ABOUT THE RELEVANCE OF MULTISPECULATION IN HLS

Alberto A. Del Barrio García[1], Román Hermida[1], Seda Ogrenci Memik[2], María C. Molina[1], José M. Mendías[1]

[1]Complutense University of Madrid

[2]Northwestern University

OUTLINE

- Introduction
  - Speculative Functional Units
- Multispeculative Functional Units
- Multispeculative datapaths
  - Addition Chains
  - Binary Addition Trees
  - Generic Additive Trees
- Some results
- Conclusions and future lines of work

OUTLINE

- **Introduction**
  - Speculative Functional Units
- Multispeculative Functional Units
- Multispeculative datapaths
  - Addition Chains
  - Binary Addition Trees
  - Generic Additive Trees
- Some results
- Conclusions and future lines of work

INTRODUCTION

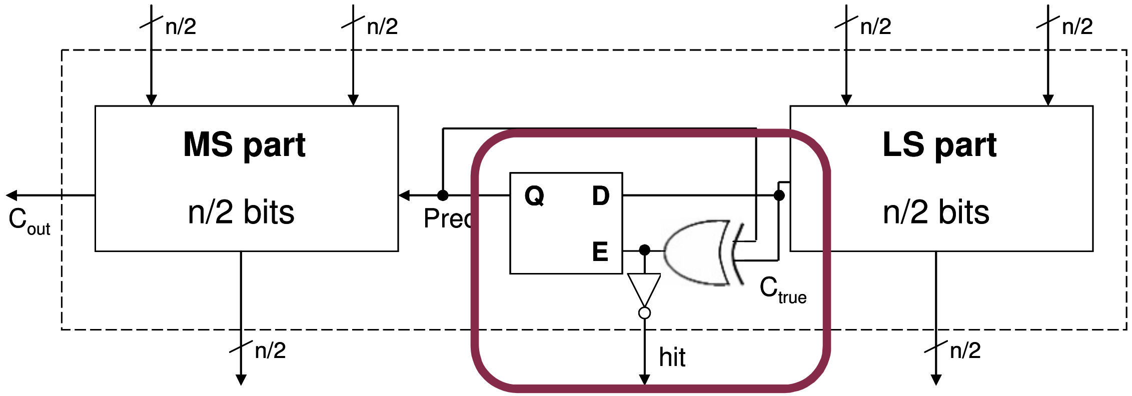
- HPC big question
  - **High Performance … but at what cost ??!**
  - Power, Energy, Area must be considered too
- Additions and products are the most common operations in datapaths
  - Products are based on additions
- Efficient Adders and Additive Structures
  - Building efficient basic blocks is essential
  - **But the ability to handle them is the key**

INTRODUCTION

- Classical Adders [Hwa79, Kor02]
  - Examples
    - Ripple Carry, Carry Select, Carry Skip
    - Carry Lookahead, Prefix Adders
  - Always work with the longest calculus time
    - Huge area/power penalty in the fastest designs
    - Many cases ***do not really need*** the longest path
- Variable Latency FUs
  - Relax some logic conditions to mostly work in ***fast mode***
  - Less area/power than Fixed Latency counterparts
  - Asynchronous and synchronous designs
  - Speculative Fus
    - Synchronous VLFUs based on carry prediction

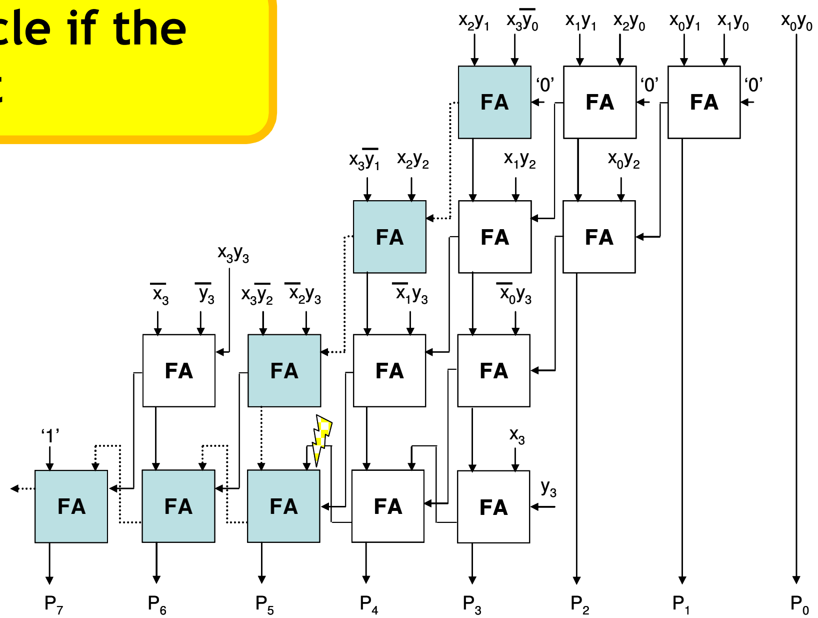
INTRODUCTION: SPECULATIVE FUNCTIONAL UNITS

Additional prediction area is negligible



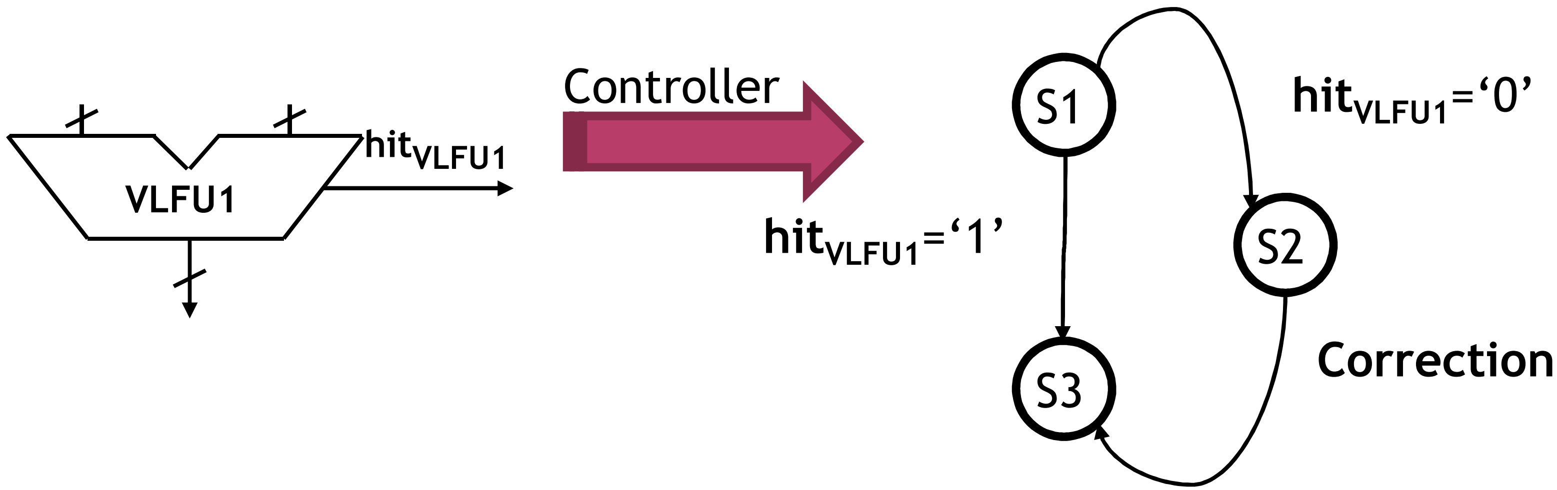
Synchronous principle: 1 short cycle if the adder hits, 2 if it does not

Speculative Multiplier is basically a CSA array with a Speculative Adder in the last stage

