Low-cost duplication is an unobtrusive, drop-in replacement for duplication that maintains its strengths without its prohibitive overheads.
What can go wrong?

Transient Faults

Environmental Timing Faults

Fabrication Faults

Permanent Faults

Design Faults

Rates? Importance? … In the future?
Error Detection Design Goals

1. Strong and Fault-Agnostic
2. Fully Separable
3. Concurrent and Low-Latency
4. Manageable Cost
Typical Separable Approaches

- Strong-ish
  + Concurrent/Low Latency
  - Expensive
Typical Separable Approaches

(a) Duplication

(b) Lazy Duplication

? Strong-ish
+ Concurrent/Low Latency
- Expensive

+ Strong
? Medium Latency
? Expensive-ish
Typical Separable Approaches

(a) Duplication
- Main Unit
- Checker
- Output
- Error?

(b) Lazy Duplication
- Main Unit
- (Buffering)
- Equality Check
- Error?

(c) Residue Checking
- Main Unit
- %
- Checker
- Output
- Error?

- Strength
- Concurrent/Low Latency
- Expensive
- Medium Latency
- Expensive-ish
- Strong
- Strong-ish
- Expensive

- Coverage holes
- Medium Latency
- Inexpensive
Fixed-Point Multiplication

(1) Partial Product Generation

(2) Multi Operand Add

(3) Carry Propagate Addition

$$\begin{array}{cccc}
1 & 1 & 1 & \\
\times & 1 & 1 & 0 \\
\hline
 & 0 & 0 & 0 \\
 & 1 & 1 & 1 \\
 & 1 & 1 & 1 \\
\hline
0 & 1 & 0 & 0 & 1 & 0 & \text{Sum} \\
+ & 0 & 0 & 1 & 1 & 0 & 0 & \text{Carry} \\
\hline
0 & 1 & 0 & 1 & 0 & 1 & 0 & \text{Product}
\end{array}$$
Duplicate Multiplication

Original Multiplication:
- PP Gen
- Multi-Operand Addition
- Carry-Prop. Addition

Strict Duplicate:
- PP Gen
- Multi-Operand Addition
- Carry-Prop. Addition
- Equality Checker

Strict Duplication

Lazy Duplicate:
- PP Gen
- Multi-Operand Addition
- Carry-Prop. Addition
- Equality Checker

“Lazy” or “Deferred” Duplication
Low-Cost Duplication

+ Strong
+ Low-Latency
+ Inexpensive or moderately expensive

Diagram:

- Main Unit
- Modified Equality Check
- Error?
- Output
- Low Cost Unit

A

B
Low-Cost Duplication

ALTERNATE NUMBER SYSTEMS
1. Carry-Save Arithmetic
2. The Residue Number System

+ Strong
+ Low-Latency
+ Inexpensive or moderately expensive
#1 Lazy Carry-Save Duplication

Lazy Duplicate

Original Multiplication

Lazy CS Duplicate

Lazy Carry-Save Duplication
#1 Lazy Carry-Save Duplication

## Carry-Save Equality Checker

- \( S \)
- \( C \)
- \( Z \)

\[ C + S = Z \]

## A Bitslice

- \( S \)
- \( C \)
- \( Z \)
- \( S' \)
- \( C' \)
- \( Cin \)
- \( Error \)
Uses the **Residue Number System (RNS)**

Weighted number systems

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>$</td>
<td>$</td>
<td>🍒</td>
</tr>
</tbody>
</table>

RNS

<table>
<thead>
<tr>
<th>7</th>
<th>$</th>
<th>$</th>
<th>🍒</th>
<th>🍒</th>
<th>🍒</th>
</tr>
</thead>
</table>

? > 🍒  🍒  🍒

#2 RNS Duplication
#2 RNS Duplication

Uses the **Residue Number System (RNS)**

- **Original Multiplication**
  - PP Gen
  - Multi-Operand Addition
  - Carry-Prop. Addition
  - Modified Checker

- **Even Base \(2^a\) RNS Multiplication**
  - PP Gen
  - Multi-Operand Modular Addition
  - Modified Checker

- **Mersenne Odd Base \(2^b-1\) RNS Multiplication**
  - RNS Gen
  - PP Gen
  - Multi-Operand Modular Addition
  - Modified Checker

- **Fermat Odd Base \(2^b+1\) RNS Multiplication**
  - RNS Gen
  - PP Gen
  - Multi-Operand Modular Addition
Low-Cost Multiplication Evaluation

Design and Latency (@ 32-bits)

Area (µm²)

FU  LZ  CS  RNS
30% latency
FU  LZ  CS  RNS
40% latency
FU  LZ  CS  RNS
50% latency
FU  LZ  CS  RNS
60% latency
Low-Cost Multiplication Evaluation

![Graph showing area comparison for different designs and latencies](image)

**Design and Latency (@ 32-bits)**

- FU
- LZ
- CS
- RNS

Latency:
- 30%
- 40%
- 50%
- 60%
Low-Cost Duplication

Reduce overheads of duplication for high-reliability processors.

