# Newton-Raphson Integer Division for Area-Constrained Microcontrollers

Nima Badizadegan

ARITH 2023
September 4–6, 2023


t0 labs

# Motivation

- Small MCUs often have single-cycle multiply, but slow divide
  - Non-restoring division or radix-2 digit recurrence
  - Sometimes microcode or software
- Fast algorithms take area for lookup tables
  - High-radix digit recurrence needs lookup table
  - Iterative algorithms need an initial approximation
- Motivating application: cryptographic MCU for algorithms that benefit from constant-time modulus, no FPU

t0

# Division by Iteration

- Start with an approximation and refine
- Quadratic convergence, 2 FMAs per round
- Large CPUs use multiplicative iteration for better parallelism, but we have no parallelism

$$
\begin{aligned}
x_0 &= A_{1/x}(D) \\
\epsilon_n &= 1 - Dx_n \\
x_{n+1} &= x_n + x_n\epsilon_n = x_n(2 - Dx_n)
\end{aligned}
$$

t0

# Initial Approximations

- Target precision: 8 bits
- Typical approach: Faithful bipartite ROM tables
  - Piecewise linear approximation
  - 3.3 kbit for 8-bit precision + supporting HW
- Our approach: Quantized 4th order polynomial approximation, integrated into existing multiplier circuits

t0

# Expression Form and Coefficients

$$\frac{1}{x} \approx 7.15766 - 20.1824x + 28.0257x^2 - 19.17x^3 + 5.17028x^4$$

$$\Downarrow \text{Factor: } 3 \rightarrow 2 \text{ multiplications}$$

$$\frac{1}{x} \approx 5.1703(1.9687 - 2.6834x + x^2)(0.70321 - 1.02432x + x^2)$$

$$\Downarrow \text{Quantize coefficients}$$

$$\frac{1}{x} \approx 101_b(c_1 - 10.1011_b x + x^2)(c_2 - 1.00001_b x + x^2)$$

t0

# Constants and Quantization

$$A_{1/x}(x) = 101_b \underbrace{(c_1 - 10.1011_b x + \underbrace{x^2}_{13\text{ bit}})}_{14\text{ bit}} \underbrace{(c_2 - 1.00001_b x + \underbrace{x^2}_{13\text{ bit}})}_{12\text{ bit}}$$

- Co-select constants and bit widths for each term: Optimal constants are not necessarily rounded unquantized values
- Brute force with 32b word, SAT/LP solver for 64b
- We set $c_1$ and $c_2$ to underestimate so $DA_{1/x}(D) < 1$

t0

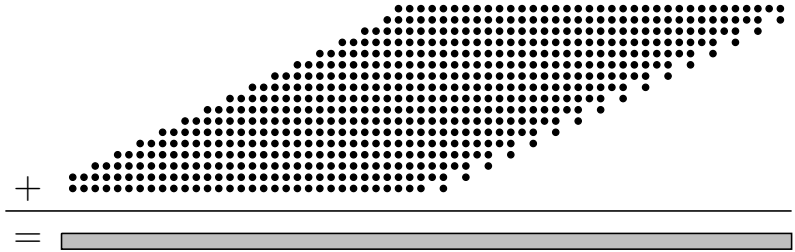
# Mapping to Hardware

$$A_{1/x}(x) = 101_b \underbrace{(c_1 - 10.1011_b x + x^2)}_{p_1} \underbrace{(c_2 - 1.00001_b x + x^2)}_{p_2}$$

- $13 \times 13$ multiply for $x^2$
- 7-term addition for $p_1$
- 5-term addition for $p_2$
- $12 \times 14$ multiply with post-add for $5p_1p_2$

t0

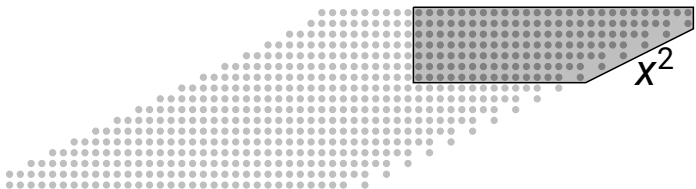
# Fitting into a Multiplier




- Booth code multiplier
- Vertical cuts: AND gates
- Horizontal cuts: multiplexing within compressor tree