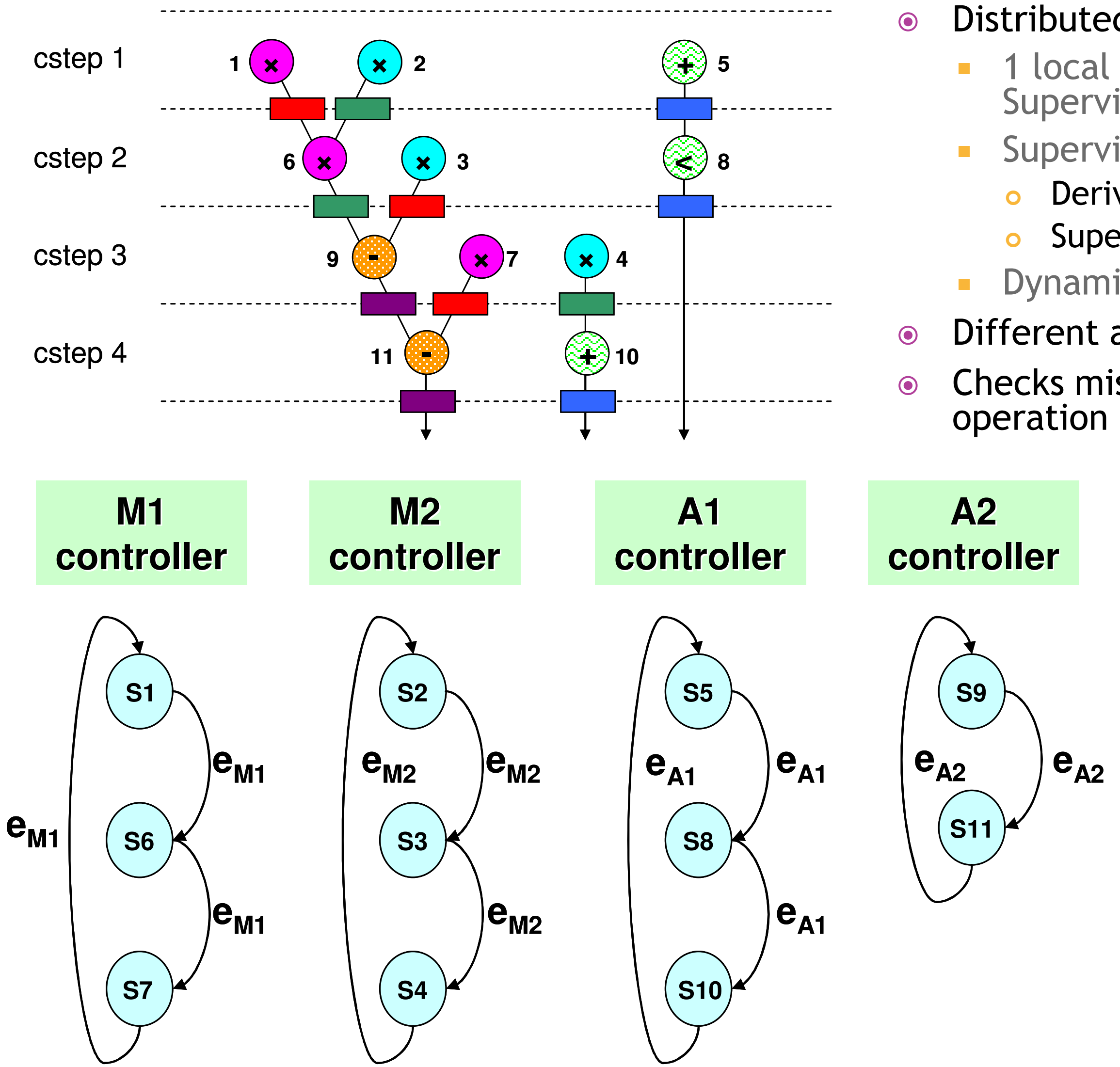


INTRODUCTION: VLFUS AND HLS STATE OF THE ART

- Raghunathan. et al. (2000); Telescopic Units by Benini et. al (1998)
  - Treat the VLFUs as conditional branches
  - This is only feasible with very few VLFUs
    - Exponential number of cases to control
      - Solution: Distributed Controller (Del Barrio et al. 2011)



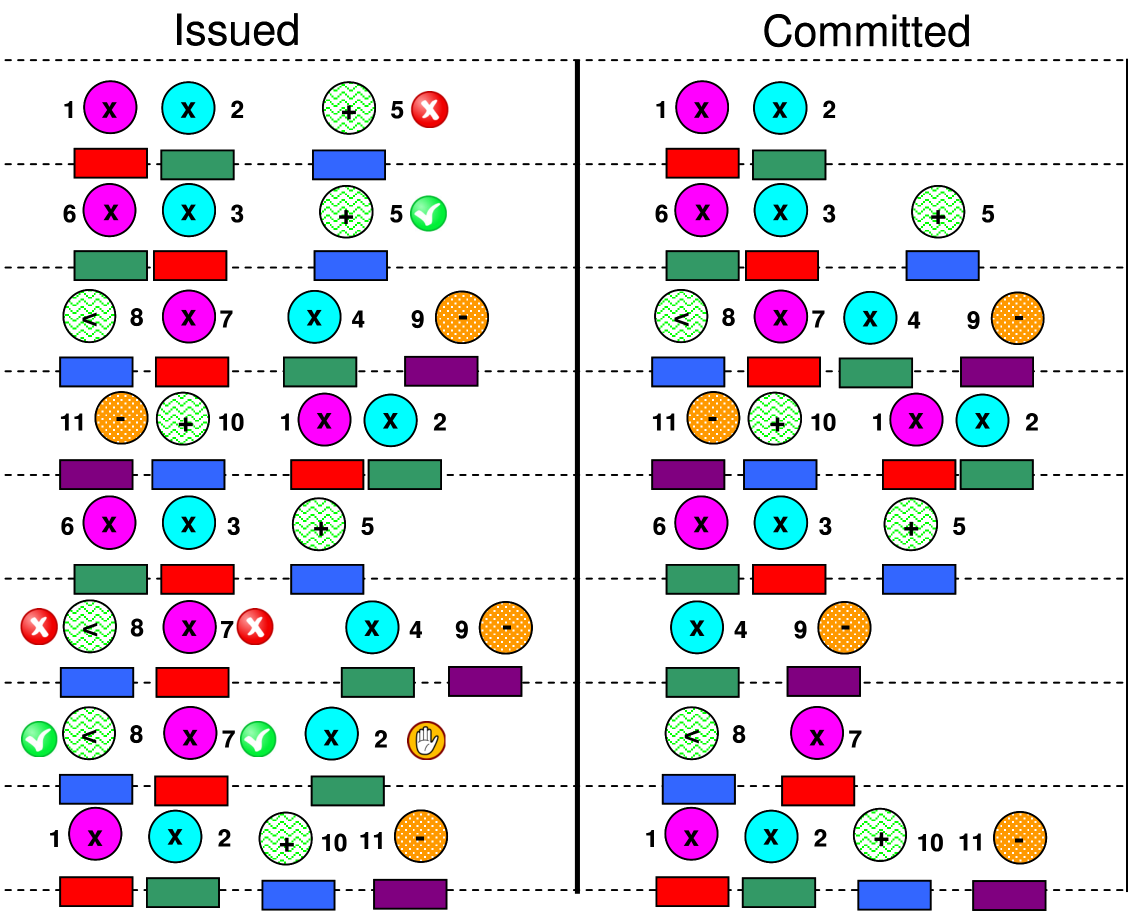
INTRODUCTION: VLFUS AND HLS STATE OF THE ART



- Distributed Controller
  - 1 local controller per FU + Supervisor
  - Supervisor
    - Derived automatically from DFG
    - Supervisor fires the transitions
  - Dynamic scheduling
- Different approach
- Checks mispredictions for every operation

