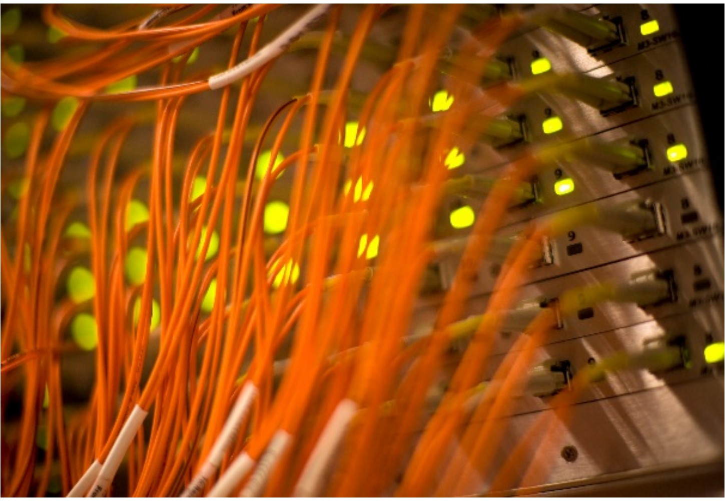
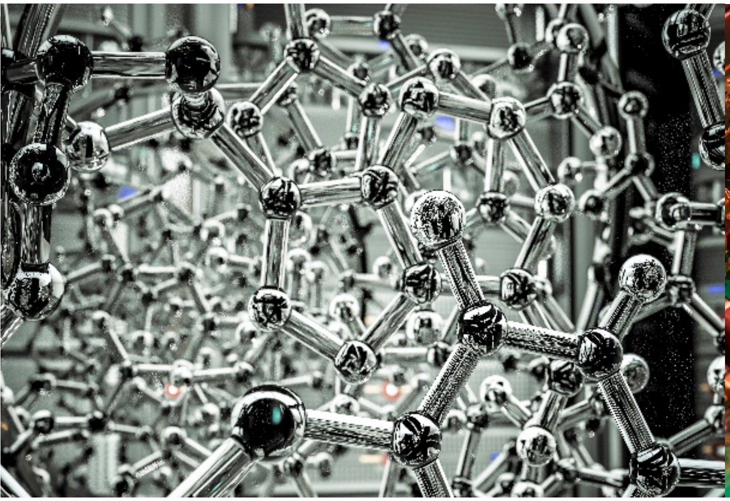
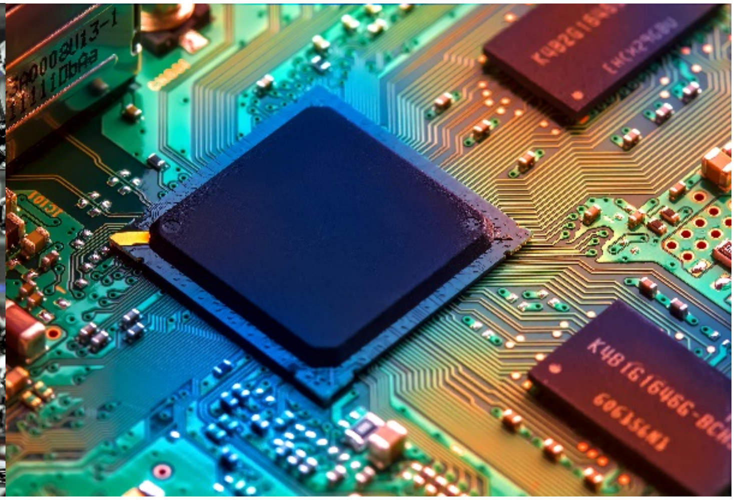
Cost-Effective Error Detection through Mersenne Modulo Shadow Datapaths

Keith Campbell, Chen-Hsuan Lin, Deming Chen
University of Illinois at Urbana-Champaign







ECE ILLINOIS

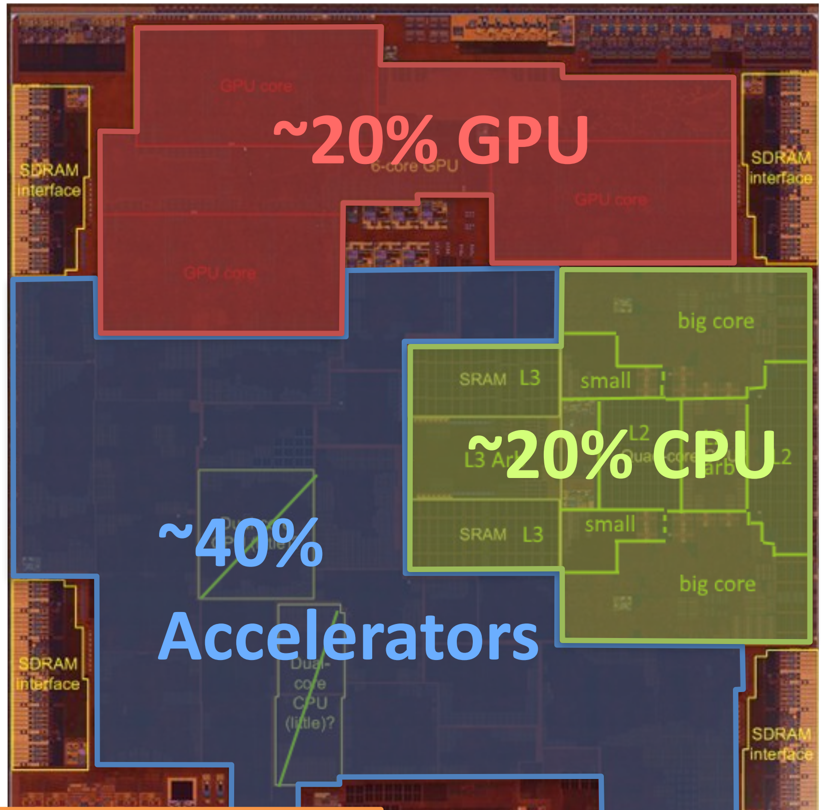
ILLINOIS

Motivation

Computation Efficiency

- Problems
  - Demand for more **computation density**
  - Demand for more **performance per watt**
- Solutions
  - Scale transistors **End of Dennard Scaling**
  - Use custom hardware
    - Accelerators in SoCs
    - FPGAs in datacenters
  - 3D Stacking

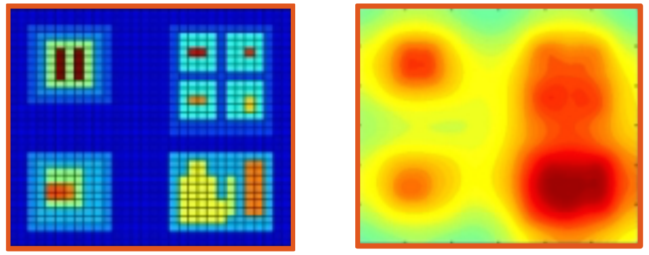
Apple A10 Fusion SoC (2016)



Xilinx's to Provide Amazon with its latest FPGA

Hardware Reliability

- Problems
  - Transistor wear-out
  - Soft errors
  - Timing errors
  - Electromigration



Power

Thermal

Hot Spots

- Solutions

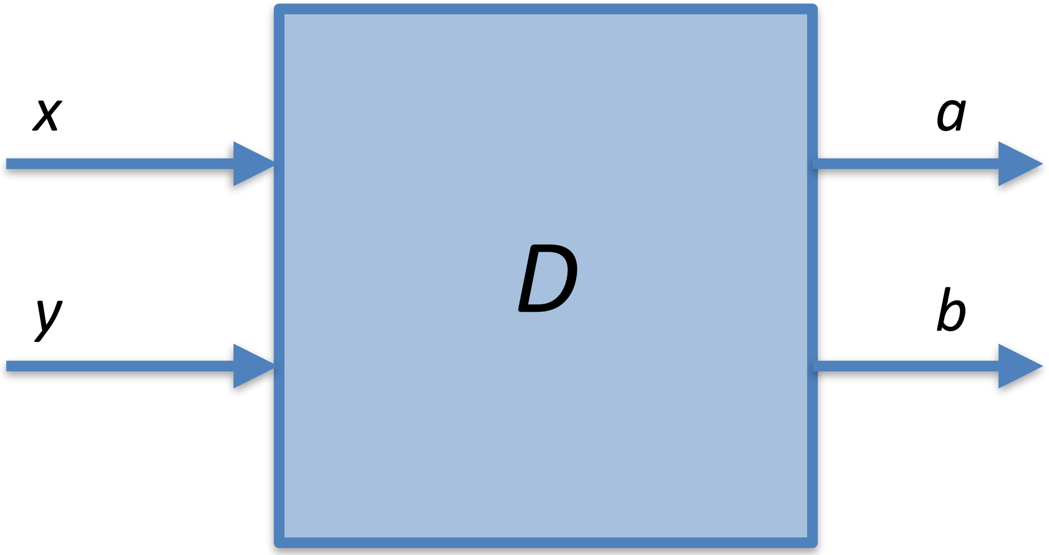
Modular / time redundancy	Has 2–3× cost.
Razor logic	Adds timing constraints, limits fault types.
Flip-flop hardening	Does not protect combinational logic.
Parity	Expensive across computation.
Algorithm-Based Fault Tolerance (ABFT)	Adds memory accesses, limited to matrices.

