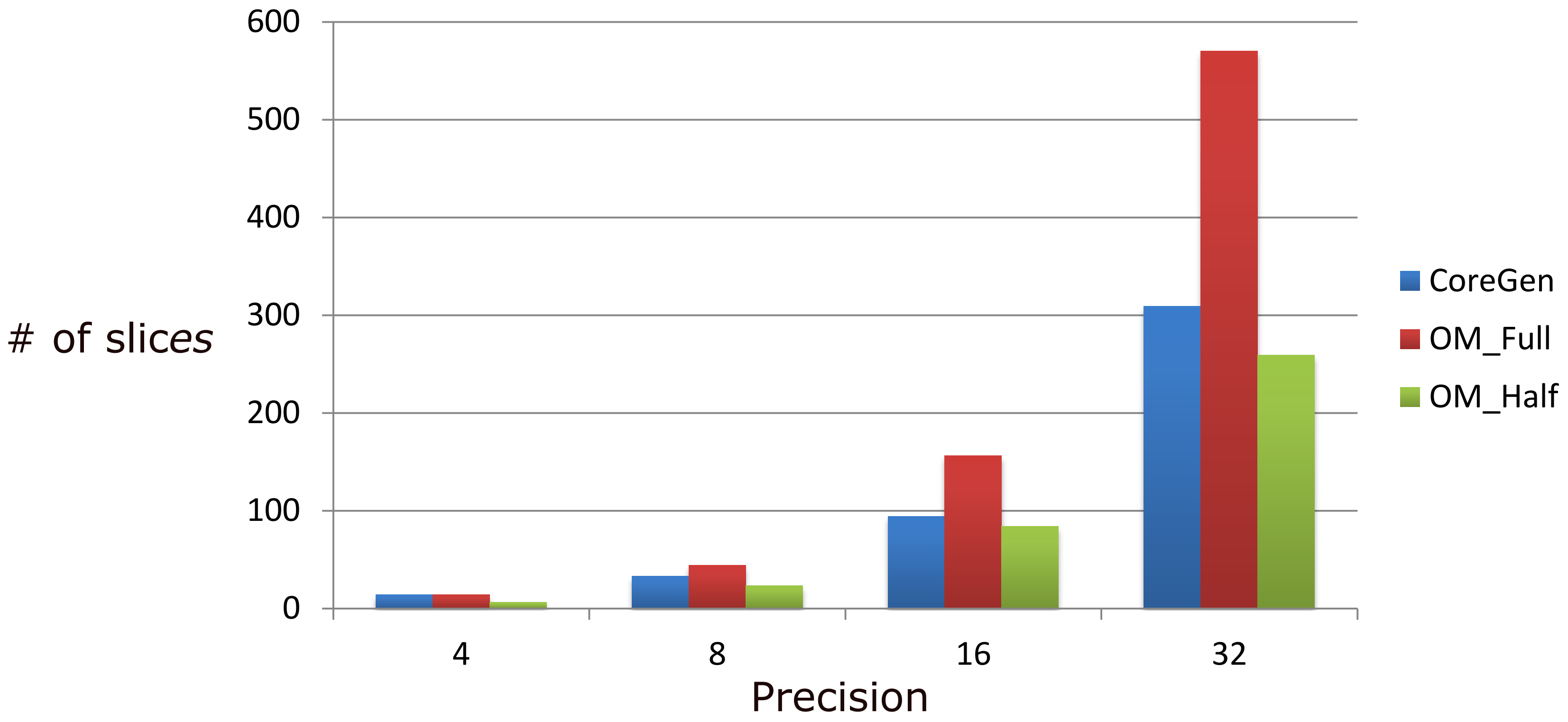
Area Comparison: Slice



Area savings over direct implementation (OM_behv)