1 state value per operation

INTRODUCTION: VLFUS AND HLS STATE OF THE ART

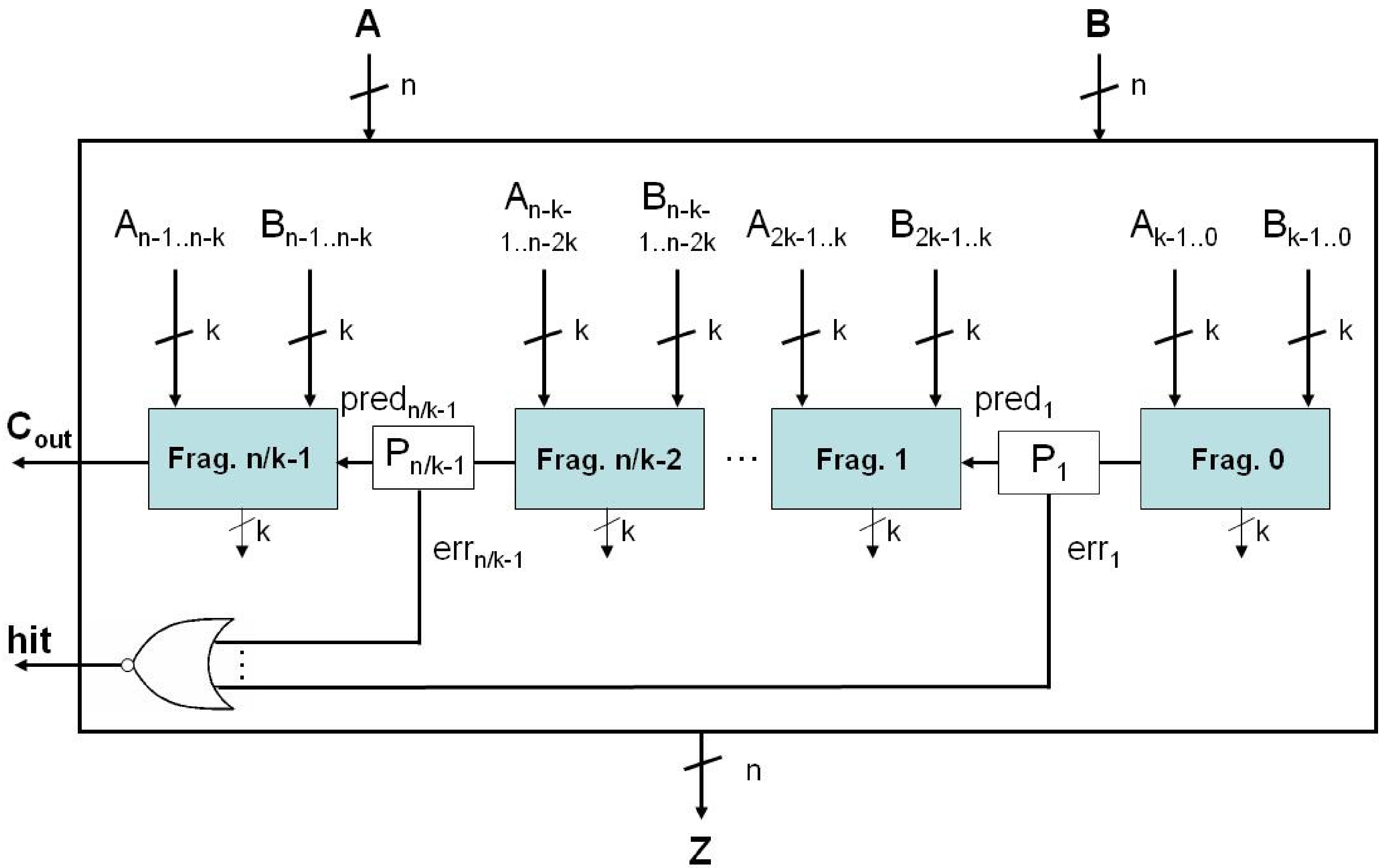


Clock Cycle	M1 State	M1 T	M2 State	M2 T	A1 State	A1 T	A2 State	A2 T
1	S1	0	S2	0	S5	0	S9	0
2	S6	0	S3	0	S5	0	S9	0
3	S7	0	S4	0	S8	0	S9	0
4	S1	1	S2	1	S10	0	S11	0
5	S6	1	S3	1	S5	1	S9	1
6	S7	1	S4	1	S8	1	S9	1
7	S7	1	S2	0	S8	1	S11	1
8	S1	0	S2	0	S10	1	S11	1

OUTLINE

- Introduction
  - Speculative Functional Units
- **Multispeculative Functional Units**
- Multispeculative datapaths
  - Addition Chains
  - Binary Addition Trees
  - Generic Additive Trees
- Some results
- Conclusions and future lines of work

MULTISPECULATIVE FUNCTIONAL UNITS

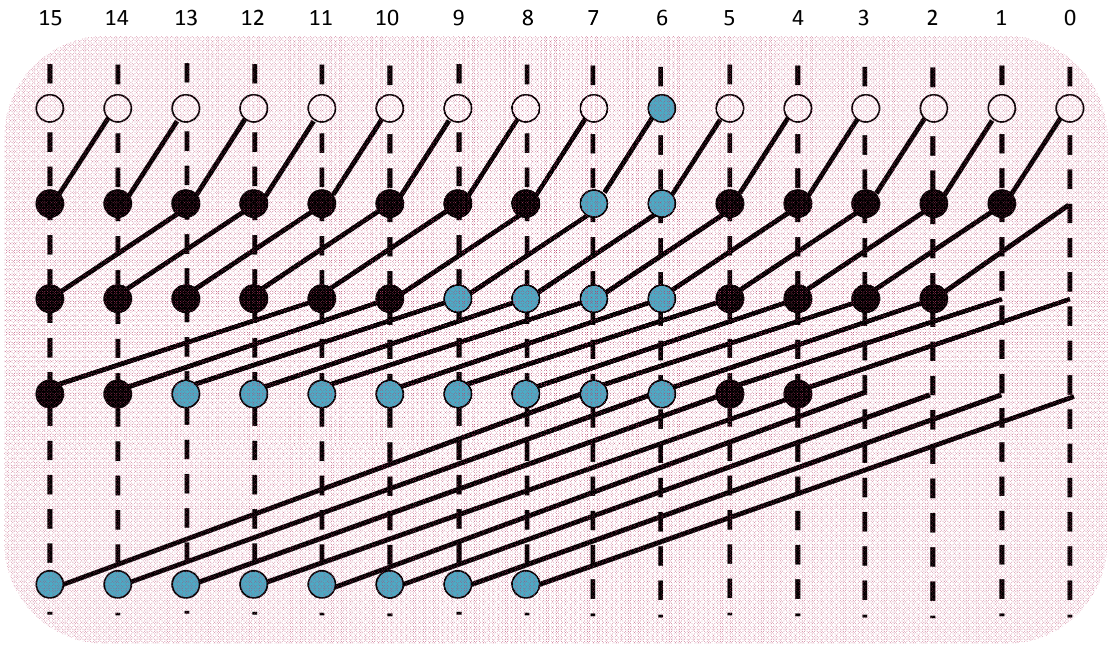


MULTISPECULATIVE FUNCTIONAL UNITS

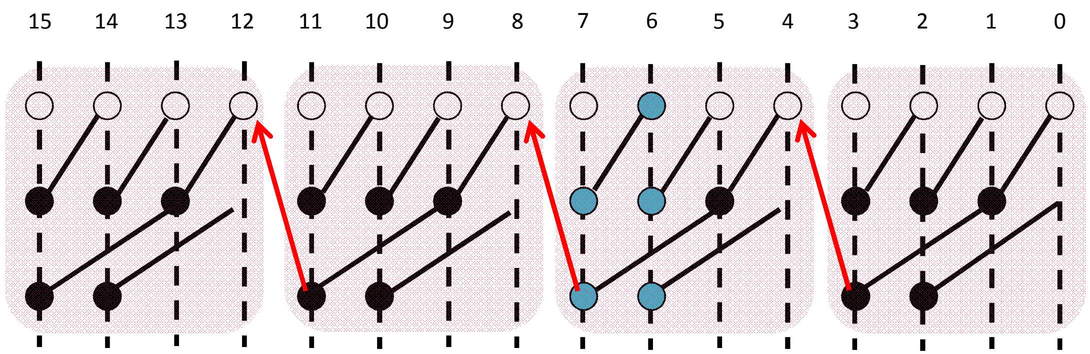
- Same interface: hit signal
- Distanced carries are quasi-independent [Nowick 1996, Lu 2004, Verma et al. 2008]
  - If the fragment size, *k*, is large enough, the probability of propagating a misprediction is close to 0
  - **Corollary. 2 very short cycles are enough to execute most of additions**
- Gains in execution time, area and energy
- Increase in the number of mispredictions

MULTISPECULATIVE FUNCTIONAL UNITS

- ***n*-bit Kogge-Stone Adder**
  - Complex carry propagation tree
  - Very fast
    - O(log(*n*))
  - Huge area
    - O(*n**log(*n*)) with large *n*
  - High switching activity



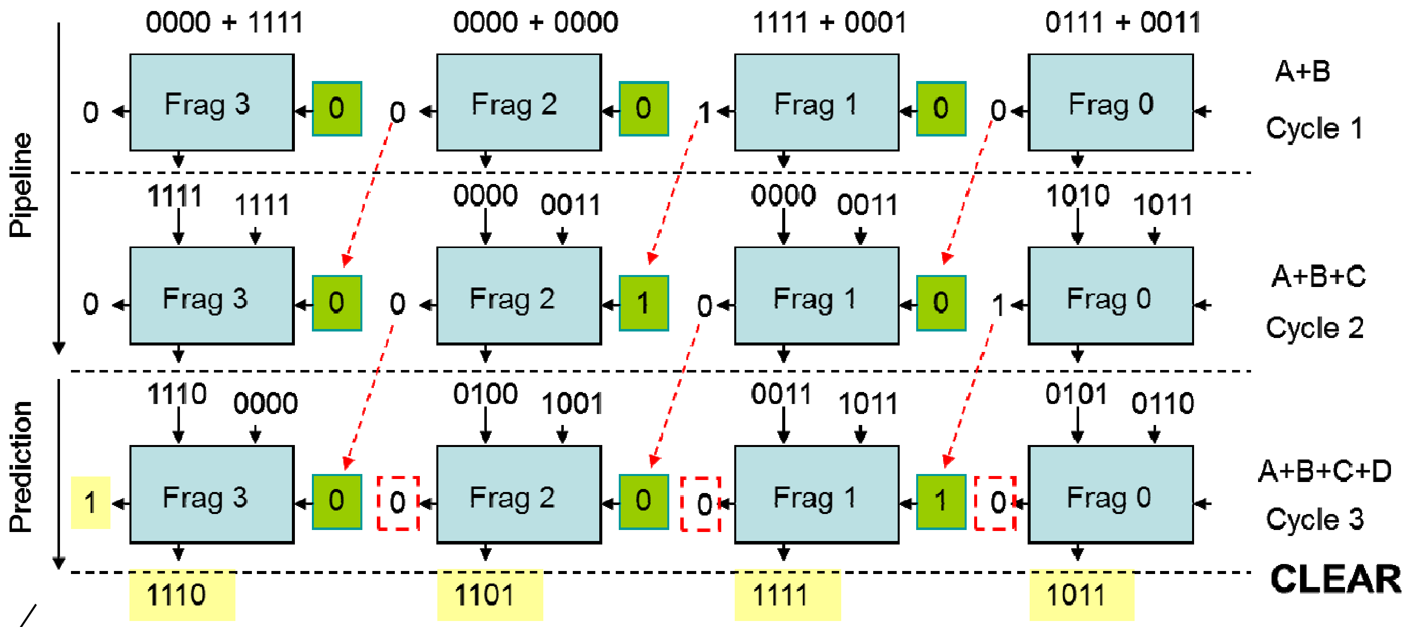
- ***n*-bit Multispeculative KS**
  - *n*/*k* simpler carry propagation trees
  - Extremely fast
    - O(log(*k*))
    - Predictors accuracy
  - Reduced area
    - Small KS have area O(*n*)
    - Area: *n*/*k**O(*k*) ≅ O(*n*)
  - Low switching activity