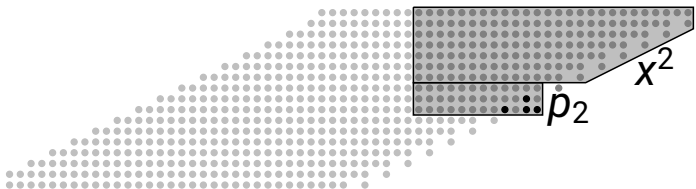
t0

# Fitting into a Multiplier



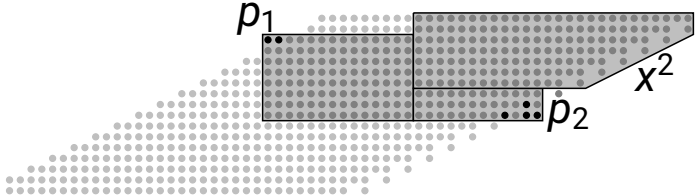

t0

# Fitting into a Multiplier



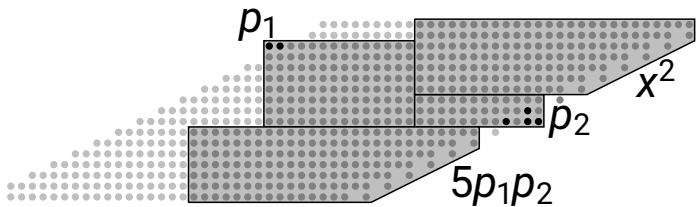

t0

# Fitting into a Multiplier




t0

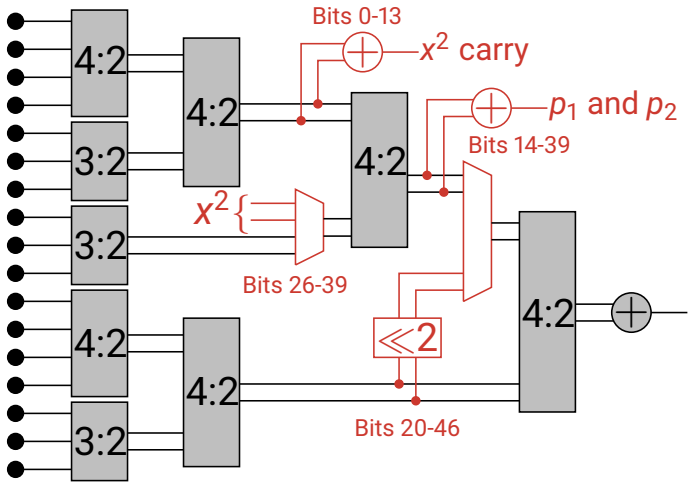
# Fitting into a Multiplier




- Horizontal cuts at bits 7 and 10 - need a compressor tree that fits

t0

# Modified Compressor Tree

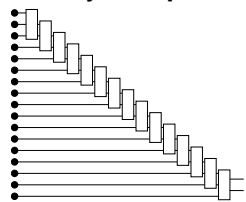



t0

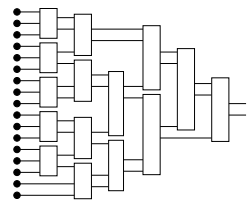
# Comparison to Common Compressor Trees

- All options use equivalent of 15 full adder cells
- 2 extra logic levels compared to Wallace Tree
- Same depth as 4:2 compressor tree with 4:2 standard cells

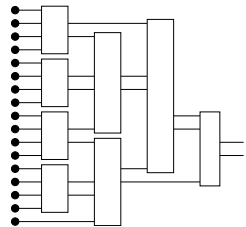
**Array Multiplier**



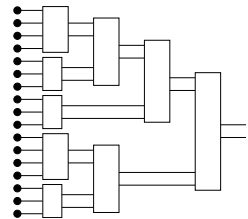
**Wallace Tree**



**4:2 Compressor Tree**



**Our Compressor Tree**



t0

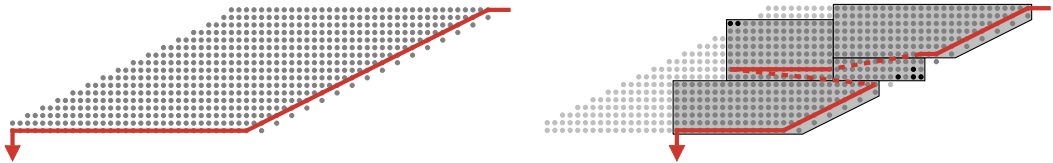
# Circuit Size Expansion for Division Estimation