Our Approach: Modulo Shadow Datapaths

- Small (2-8 bit) modulo 2^{n-1} checksums
  - Like ECC, but for *computation*
- **Minimum assumptions** about fault behavior
  - Protect against bits flips anytime, anywhere
- Focus on complex datapaths
- **Maximum cost-effectiveness**
  - Rethink gate-level architecture
  - 6–10% area cost for 3–61× reliability benefit
  - 15–20% area cost for 121–2,477× reliability benefit

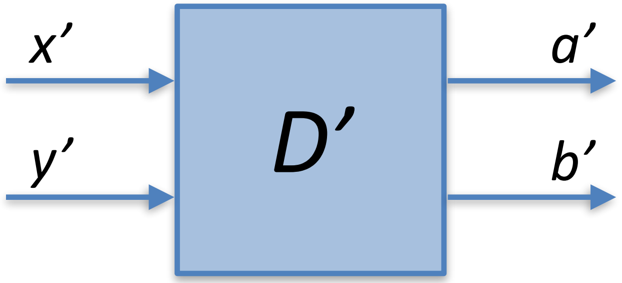
What is a shadow datapath?

Given a datapath *D*, we define a shadow datapath *D'* as a redundant datapath that performs a similar computation as *D*, but with compressed versions of the inputs and outputs.



In math speak: *D'* does computation in a *homomorphism* of the algebra of *D.*

- A *reduction* function maps inputs and outputs to shadow inputs and outputs.
- Shadow outputs are computed redundantly, enabling output checking

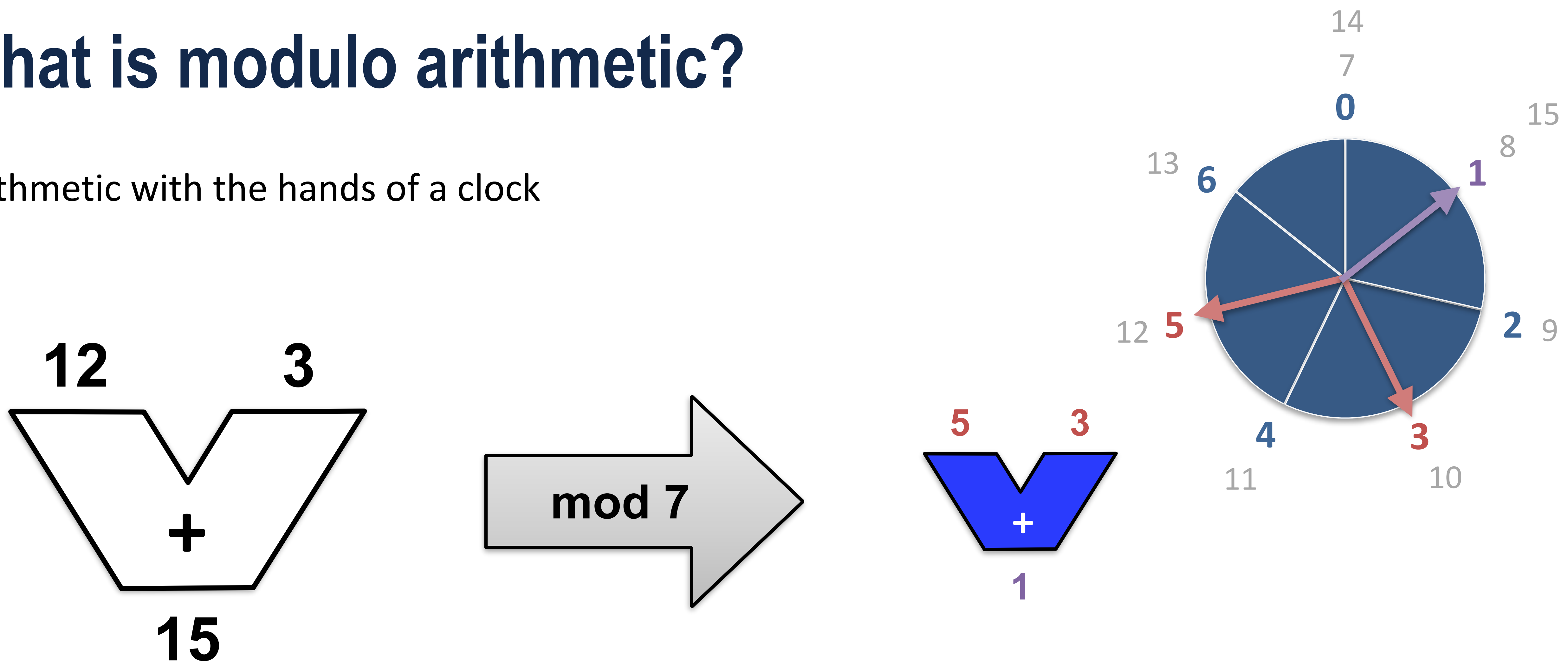


x' **← reduce(*x*)** ***a'*** **?= reduce(*a*)**

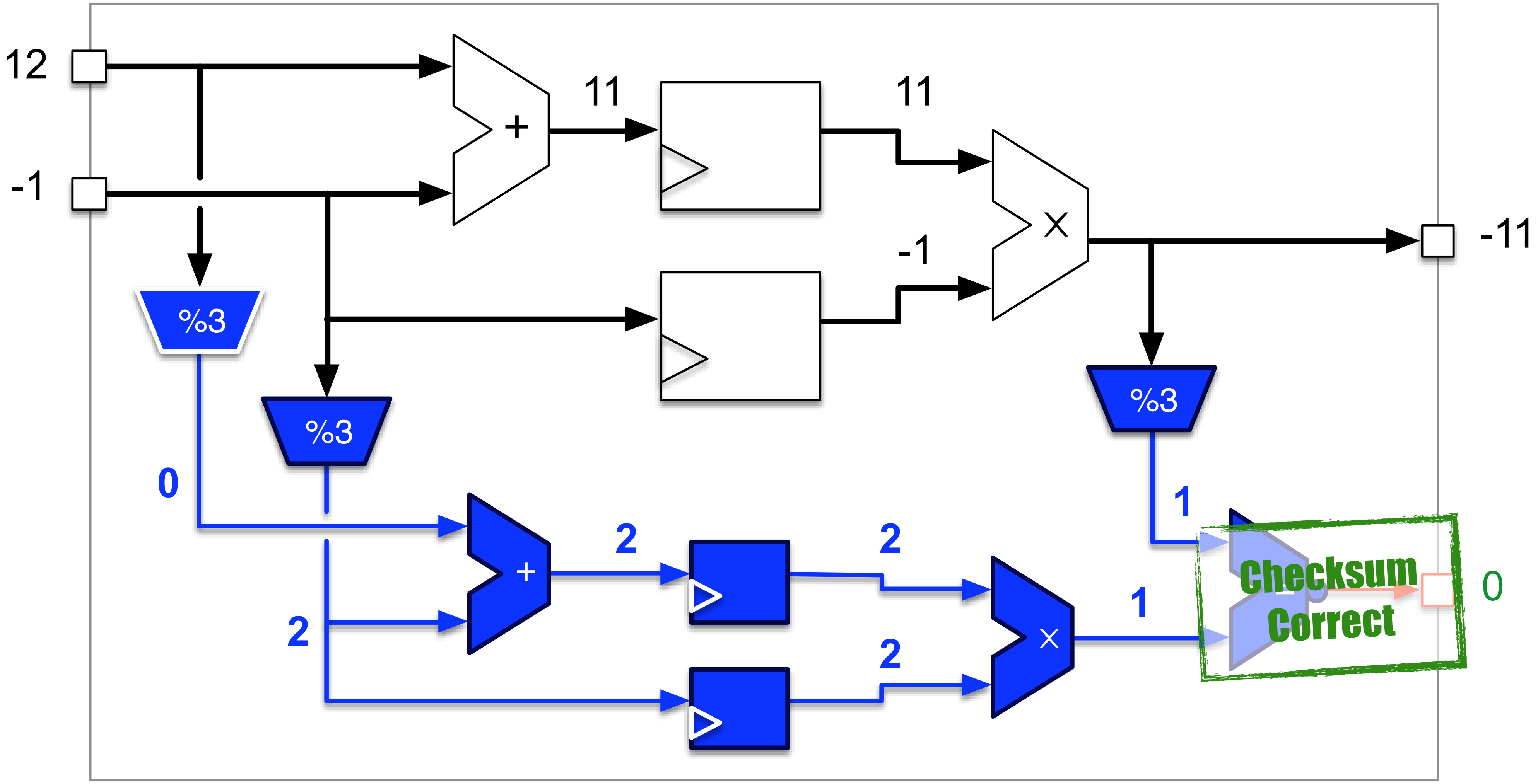
y' **← reduce(*y*)** ***b'*** **?= reduce(*b*)**