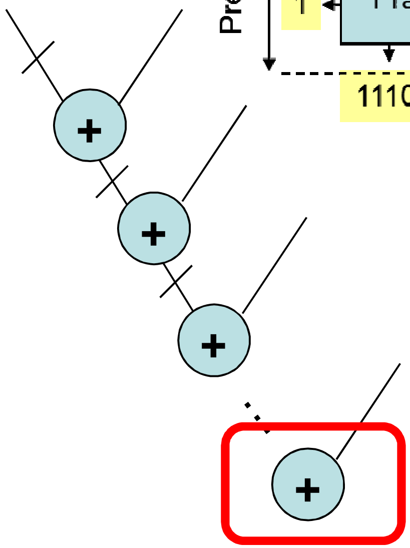


OUTLINE

- Introduction
  - Speculative Functional Units
- Multispeculative Functional Units
- **Multispeculative datapaths**
  - **Addition Chains**
  - Binary Addition Trees
  - Generic Additive Trees
- Some results
- Conclusions and future lines of work

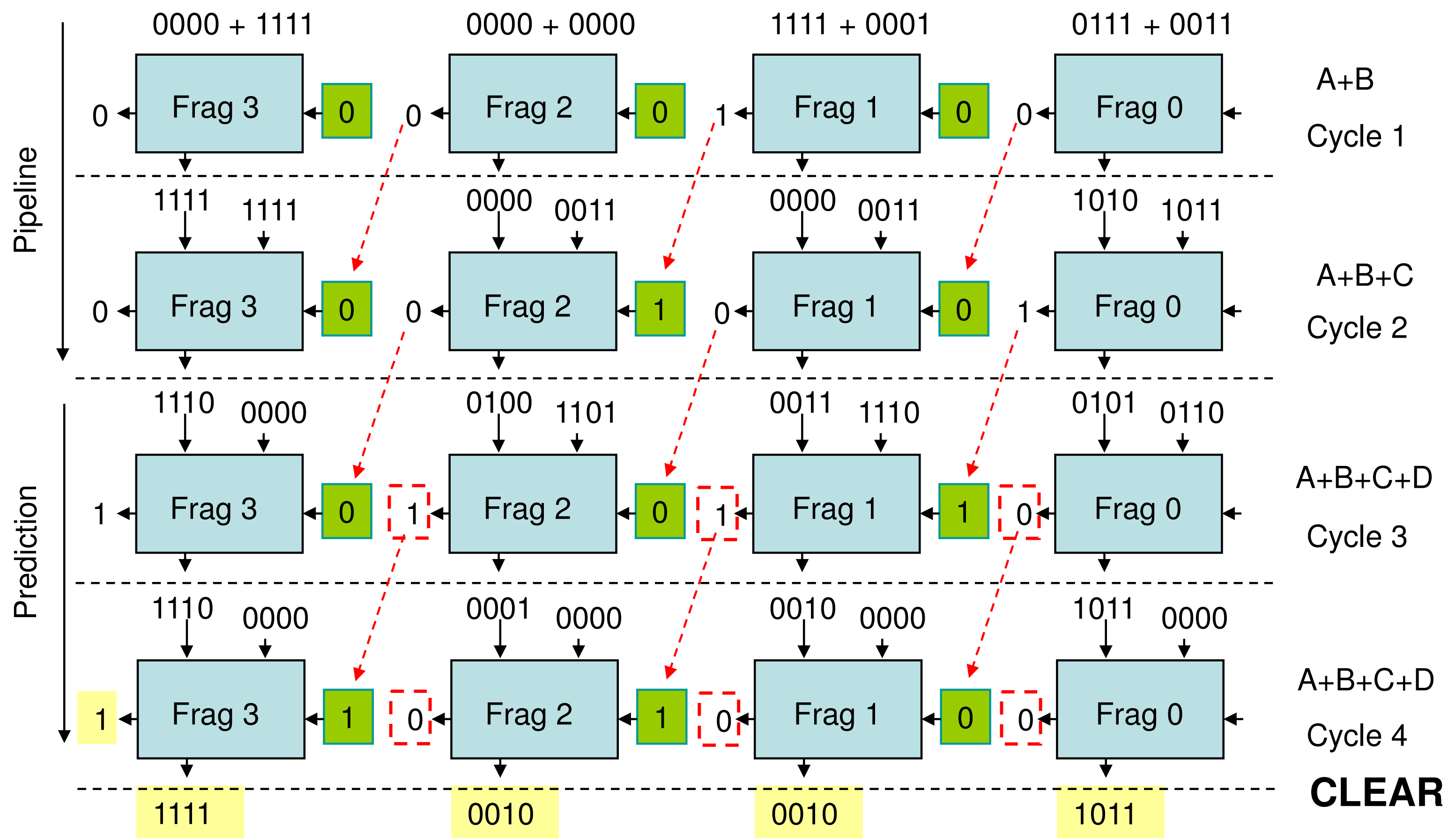
DATAPATH OPTIMIZATIONS: ADDITION CHAINS





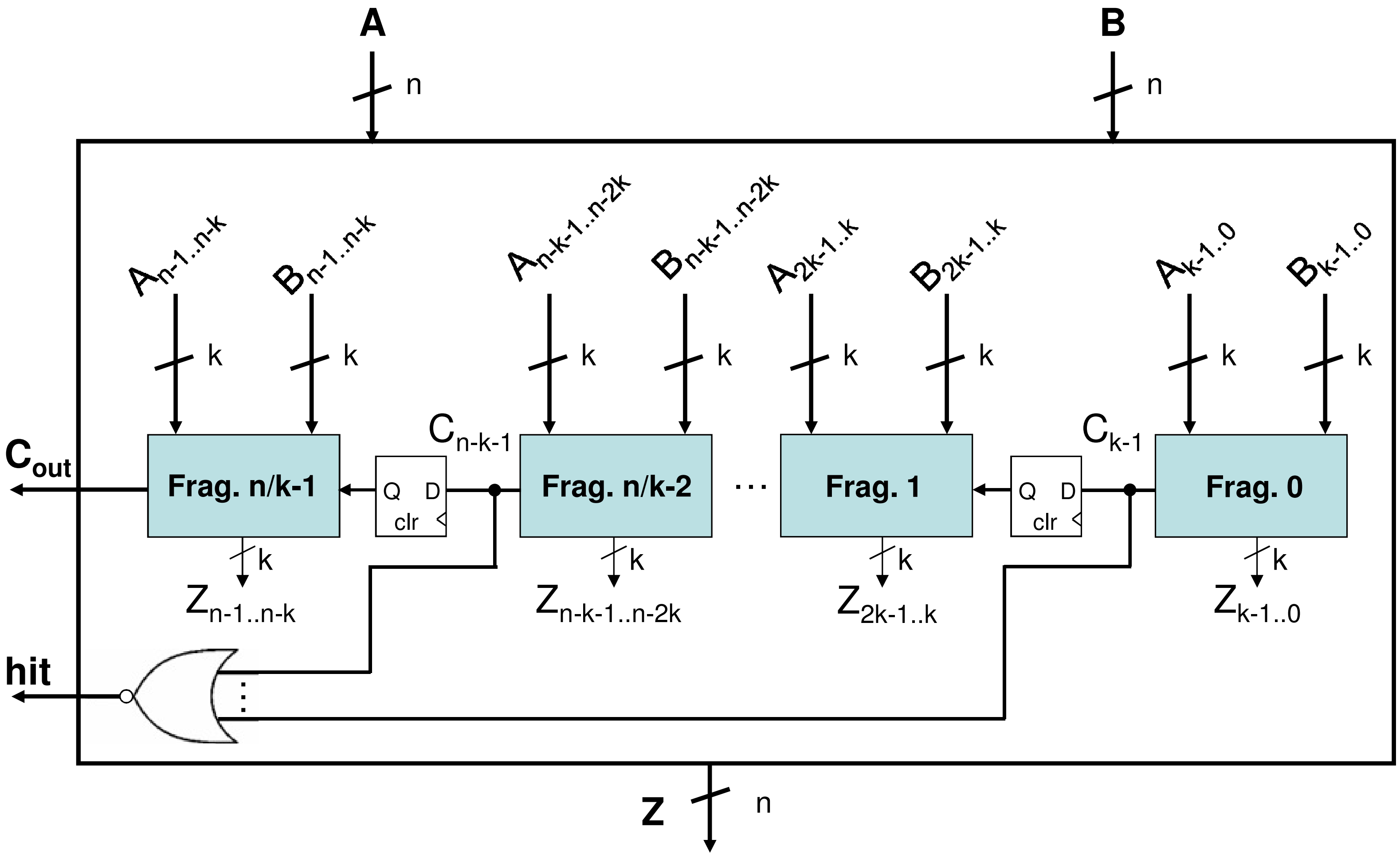
It is only necessary to apply prediction for the last addition ➔ Only the last addition can mispredict