WL	4	8	16	32
OM_Full	72.0%	81.7%	79.5%	68.6%
OM_Half	88.1%	90.4%	89.0%	85.7%

Area Comparison: Slice



Area savings/overhead over CoreGen multiplier

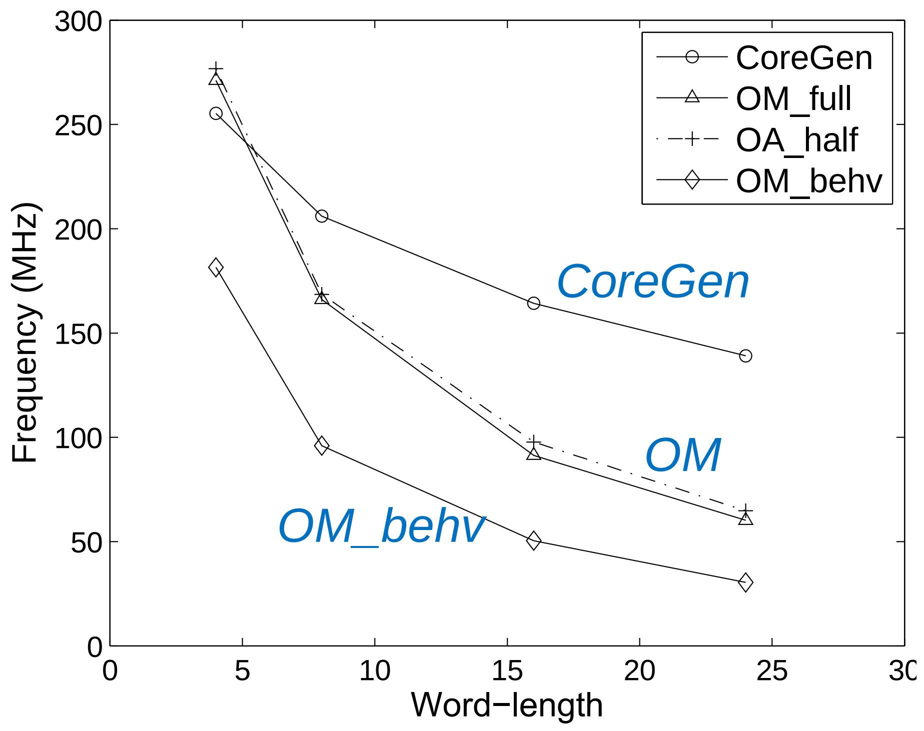
WL	4	8	16	32
OM_Full	0%	33.3%	65.9%	84.4%
OM_Half	57.1%	30.4%	10.7%	16.2%

Imperial College London

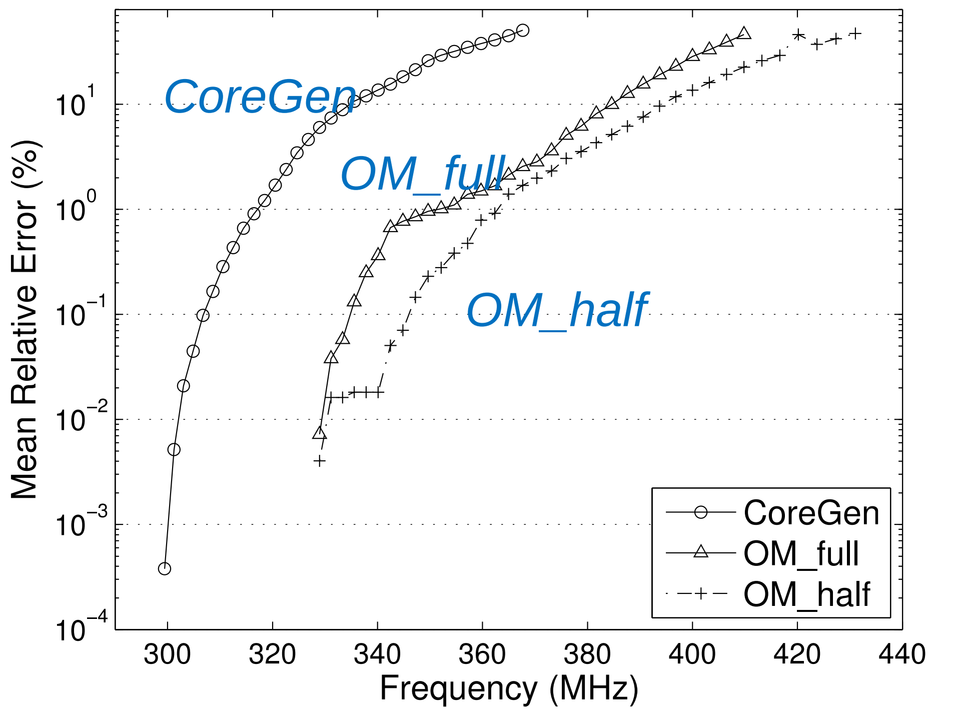
Online Multiplier: Performance

- Frequency comparison

Rated Frequencies



Overclocked 8-digit design: error vs freq



Conclusion

- Efficient mapping of digit parallel online arithmetic operators to FPGAs
- Our mapping method targets modern FPGAs:
  - 6-input LUTs and 4 LUTs per Slice
- Empirical results demonstrate benefits over direct implementation:
  - Area reduction
  - Frequency speed-ups