What is modulo arithmetic?

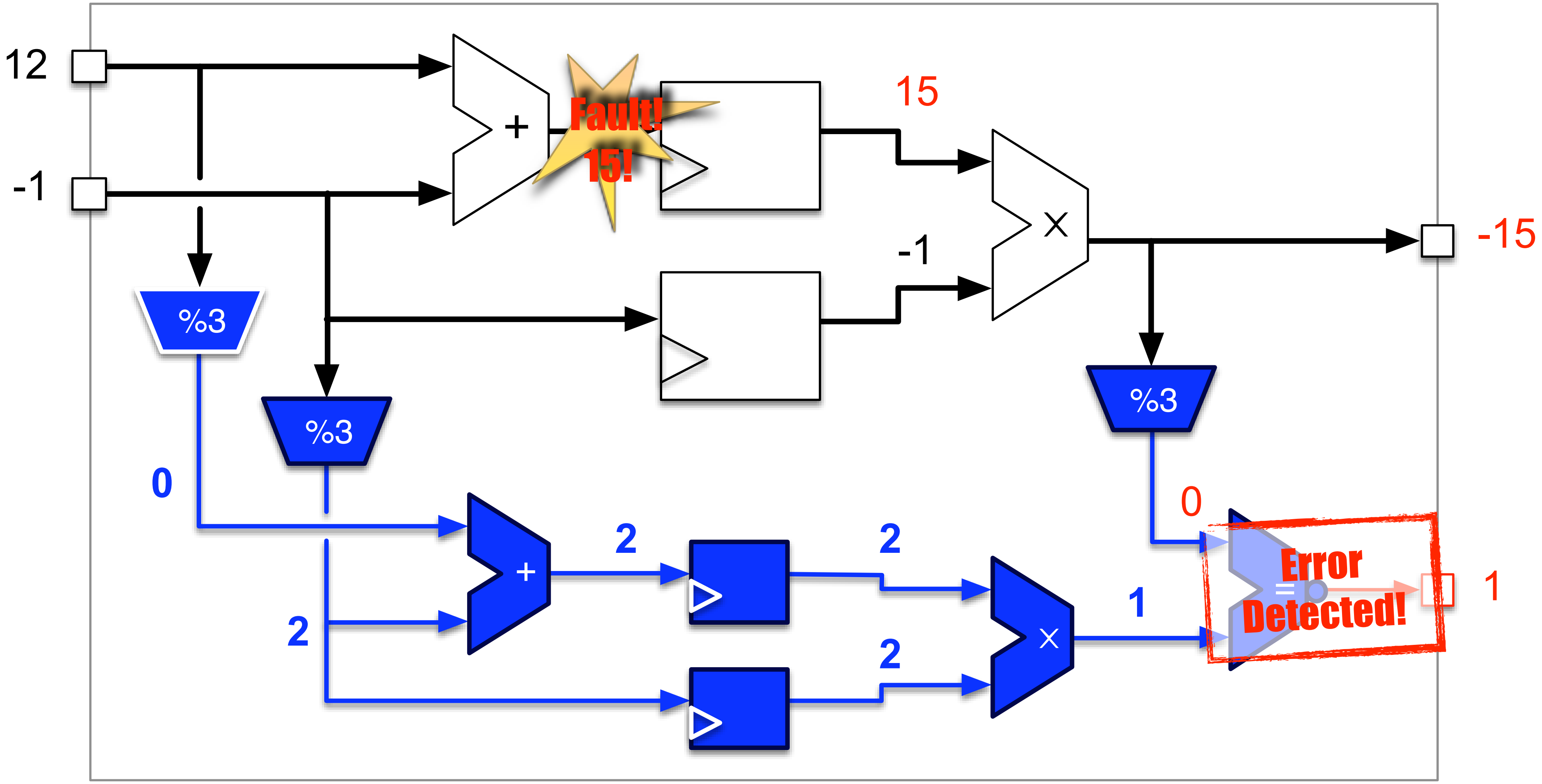
Arithmetic with the hands of a clock



How do modulo shadow datapaths work?



How do modulo shadow datapaths work?



Modulo Arithmetic: Existing Solutions

- Lookup-table based approach for mod-3 (DAC’15)
- Interleaved full adders and inverters for mod-3 reduction (Piestrak et al., EUSIPCO’98)
- Signed-digit architecture (Wei et. al., JSCS’03)
- Modulo exponentiation architectures for cryptography