- 275 2:1 MUXes
- 46 adder cells
- 10 AND gates
- Under 350 gates total

| | Count | Element |
|---|---|---|
| **Generation of $x^2$** | | |
| Booth encoding MUXes | 14 | 2:1 MUX |
| Partial product MUXes | 26 | 2:1 MUX |
| Carry out | 14 | Adder Cell |
| **Generation of $p_1$ and $p_2$** | | |
| New multiplier bits | 6 | Adder cell* |
| Carry chain breaks | 10 | AND gate |
| Adding $x^2$ to $p_1$ | 28 | 2:1 MUX |
| Partial product overrides | 106 | 2:1 MUX |
| Final computation adders | 26 | Adder Cell |
| **Final product computation** | | |
| Booth encoding MUXes | 13 | 2:1 MUX |
| Partial product MUXes | 24 | 2:1 MUX |
| Multiplication by 5 | 32 | 2:1 MUX |
| Extra output MUX port | 32 | 2:1 MUX* |

t0

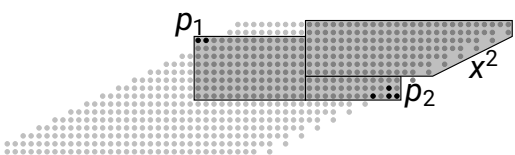
# Multiplication vs. Division Critical Path



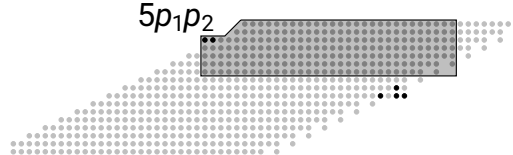
- Critical paths similar length, but use the same cells in different order
- Problem for STA-based design flows

t0

# Two-Cycle Circuit: Bit Mapping




**Cycle 1**

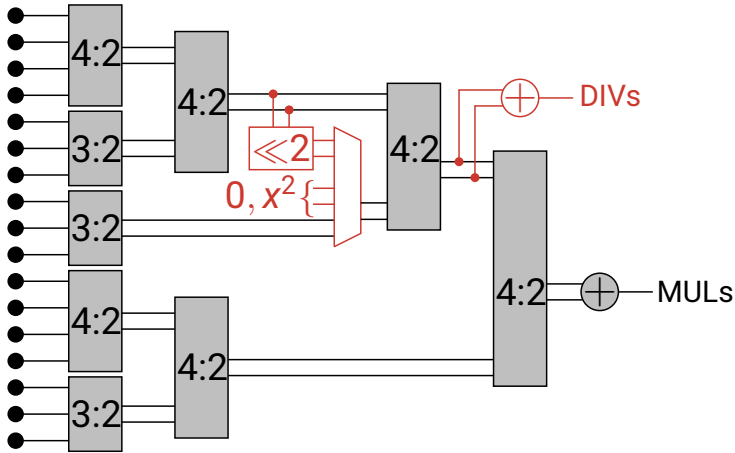



**Cycle 2**

- Separate output port from the top of the multiplier array, used for both cycles

t0

# Two-Cycle Circuit: Compressor Tree




t0

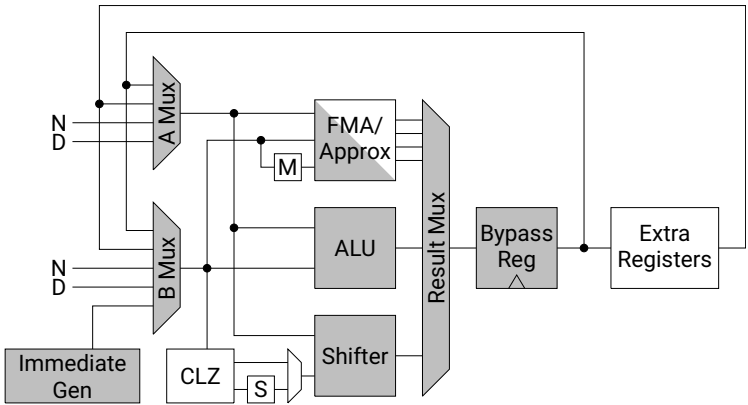
# Division Estimation Circuit Synthesis

| Synthesized Circuit | Logic Elements | $f_{max}$ from STA |
|---|---|---|
| Wallace Tree Multiplier | 2,172 | 80 MHz |
| Multiplier Tree Only | 2,175 | 76 MHz |
| Division Estimation Only | 1,310 | 82 MHz |
| One-cycle MUL/DIV | 2,340 (+7.6%) | 43 MHz (-41%) |
| Two-cycle MUL/DIV | 2,345 (+7.8%) | 71 MHz (-6.6%) |