MULTISPECULATIVE DATAPATHS: ADDITION CHAINS



Misprediction ➔ 2 cycles will be enough for the last addition, in most of the cases

MULTISPECULATIVE FUNCTIONAL UNITS: A MSADD FOR DATAPATHS

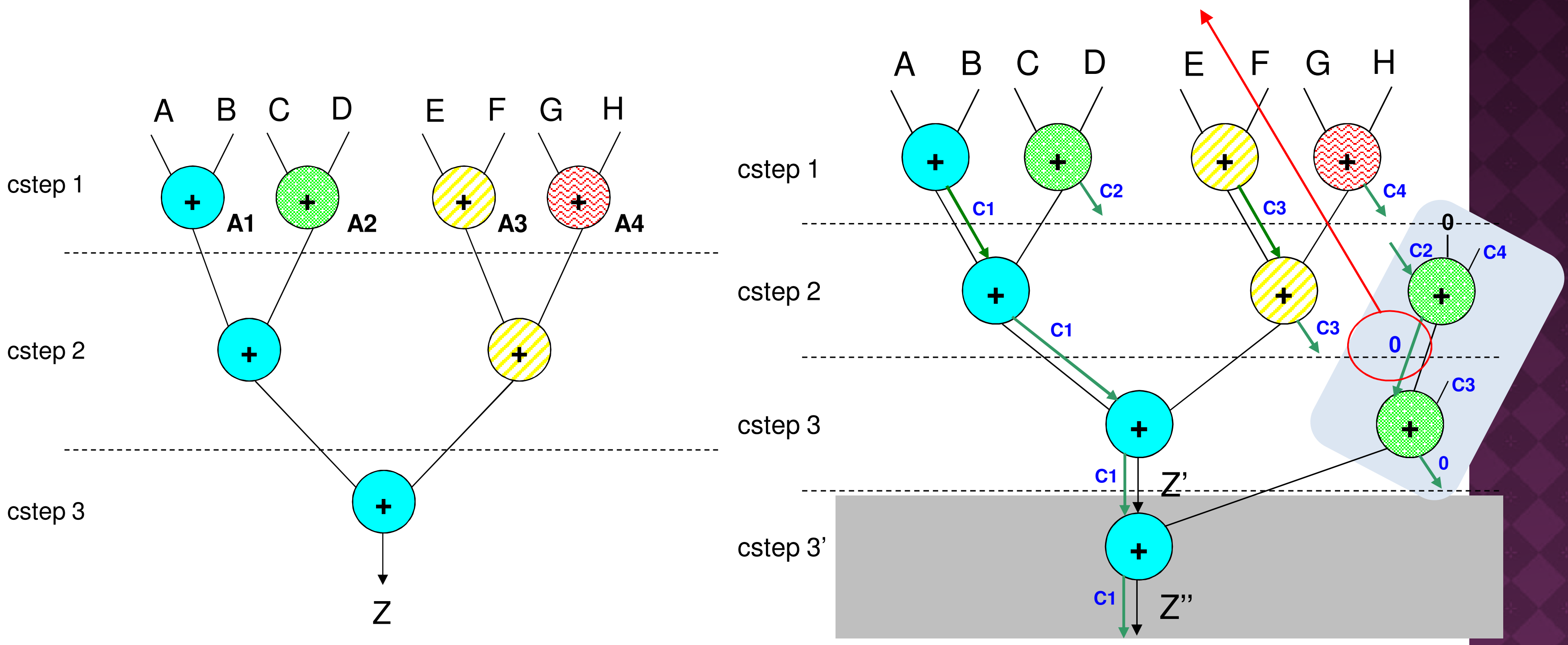


OUTLINE

- Introduction
  - Speculative Functional Units
- Multispeculative Functional Units
- **Multispeculative datapaths**
  - Addition Chains
  - **Binary Addition Trees**
  - Generic Additive Trees
- Some results
- Conclusions and future lines of work

MULTISPECULATIVE DATAPATHS: BINARY ADDITION TREES

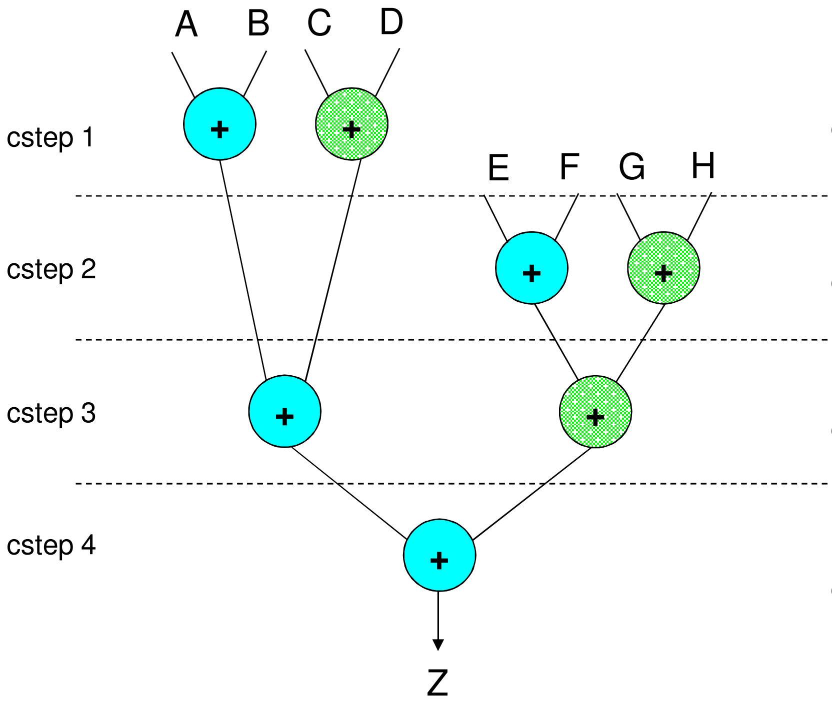
If $k > 1$, this carries array will become 0 for sure.



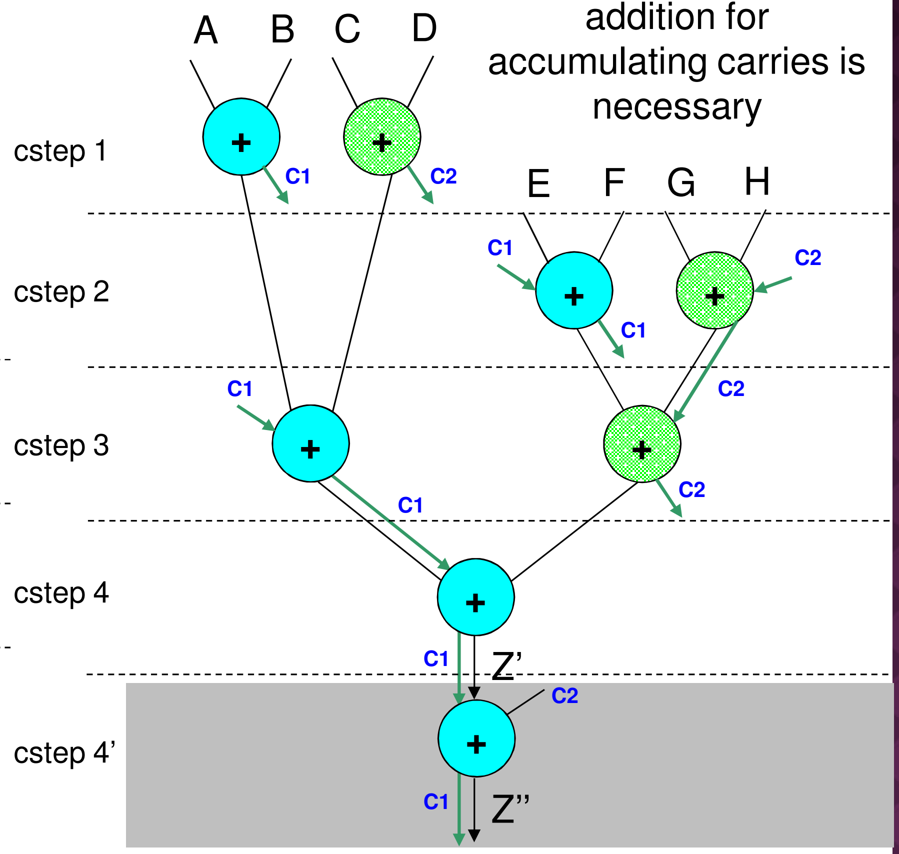
Z'=Z ➔ It can be possible, but the extra cycle will be unavoidable unless the result coming from A2 is 0