Exponential scaling to wider bases

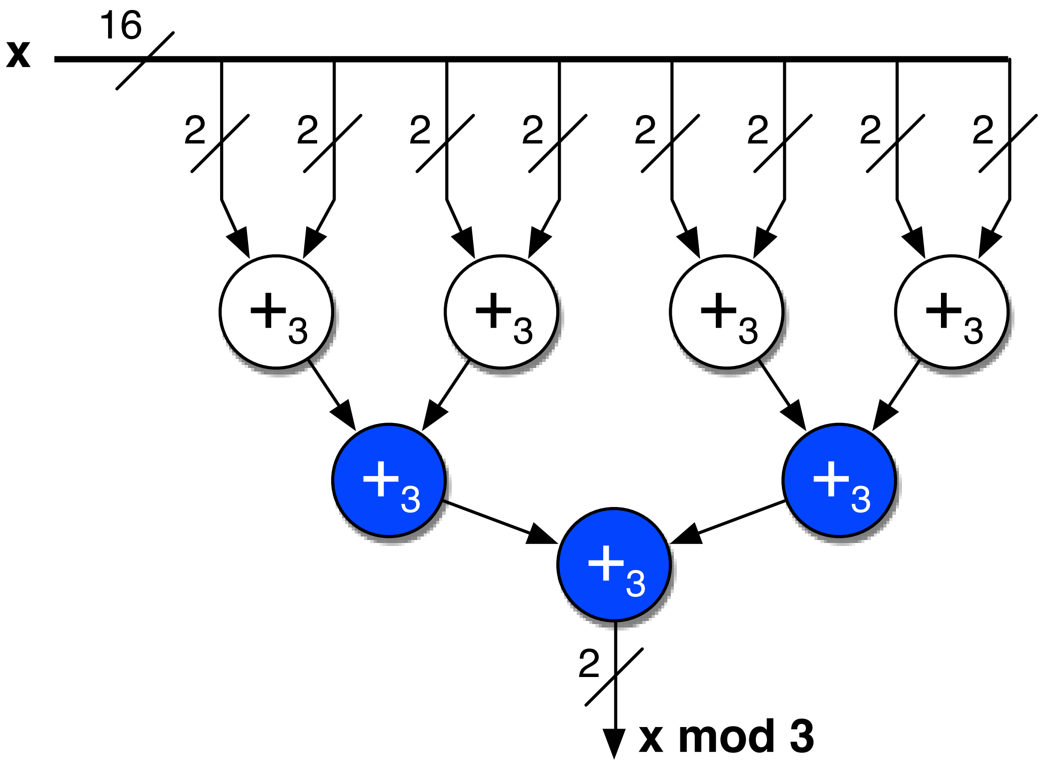
Trick only works for mod-3

Requires 2 bits per digit

Highly specialized algorithms

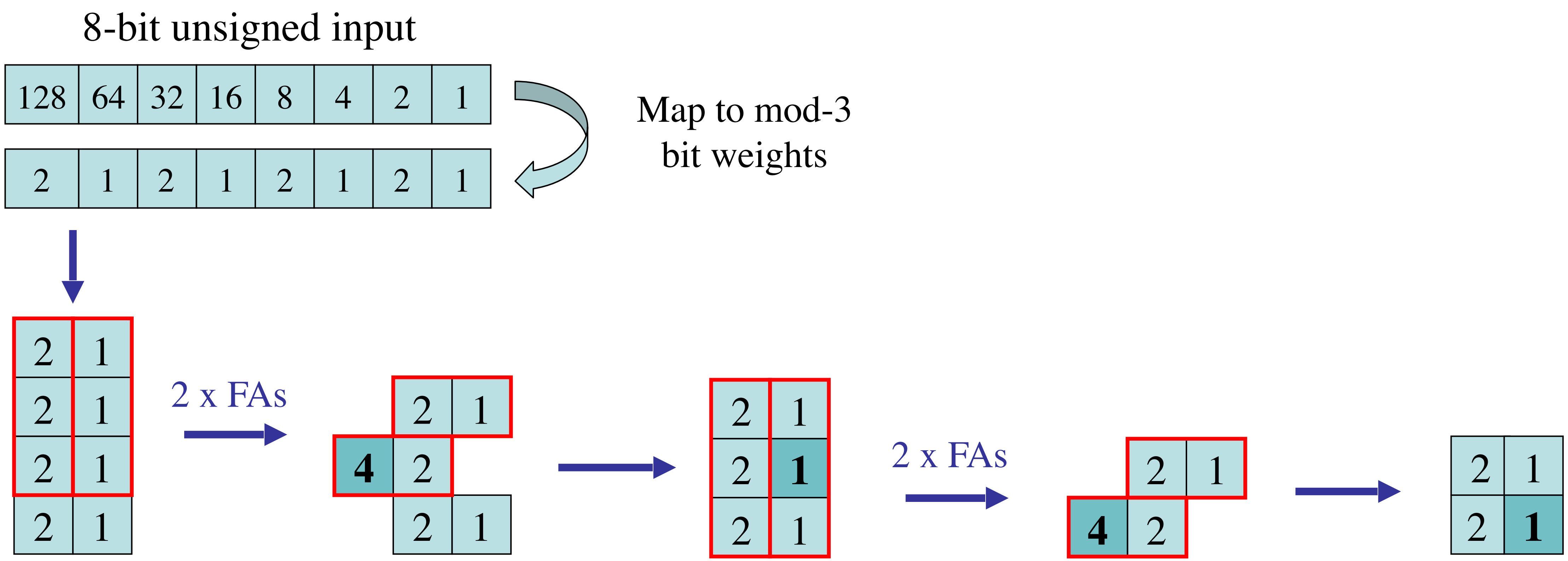
Mersenne Modulo Hardware Architectures

Previous Modulo Arithmetic Design (DAC'15)



- Cost is dominated by reducer functional units
- Traditional design is a tree of modulo adders
- Cost increases exponentially with larger bases
- Not a typical pattern for logic synthesis tools

New Wallace-Tree Like Reduction Strategy



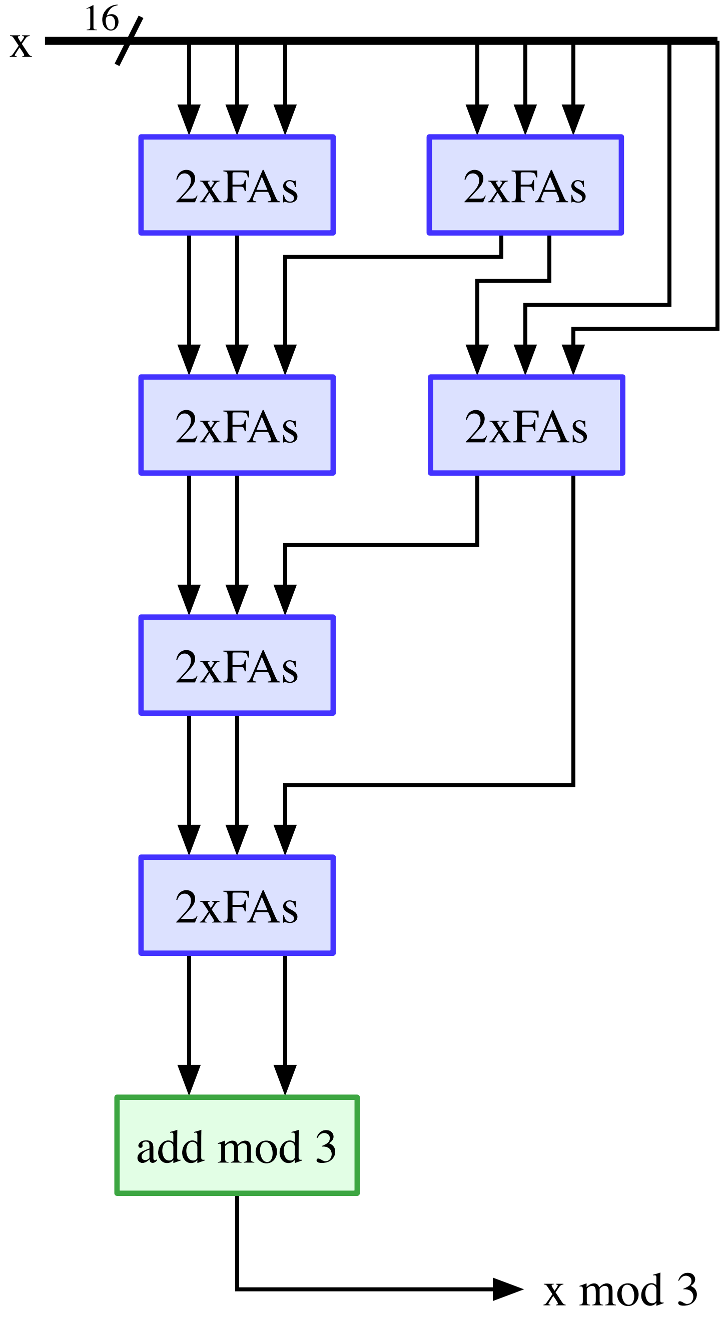
8-bit unsigned input

128	64	32	16	8	4	2	1
2	1	2	1	2	1	2	1

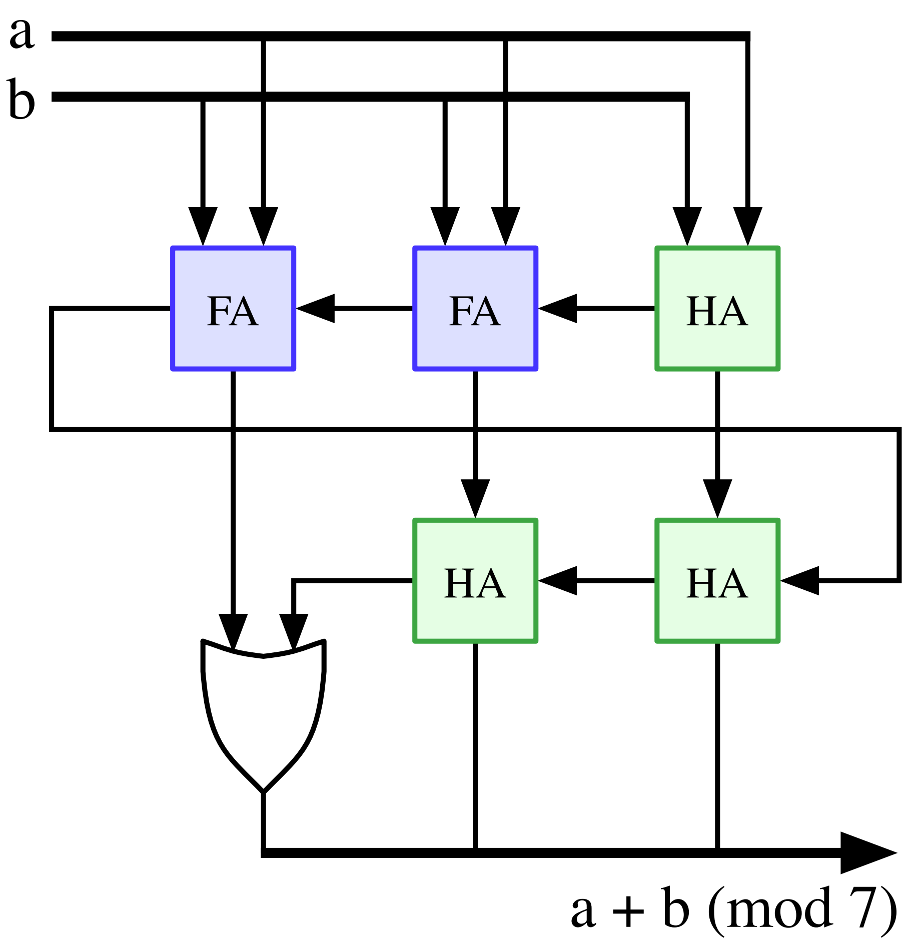
Map to mod-3 bit weights

New Reducer Design

- Almost all full adders (standard cells)
- Extends to any Mersenne number base (2^n-1)
  - mod-7, mod-15, mod-31, etc.
- Each full-adder reduces (eliminates) one bit
- Fixed area cost for given input width