- Synthesized on MAX10 FPGA as a proxy for ASIC
- STA pessimistically estimates 41% frequency loss for one-cycle circuit

t0

# Division State Machine




- Self-contained system with just MCU datapath (gray) and added circuits for division (white)

t0

# Division State Machine

- RISC-V MCU datapath (two operands, one result)
- Added leading zero count to shifter
- Added two registers
- Added multiplier capabilities for division:
  - Division approximation
  - Integer FMA with shadow register for 3rd operand
  - Adding $2^{32}$ to operand A on a signed multiplication
  - Negating a multiplication
  - Right-shifting output by 1

t0

# Division State Machine Algorithm

| Cycle | Description | Operation |
|---|---|---|
| 1 | Left-justify $D \rightarrow D' \in [0.5, 1)$ in UQ0.32 | $D' = D \ll \text{CLZ}(D)$ |
| 2 | Estimate $X = 1/D$ in UQ1.31 | $X = A_{1/x}(D')$ |
| 3 | NR round 1: Get error in UQ0.32 | $T = \text{mulhi}(X, D') \gg 1$ |
| 4 | NR round 1: Refine $X$ | $X = \text{mulhi}[X, (2^{32} - T)]$ |
| 5 | NR round 2: Get error | $T = \text{mulhi}(X, D') \gg 1$ |
| 6 | NR round 2: Refine $X$ | $X = \text{mulhi}[X, (2^{32} - T)]$ |
| 7 | Calculate quotient | $Q' = \text{mulhi}(N, X)$ |
| 8 | Shift quotient back | $Q = Q' \gg \text{CLZ}(D)$ |
| 9 | Calculate remainder | $R = N - DQ$ |
| 10 | Check remainder | $(R \geq D)?R = R - D$ |
| 11 | Fix quotient (if needed) | $(R \geq D)?Q = Q + 1$ |

t0

# Division State Machine Algorithm

- Compute MOD before DIV
- 11 cycles for full computation of 32 bit integer division
  - 1 cycle preparation
  - 6 cycles core division algorithm
  - 2 cycles finalization
  - 2 cycles correction
- Preparation and finalization steps unique to integer ops
- Cycles 10-11 (correction) come from truncation in cycle 6, 11th cycle only when *N* is full width

t0

# Division State Machine: Early Termination

- Optional early termination for non-cryptographic division
- Uses leading zero count to upper-bound result MSB index

| Case | Condition | Total Cycles |
|---|---|---|
| Divide by zero | $D = 0$ | 3 |
| Power of 2 | $D' = 0.5$ | 3 |
| Known zero result | $MSB < 0$ | 4 |
| 8-bit precision (skip NR) | $0 \leq MSB < 8$ | 6–7 |
| 16-bit precision (1 NR rounds) | $8 \leq MSB < 16$ | 8–9 |
| Full precision (2 NR rounds) | $16 \leq MSB < 32$ | 10–11 |

t0

# Division State Machine Synthesis

| Component | Size(LEs) |
|---|---|
| FMA with division estimation | 2,491 |
| Leading zero count | 47 |
| Power of 2 detection | 13 |
| Control state machine | 56 |
| Extra registers | 66 |
| Common datapath components | 587 |

- 3,260 LEs total in MAX10 FPGA
  - 2,673 of MUL/DIV components vs. 2,172 for MUL
- 501 LEs added for fast division vs. no division (+18%)
- Synthesizes to $f_{max}$ of MUL/DIV circuit (42 or 72 MHz)

t0

# Conclusions

- Microarchitecture for 32-bit integer division
- 11 cycle DIV and 10 cycle MOD with early termination
- Use quantized polynomial instead of lookup tables, combined with multiplier
- Combined MUL/DIV circuit is 8% larger than multiplier at 7% frequency cost
- Fast division for comparable increase in area to bit-at-a-time algorithms

t0