Z''=Z ➔ True with an extremely high probability

MULTISPECULATIVE DATAPATHS: BINARY ADDITION TREES



In this case no extra addition for accumulating carries is necessary



Z'=Z ➔ It can be possible, but the extra cycle will be unavoidable unless the result coming from A2 is 0

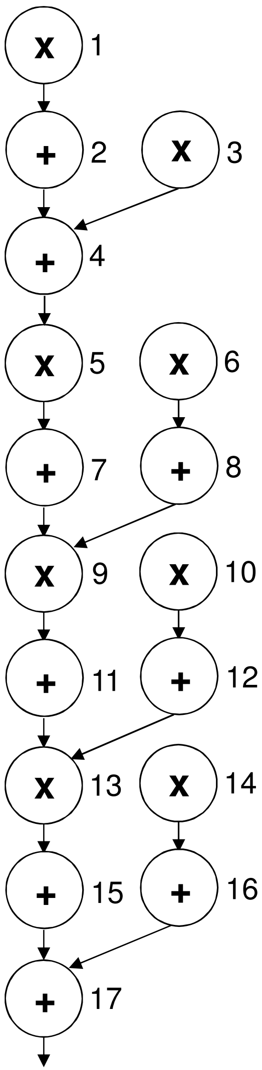
Z''=Z ➔ True with an extremely high probability

OUTLINE

- Introduction
  - Speculative Functional Units
- Multispeculative Functional Units
- **Multispeculative datapaths**
  - Addition Chains
  - Binary Addition Trees
  - **Generic Additive Trees**
- Some results
- Conclusions and future lines of work

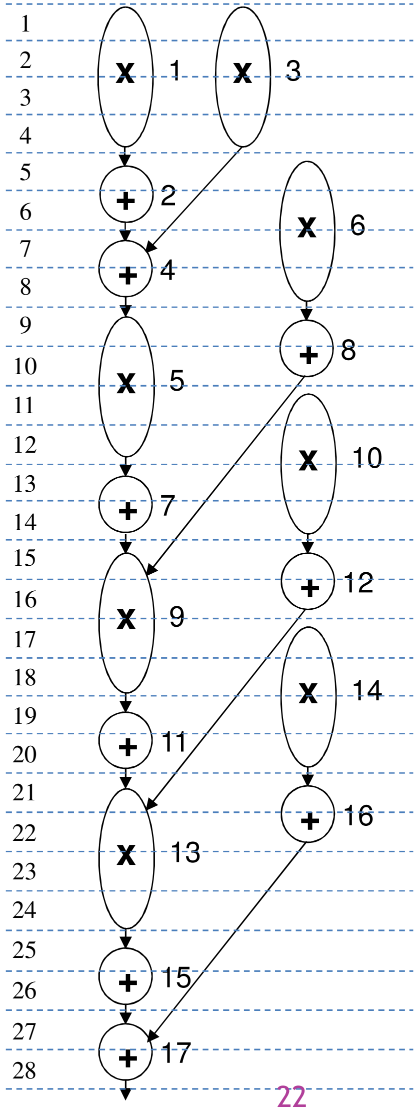
MULTISPECULATIVE DATAPATHS: GENERIC ADDITIVE TREES

Discrete Wavelet Transform



L(+): 2 cycles
L(*): 4 cycles

Overall latency: 28 cycles

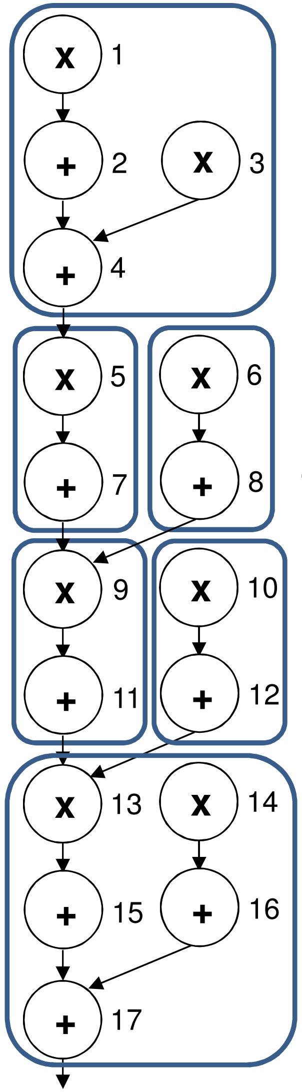


MULTISPECULATIVE DATAPATHS: GENERIC ADDITIVE TREES

- Algorithm
  - Step 1: Identify the additive trees
    - Additive trees can include products in the leaf nodes
  - Step 2: Introduce a recovery addition per tree
  - Step 3: Combined scheduling and binding
    - Resource constrained
    - Free MSFUs
      - Finished operation
      - Evaluate if carries have been consumed
      - Evaluate if scheduling/binding an operation can block the algorithm

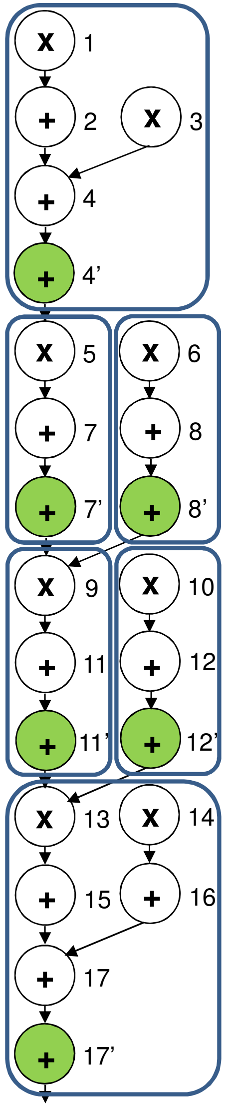
MULTISPECULATIVE DATAPATHS: GENERIC ADDITIVE TREES

Additive Tree Detection



Step 1

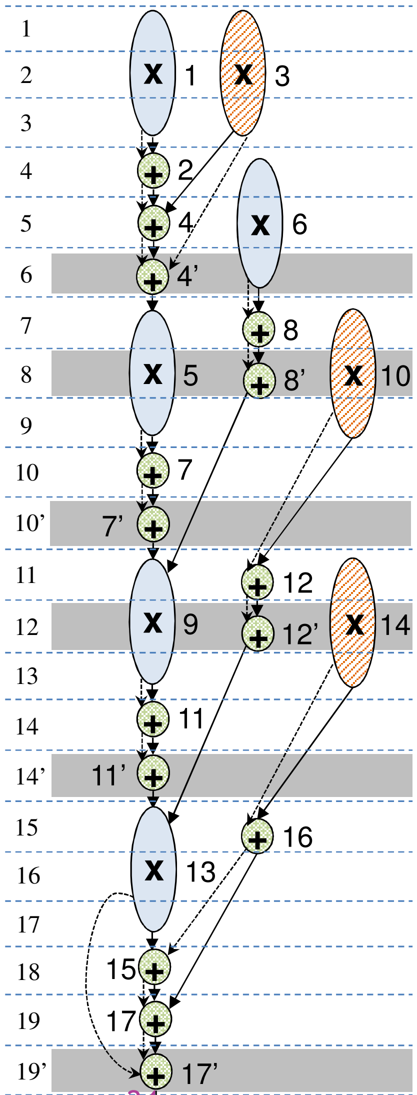
Recovery Additions introduction, 1 per tree