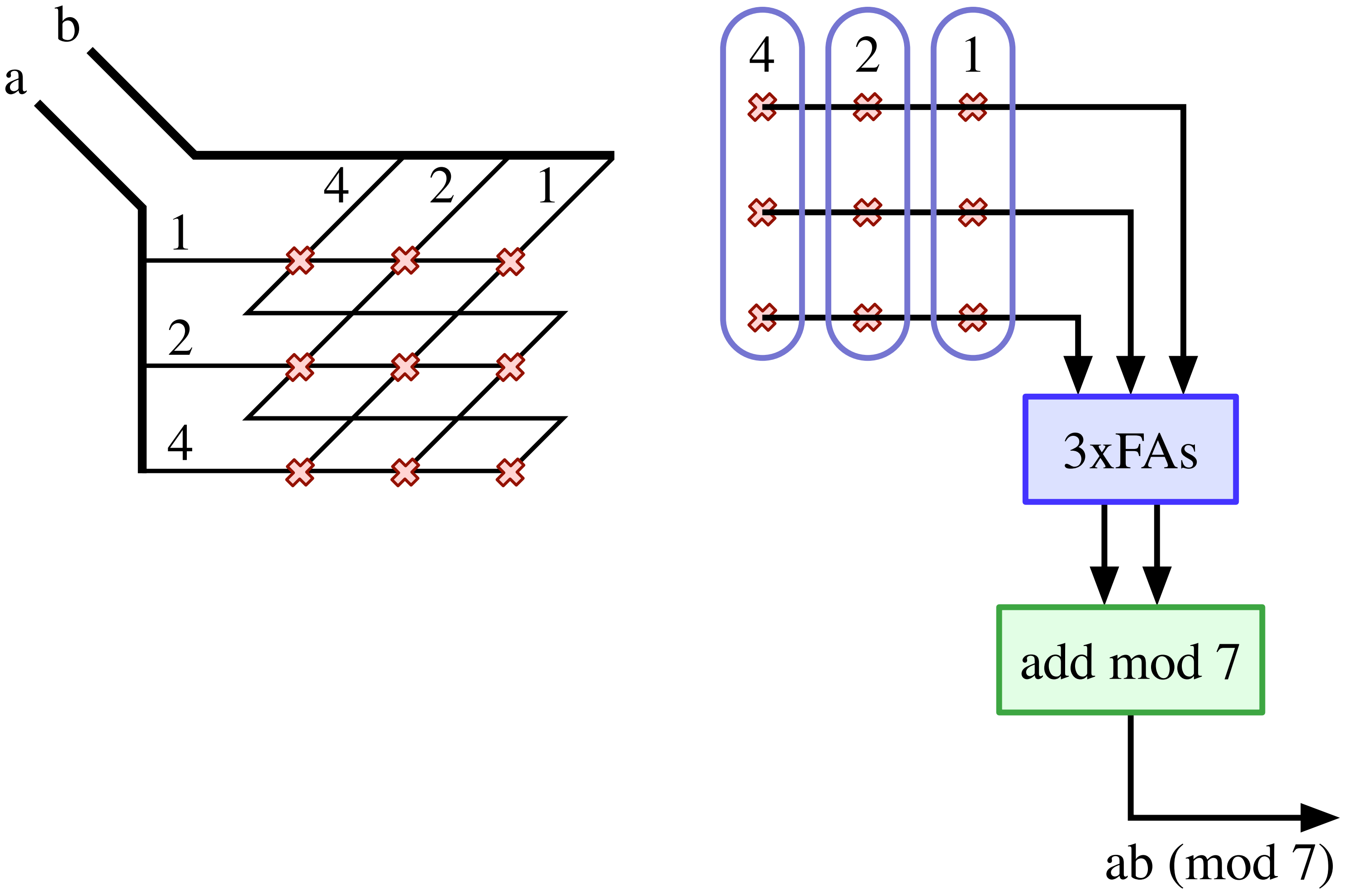
New Modulo Adder Design



- Ripple-carry adder with a “wraparound” twist
- 8 = 1 (mod 7)
- Carry circulation eventually stops
- Output not normalized
- Two representations for “0”: 000 and 111

New Modulo Multiplier Design

- Array multiplier with a "wraparound" twist
- Leverage reduction technique to reduce product bits



Cost-Effectiveness Results

Functional-Unit Level Cost (32-bit main datapath)

- Large reductions in reducer cost
- New designs scale well to larger Mersenne bases

	Functional Unit	DAC’15		Ours		Difference	
		Area	Delay	Area	Delay	Area	Delay
Mod-3 (2-bits)	add	8.30	0.09	12.76	0.13	53.7%	42.9%
	subtract	8.30	0.09	14.02	0.15	68.9%	60.4%
	multiply	4.47	0.04	17.84	0.16	299.3%	292.7%
	reduce	177.80	0.73	155.56	0.39	-12.5%	-47.1%
Mod-7 (3-bits)	add	55.86	0.32	21.06	0.21	-62.3%	-35.8%
	subtract	59.69	0.33	22.95	0.22	-61.6%	-32.7%
	multiply	30.01	0.21	47.79	0.30	59.2%	42.3%
	reduce	493.18	1.27	153.64	0.61	-68.8%	-52.0%
Mod-15 (4-bits)	add	188.01	0.46	29.33	0.27	-84.4%	-41.6%
	subtract	192.80	0.53	31.85	0.29	-83.5%	-46.0%
	multiply	133.43	0.51	90.45	0.42	-32.2%	-16.8%
	reduce	687.57	1.55	151.73	0.53	-77.9%	-66.1%

45nm ARM technology library   Area unit: μm^2 Delay unit: ns

Aside: Reliability Evaluation

- Reliability models not created equal
- Ad-hoc error injection
  - Error injection method does not model real faults.
- Assumptions about fault behavior
  - “Soft errors only affect flip-flops.”
- Specialized to a particular kind of fault
  - Stuck-at faults
  - Timing errors