More aggressive baseline for DMR organizations.

Opens up new research opportunities.
Future Research?

- Different organizations and number systems
- Optimized low-speed checking multiplier
- Improving the power of the checker
  - Multi-Vth synthesis and multiple supply voltages
- Reliability for speed/power
  - Timing/power speculation or stochastic logic
- Other application domains? Security?
Backup Slides
Arithmetic Reliability

Memory

Rates High

Data Storage

Low-Level Correction

No Semantic Information

Arithmetic

Rates Low-to-???

Data Transform

Low-Level Detection

Data Types Matter
Arithmetic Reliability

Memory

- Rates High
- Data Storage
- Low-Level Correction
- No Semantic Information

Arithmetic

- Rates Low-to-???
- Data Transform
- Low-Level Detection
- Data Types Matter
Background #2: Fully Separable

Inseparable

Separable
Background #3 & #4

**Low-Latency, Concurrent Detection**

- Simplifies higher-level recovery mechanisms.
  - Target: Synchronous, 1-2 cycle latency.

**Manageable Costs**

- Full duplication > 100% logic overheads (area and power).
  - Low-cost duplication 30-50% overheads.
Background #2: Fully Separable

Increasing Separability

Signal Independence?

Design Independence?

Design Opacity?

Timing Opacity?

Systematicity?

(1) Non-Systematic

(2) Inseparable

(3) Design Co-Dependent

(4) Design Reactive

(5) Timing Reactive

(6) Fully Separable
Background #2: Fully Separable

Increasing Separability

- Signal Independence?
  - Y: Systematicity?
    - Y: (1) Non-Systematic
    - N: (2) Inseparable
  - N: Design Independence?
    - Y: Design Co-Dependent
    - N: Design Reactive
- Design Opacity?
  - Y: (4) Design Reactive
  - N: Timing Opacity?
    - Y: (5) Timing Reactive
    - N: (6) Fully Separable
Background #2: Fully Separable

- Increasing Separability
- Systematicity? (Y N)
  - Signal Independence? (Y N)
  - Design Independence? (Y N)
  - Design Opacity? (Y N)
  - Timing Opacity? (Y N)
  - Fully Separable
  - Design Co-Dependent (1) Non-Systematic
  - Inseparable (2) Non-Systematic
  - Design Reactive (3) Design Co-Dependent
  - Timing Reactive (4) Design Reactive
  - (5) Timing Reactive
Background #2: Fully Separable

- (6) Fully Separable
- (5) Timing Reactive
- (4) Design Reactive
- (3) Design Co-Dependent
- (2) Inseparable
- (1) Non-Systematic

Increasing Separability

- Systematicity?
  - Y N
- Signal Independence?
  - Y N
- Design Independence?
  - Y N
- Design Opacity?
  - Y N
- Timing Opacity?
  - Y N

- Fully Separable
Background #2: Fully Separable

Diagram:
- Increasing Separability
- Systematicity?
- Y N
- (1) Non-Systematic
- Y N
- (2) Inseparable
- Y N
- (3) Design Co-Dependent
- Y N
- (4) Design Reactive
- Y N
- (5) Timing Reactive
- (6) Fully Separable

- Signal Independence?
- (1) Non-Systematic
- Y N
- (2) Inseparable
- Y N
- (3) Design Co-Dependent
- Y N
- (4) Design Reactive
- Y N
- (5) Timing Reactive
- (6) Fully Separable

- Design Independence?
- (1) Non-Systematic
- Y N
- (2) Inseparable
- Y N
- (3) Design Co-Dependent
- Y N
- (4) Design Reactive
- Y N
- (5) Timing Reactive
- (6) Fully Separable

- Design Opacity?
- (1) Non-Systematic
- Y N
- (2) Inseparable
- Y N
- (3) Design Co-Dependent
- Y N
- (4) Design Reactive
- Y N
- (5) Timing Reactive
- (6) Fully Separable

- Timing Opacity?
- (1) Non-Systematic
- Y N
- (2) Inseparable
- Y N
- (3) Design Co-Dependent
- Y N
- (4) Design Reactive
- Y N
- (5) Timing Reactive
- (6) Fully Separable