Step 2

L(+): 1 cycle
L(*): 3 cycles

Avg. latency: 19-22 cycles

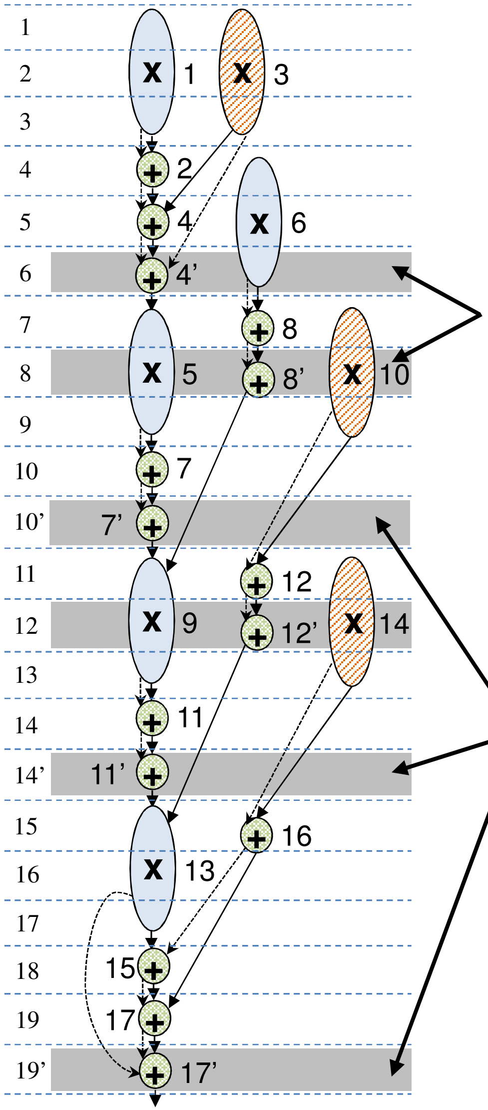


Step 3

MULTISPECULATIVE DATAPATHS: GENERIC ADDITIVE TREES

L(+): 1 cycle
L(*): 3 cycles

Overall latency: 19-22 cycles



Always executed: provides a recovery cstep/slot for operations 4 and 8, if necessary

Repeated iff operations 4' or 8' do not hit, respectively

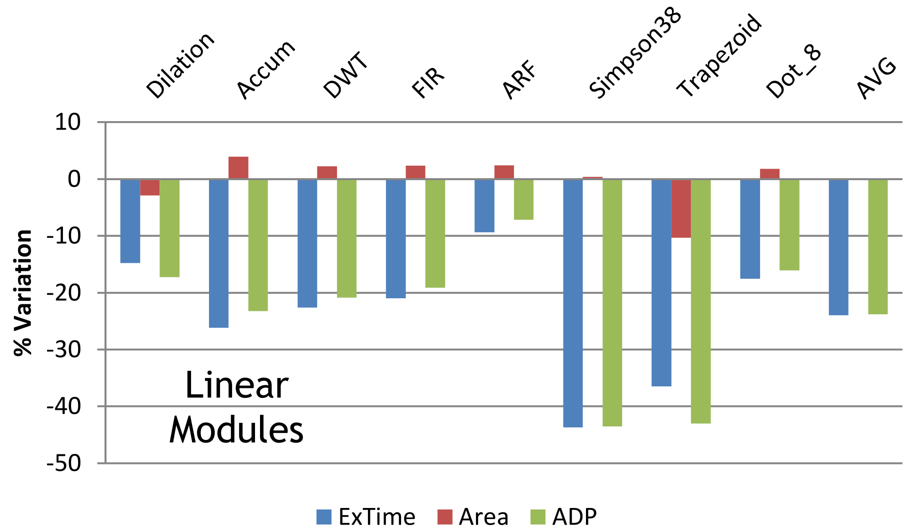
Not necessary iff operations 7, 11 and 17 hit, respectively

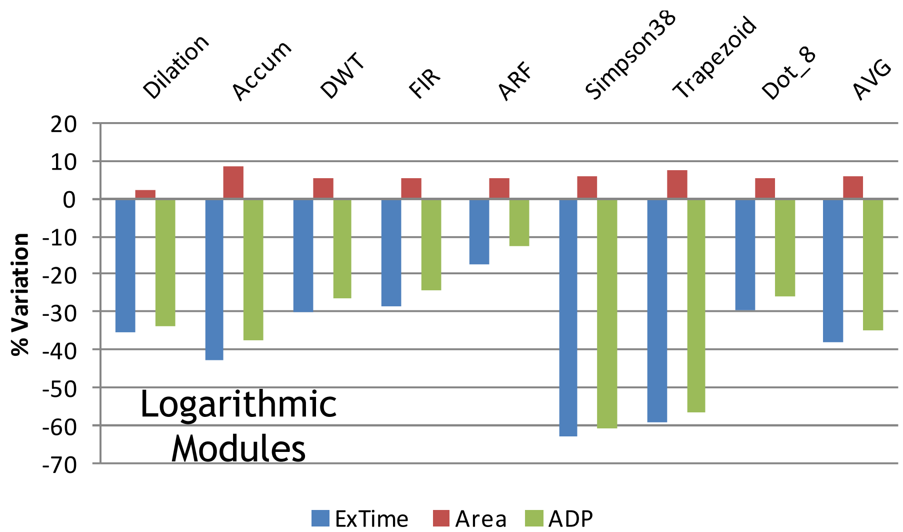
Repeated as many times as needed (expected: 0 or 1 cycles)

OUTLINE

- Introduction
  - Speculative Functional Units
- Multispeculative Functional Units
- Multispeculative datapaths
  - Addition Chains
  - Binary Addition Trees
  - Generic Additive Trees
- **Some results**
- Conclusions and future lines of work

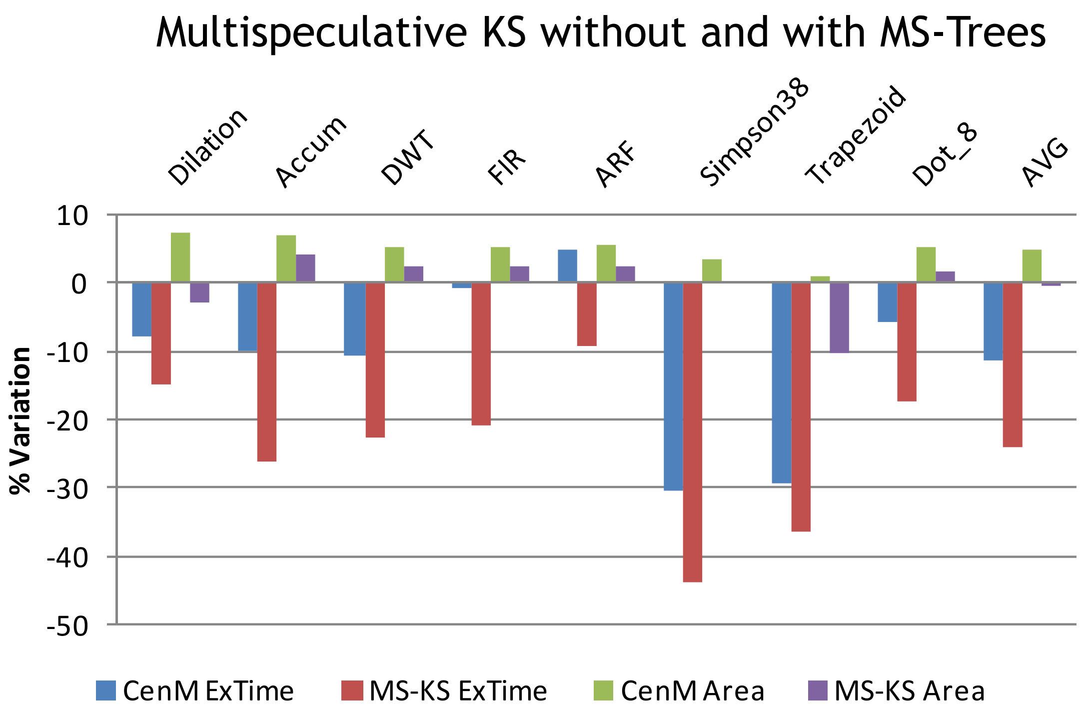
EXPERIMENTAL RESULTS: MSTREES VS BASELINE





- **Logarithmic modules: KS-based**
  - Less ExTime reduction
  - Negligible area increase
- **Linear modules: RCA-based**
  - Slight Increase in area
    - Splitting a RCA does not reduce its area
  - Greater ExTime reduction
    - RCA carry chain is not optimized
- **Best results with larger bitwidths**
  - 32-bits: Simpson38, Trapezoid
  - 16-bits: the rest

EXPERIMENTAL RESULTS: WITH OR WITHOUT MSTREES



- **Advantages of MS-Tree Management**
  - Greater ExTime Reduction
  - Lower Area Penalty

OUTLINE

- Introduction
  - Speculative Functional Units
- Multispeculative Functional Units
- Multispeculative datapaths
  - Addition Chains
  - Binary Addition Trees
  - Generic Additive Trees
- Some results
- **Conclusions and future lines of work**

CONCLUSIONS AND FUTURE LINES OF WORK

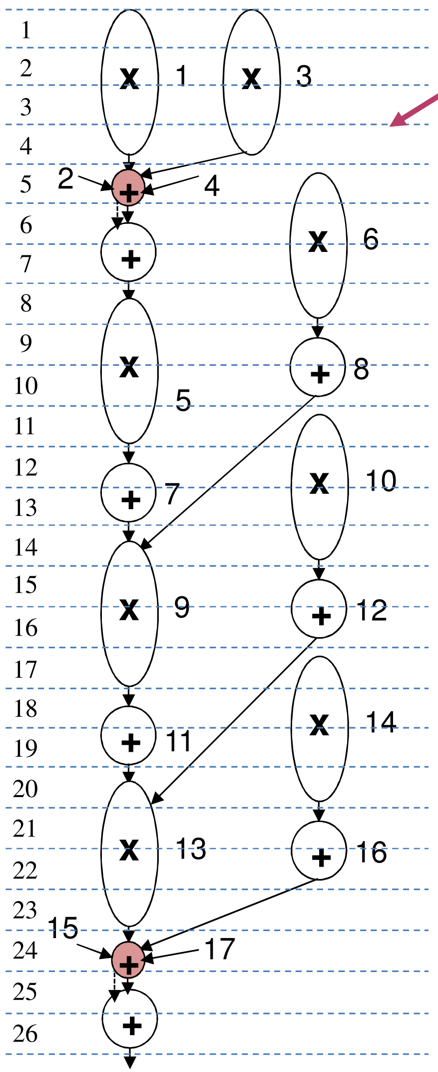
- (Multi)Speculative FUs are efficient
- We propose strategies for utilizing these efficient (M)SFUs in the Design Automation context
  - Distributed Management
  - MSTrees Management
- More applications
  - MSFUs behave better with large bitwidths
    - Design of Floating Point Units
- Next step
  - Integration with Distributed Management
  - Integrate CSA and MS-Trees

THANK YOU FOR YOUR ATTENTION !!!