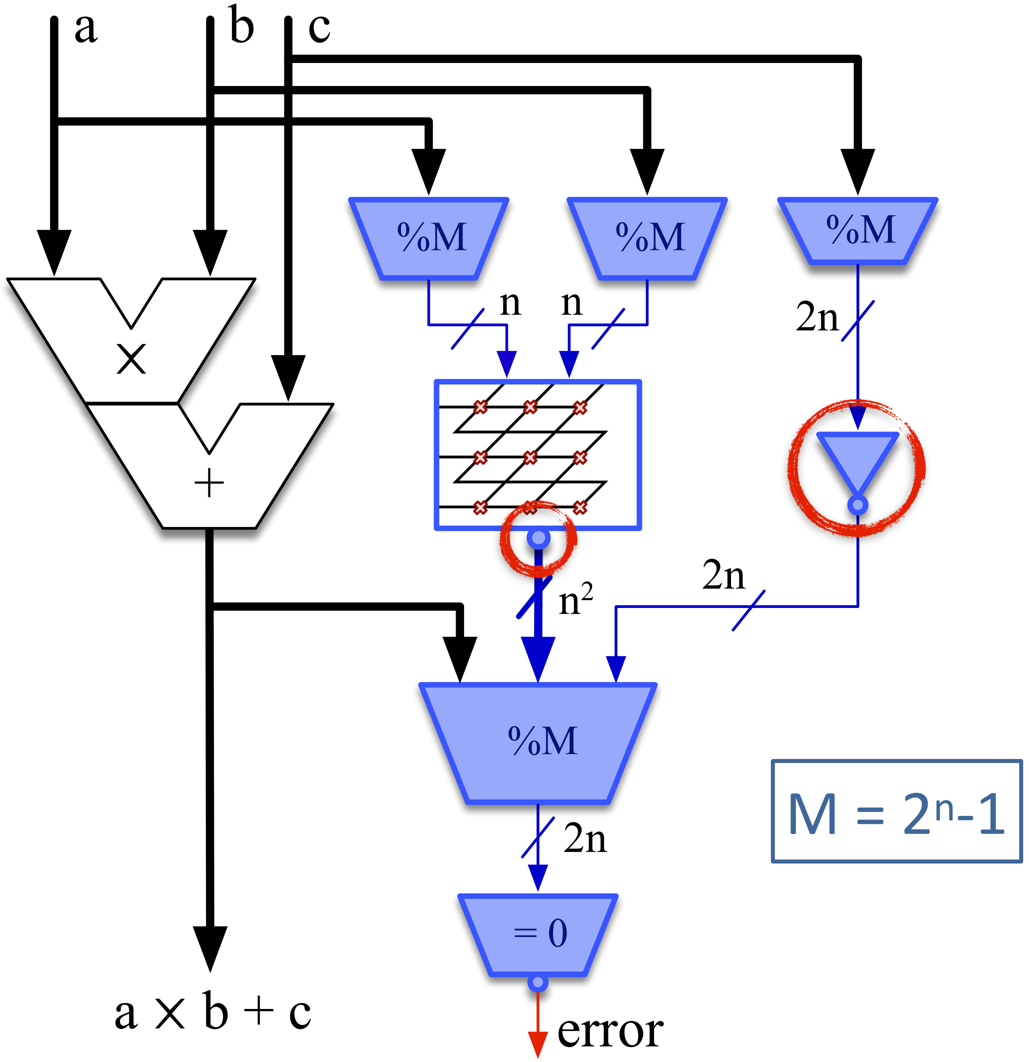
Did you forget Murphy’s law?

Reliability Evaluation: Our Approach

- Model Single-Event Transients (SETs)
  - Flip random gate output at random cycle
- **Single bit flip** for each experiment
- **All gates** are sampled
  - Both combinational and sequential logic
  - Both main and shadow datapaths
- Harder test than other fault models
  - Stuck-at faults last multiple cycles
  - Superset of flip-flop bit flips
- **Reliability Metric:** probability of undetected error

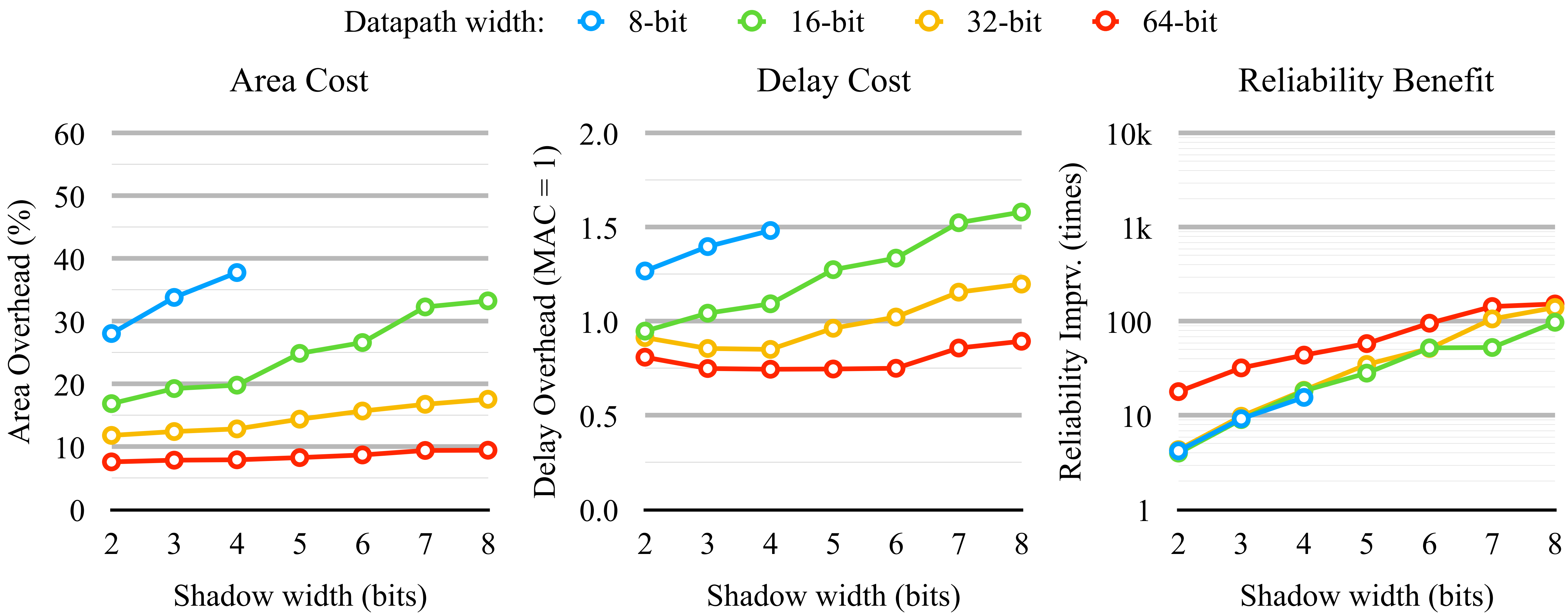
Self-Checking Multiply Accumulate (MAC) Design

- Reducers double as summation blocks
- Negation = Bitwise NOT
- Avoid reduction below 2*n* bits
  - Similar to carry save
- Zero comparator optimized for 2*n* bit input
  - Handles non-normalized input
- Shadow datapath uses only 1× gates
  - Area and energy efficient



