Long Residue Checking for Adders

Michael Sullivan and Earl E. Swartzlander, Jr.

The University of Texas at Austin

Introduction

Error-free addition is important.

- Addition is heavily utilized for data manipulation, memory addressing, and control flow throughout the system.
- Errors can manifest in many ways, ranging from silent data corruption to catastrophic system failure.

Despite this, dynamic error detection in adders remains relatively expensive. We propose a novel separable error detection mechanism that provides **strong**, **low latency** error detection for a single fast adder at **less cost than any other separable design**.

Long Residue Checking

We investigate a modified residue checker which checks for cancellation.

Traditional Residue Checking

$$\left| |a|_A + |b|_A \right|_A = |c|_A$$

Modified Residue Checking

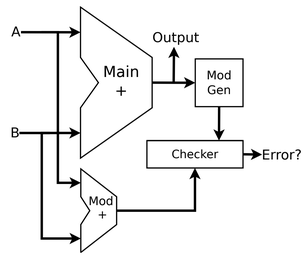
$$\left| |a|_A + |b|_A - |c|_A \right|_A = 0$$

The modified checker with the largest possible residue width ($a = n$) is the:

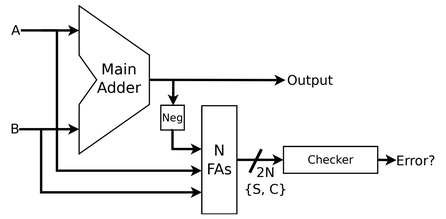
- least complex
- most power efficient
- has the highest error coverage
- has the lowest latency

We call this checking algorithm the *Long Residue Checker (LRC)*. The LRC can detect a fault in any single component, providing complete coverage against single event upsets (SEUs).

Long Residue Checking (cont.)



(a) Residue Checking

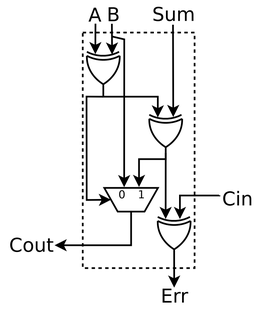


(b) Long Residue Checking ($a = n$)

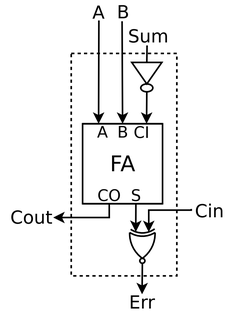
Prior Work

The LRC shares similarities with the *lazy error checker*.

- Bit-sliced design
- Similar error coverage



(c) Lazy Checker



(d) LRC

LRC Benefits

Only standard cells:

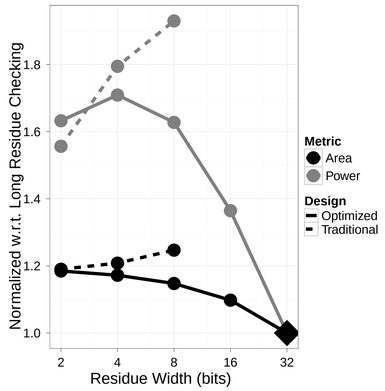
1. FA cell leads to benefits
   - ~10% less area
   - ~20% less power
2. Simplified design
   - Just a CSA/checker!

Evaluation

Separable detection mechanisms.

Design	Latency
Lazy Checker	1 cycle
DMR (Serial)	
DMR (Sklansky)	
Residue Checking	{2,3,4} cycles

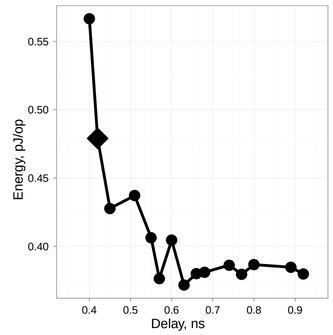
LRC vs. Residue Checking



Lazy Checker		
Word Width	Area (%)	Power (%)
16	36	48
32	9	24
64	10	23
48	10	25
Duplication (Serial Prefix)		
Word Width	Area (%)	Power (%)
16	65	28
32	92	37
64	97	41
48	100	38
Duplication (Sklansky)		
Word Width	Area (%)	Power (%)
16	193	111
32	205	98
64	186	59
48	188	48

Methodology

Pareto-efficient 16-bit adder designs. The highlighted baseline minimizes the ED^2 metric.



All selected baselines.

Adder Width	Delay (ns)	Area (μm^2)	Power (mW)
16	0.42	381.35	0.657
32	0.55	848.79	1.133
64	0.67	1546.06	1.616
128	0.7	3247.85	3.161

The overhead of long residue checking.

Adder Width	% Area Overhead	% Energy Overhead
16	38	69
32	33	70
64	36	84
128	34	86

Key Findings

Savings over the state-of-the-art.

- ~10% area and ~25% power

Simple, modular design.

- Bit-sliced design
- Maximum fanout of 1
- Perfect for standard cell synthesis

Further optimizations possible.

Some standard cells (e.g. D flip-flops) offer dual rail outputs with little additional complexity or power usage.

- Further ~7% area, ~12% power savings

Mail: mbsullivan@utexas.edu

Web: http://webspace.utexas.edu/mbs954