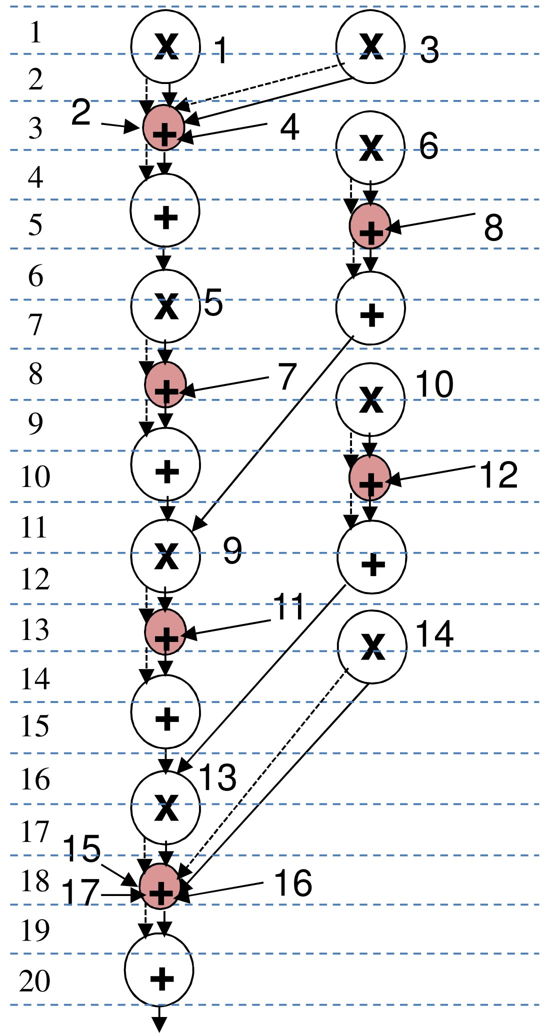
And remember … The important thing is not to stop questioning; curiosity has its own reason for existing (*Einstein*)

You can em@il me to:
abarriog@ucm.es

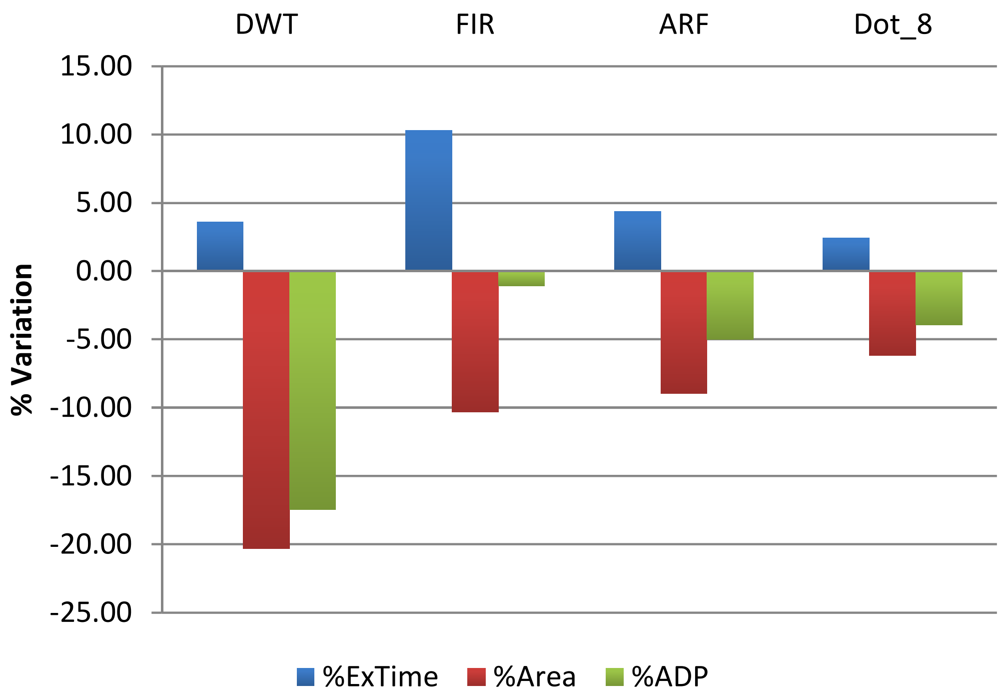
MULTISPECULATIVE DATAPATHS: GENERIC ADDITIVE TREES



- Moderate CSA
  - Latencies
    - L(*): 4 cycles
    - L(+, CPA): 2 cycles
    - L(+, CSA): 1 cycle
  - Limited performance
- Extreme CSA
  - Latencies
    - L(*, CSA): 2 cycles
    - L(+, CPA): 2 cycles
    - L(+, CSA): 1 cycle
  - Increase in area
    - CSAs. CPAs are still necessary
    - Routing and registers. CSAs produce 2 bit-vectors
  - Low performance difference
  - Limitation imposed by CPAs still exists
    - In our flow, a CPA is substituted by a MSADD+recovery addition
    - Solution: Integration with Distributed controller

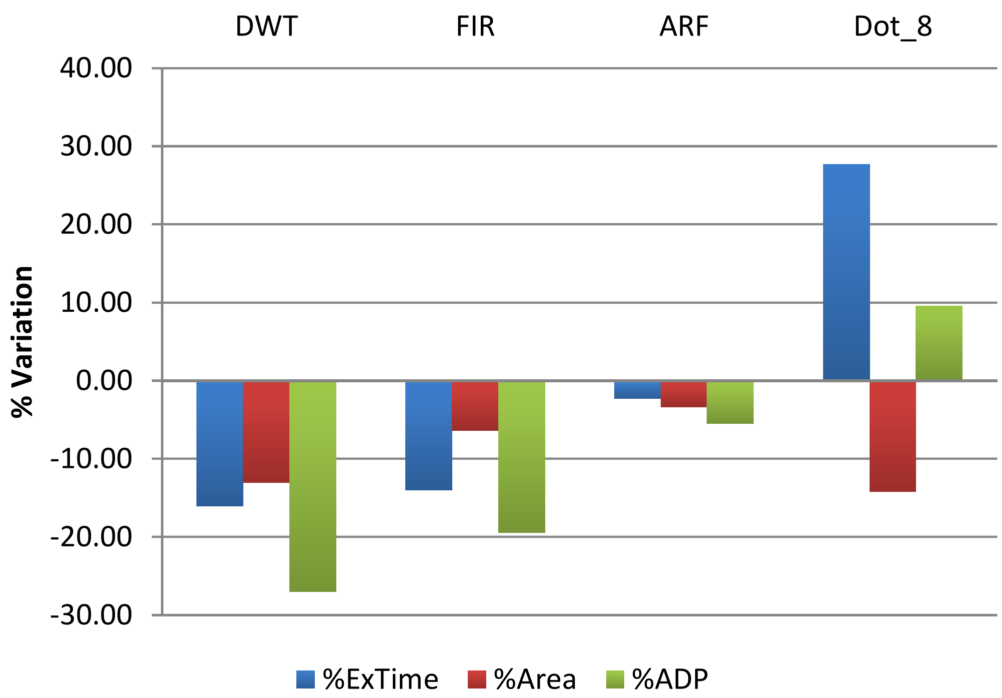


EXPERIMENTAL RESULTS: MSTREES VS EXTREME CSA



- **MSTrees vs Extreme CSA (16-bits)**
  - Slight performance difference
  - Less area
  - Overall: better Area Delay Product

EXPERIMENTAL RESULTS: MSTREES VS EXTREME CSA



- **MSTrees vs Extreme CSA (32-bits)**
  - ExTime reduction (CPAs greater penalty)
  - Less area reduction (Multipliers weight)
  - Overall: better Area Delay Product