M = 2^n-1

Self-Checking MAC Cost-Effectiveness

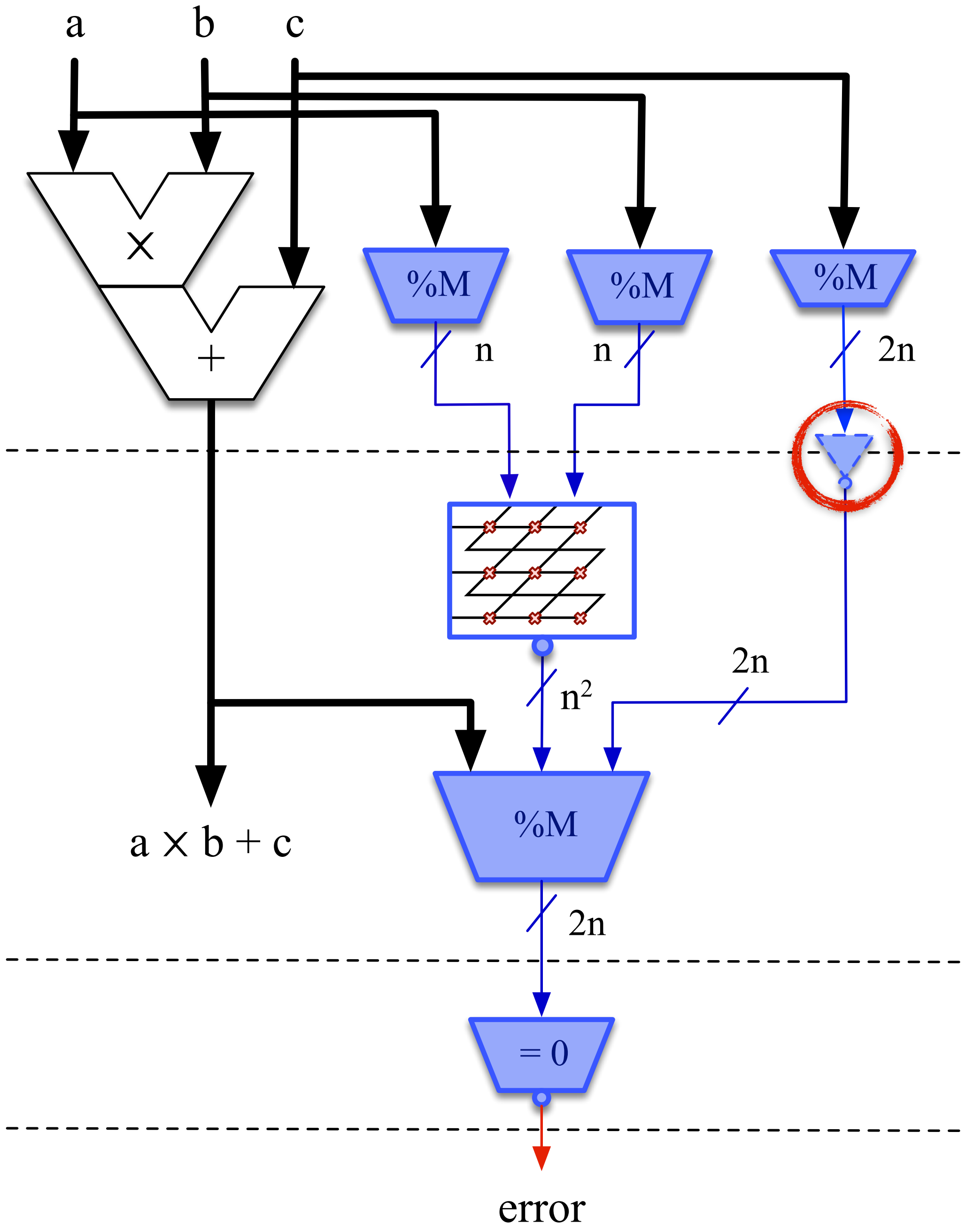


Aside: Error Recovery

- Error detection should be integrated into a higher-level recovery strategy
  - Restart the accelerator
  - Flush the pipeline
  - Rollback to a checkpoint

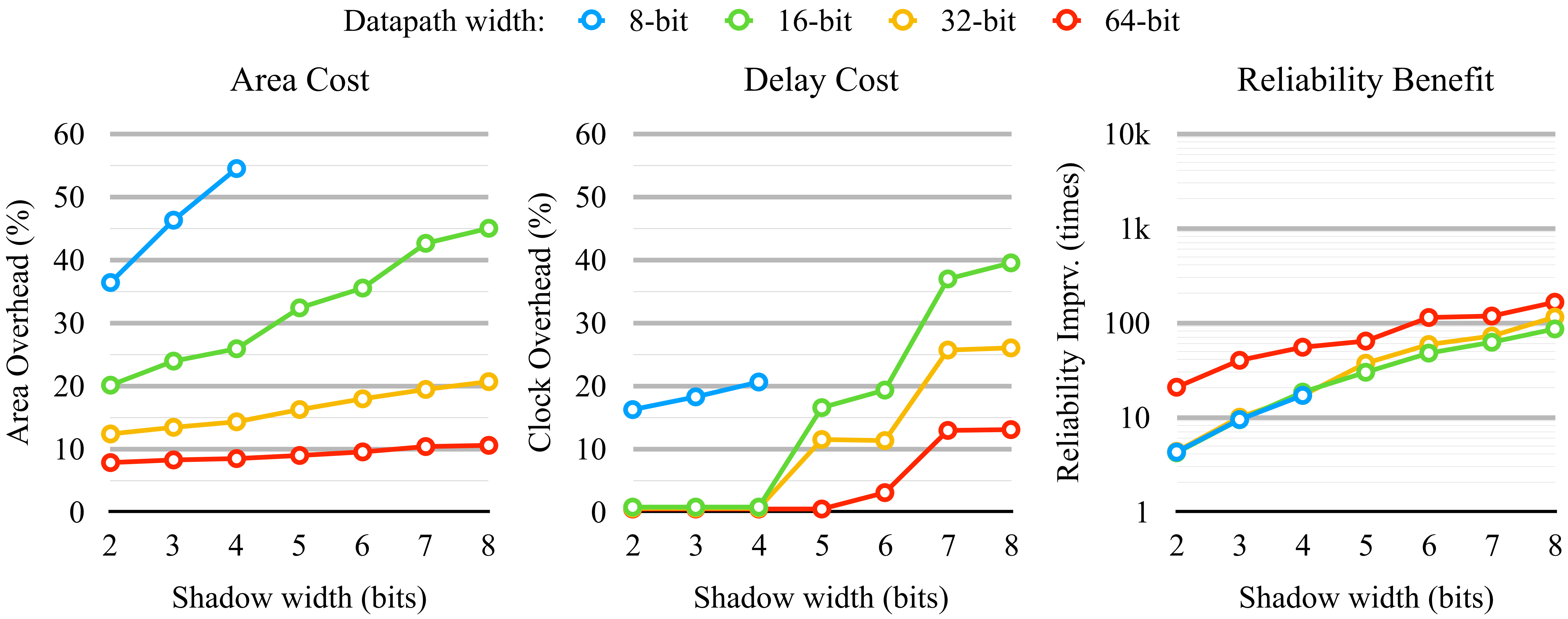
Adding pipeline stages

- Bake inverters into flip-flops
- Error signal has 2-cycle delay
  - Not in the critical path in error-free operation
- Target clock period is minimum delay of original MAC



Cycle boundary

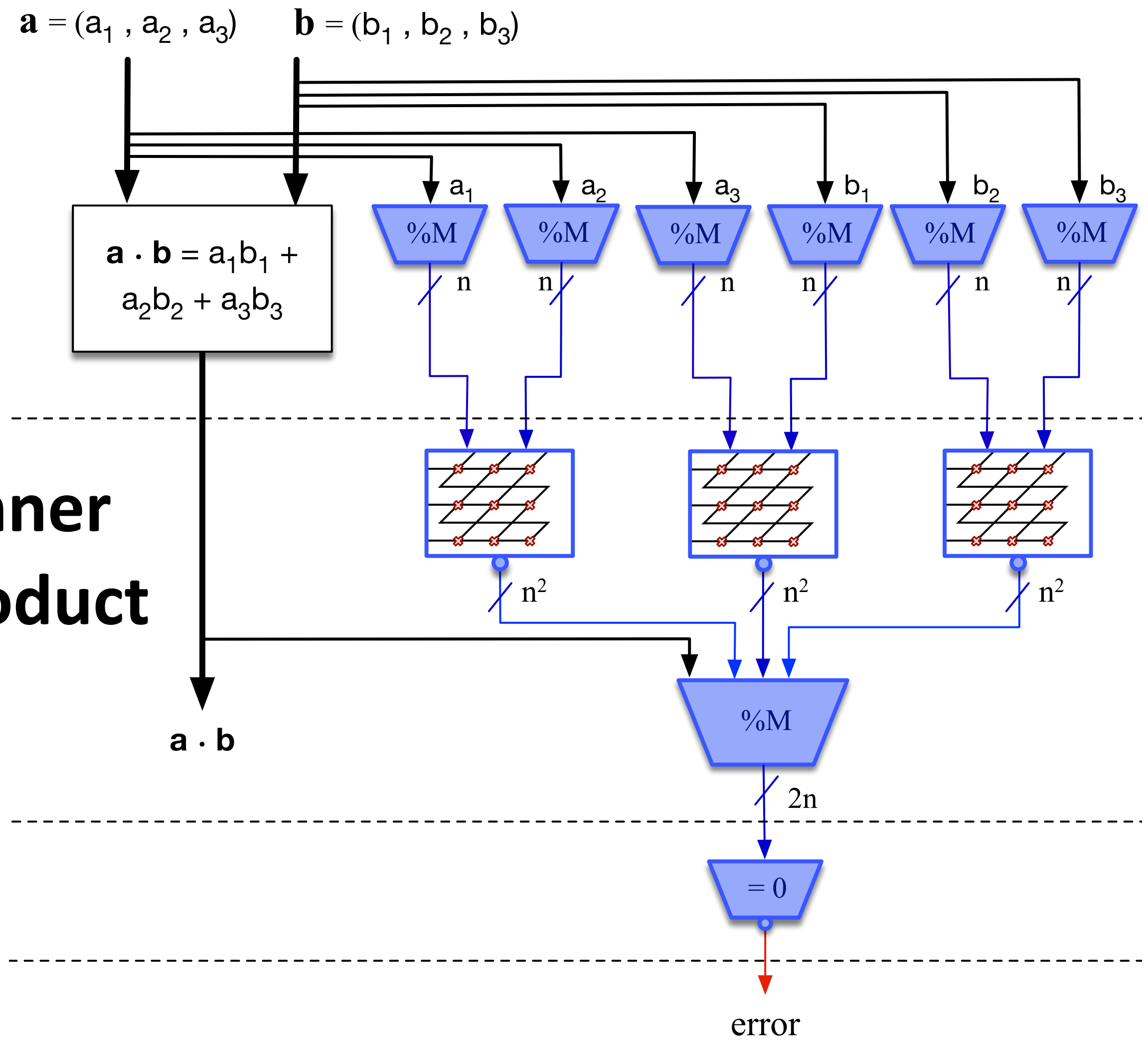
Pipelined MAC Cost-Effectiveness



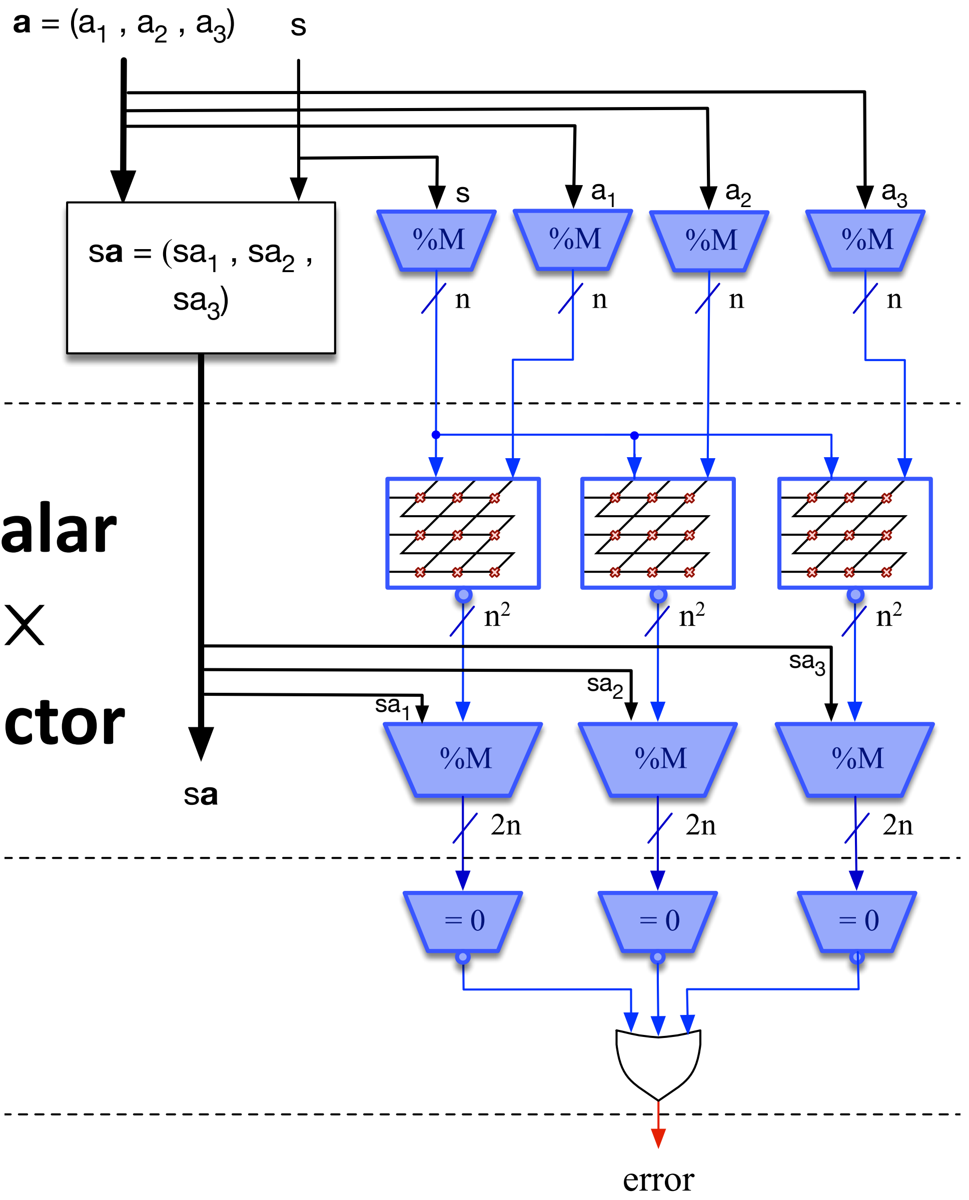
Aside: Pushing Maximum Performance

- Clock period target is very aggressive: single multiplier
- Pushing shadow datapath f_{max} higher is possible:
  - Retime flip-flops
  - Resize gates
  - Add pipeline stages

Self-Checking Linear Algebra

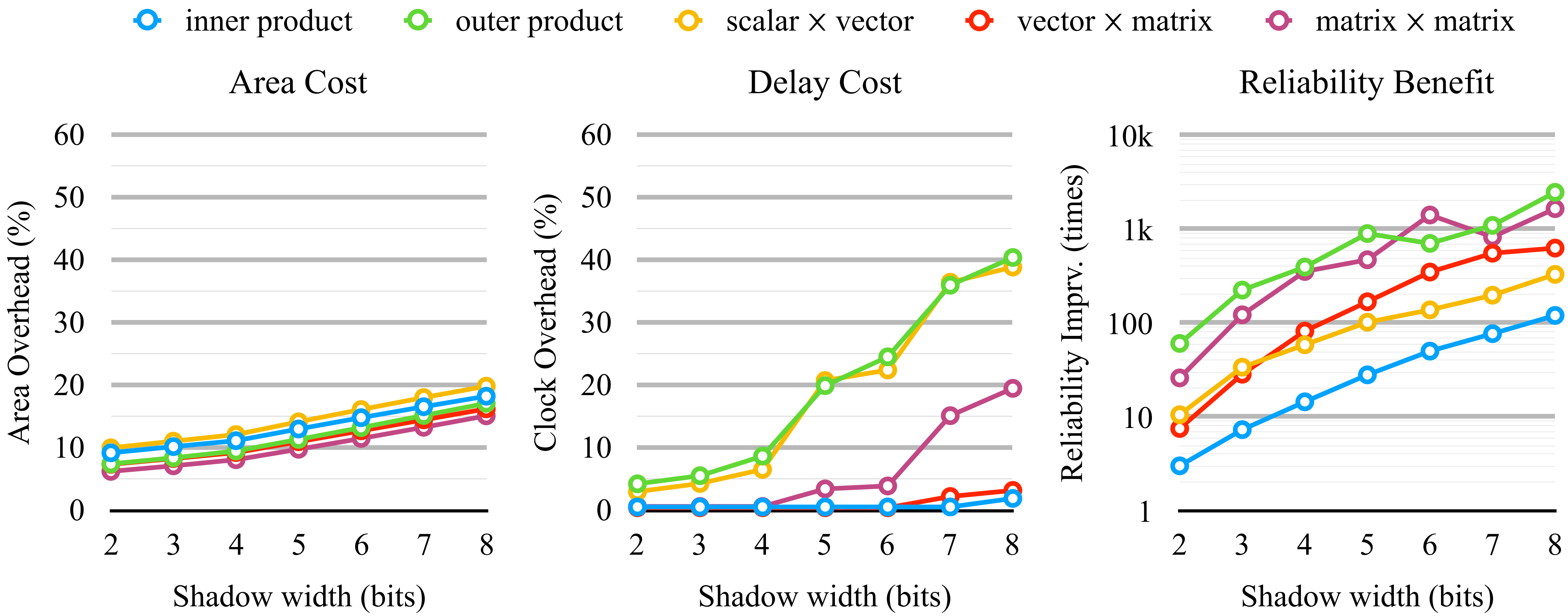


Inner Product



Scalar × Vector

Linear Algebra Primitive Cost Effectiveness (32-bit datapath)



Conclusions: Modulo Shadow Datapaths

- Revisited gate-level optimization
  - Found significant savings
- Implemented self-checking linear algebra
  - Key application for accelerators
- Considered single-event transients (SETs)
  - Hard fault to detect
- Technique is cost-effective
  - 6–10% area cost for 3–61× reliability benefit
  - 15–20% area cost for 121–2,477× reliability benefit

- Future work
  - Fixed-point optimization
  - Synthesis automation
  - Other kinds of shadow datapaths