New Division Algorithms By Digit Recurrence

Jo Ebergen, Ivan Sutherland,
Ajanta Chakraborty

Sun Microsystems Laboratories

In The News

“The Pentium Bug,” 1997

“Pope Ends Long Division, ”
LA Times, May 2001

Introduction

- **A novel division algorithm**
- **Compared to radix-2 division with carry-save addition:**
  - **Simpler selection logic**
    - **reduces critical path delay**
    - **no carry-completion adder**
  - **But larger digit set {-2,-1,0,1,2} instead of {-1,0,1}**
  - **On average 36% less additions**
- **Novel graphical representation of operations on remainder in carry-save format**

The Problem

- Compute Q = N/D
- Q = quotient
- N = dividend $N \in [-2^{K+1}, 2^{K+1})$
- D = divisor $D \in [\, 2^{K}, 2^{K+1})$
- Error must be less than 2^{-L}
  - L = # fractional bits
- For IEEE floating point format, choose
  - K=0 and L=23 or L=52

Invariants Of Repetition

- Goal:
  - $Q*D = N$
- Introduce variables q, r, c, and n
  - q = quotient “thus far”
  - r = remainder “thus far”
- Invariants
  - $q*D + c*r = N$ and $c=2^{-n}$
  - Range invariant: $|r| \le 2*D$
- Initially, q=0, c=1, r=N, n=0
- Upon termination $|c*r| < 2^{-L}*D$

General Algorithm

```
q:=0; c:=1; r:=N; n:=0;
while B0 do {
   if     B1 then { r:=r*2; c:=c/2; n:=n+1 }
   elseif B2 then { r:=r-D; q:=q+c }
   elseif B3 then { r:=r+D; q:=q-c }
   endif
}
```

- **Maintains invariant $q*D + c*r = N$**
  - **independent of choice for B0-B3 !**
- **Choice of B0-B3 affect range invariant $|r| \leq 2*D$**
- **What are costly operations?**

An Optimization

- **When each add/subtract is followed by a "shift"**
- **Let each repetition end with a "shift"**
- **Value of $c = 2^{-n}$ is next quotient digit q_n**
- **Body of repetition:**

```
if     B1 then { qn:=0 }                 "2X"
elseif B2 then { r:=r-D; qn:=1 }         "SUB + 2X"
elseif B3 then { r:=r+D; qn:=-1 }        "ADD + 2X"
endif
r:=r*2; n:=n+1
```

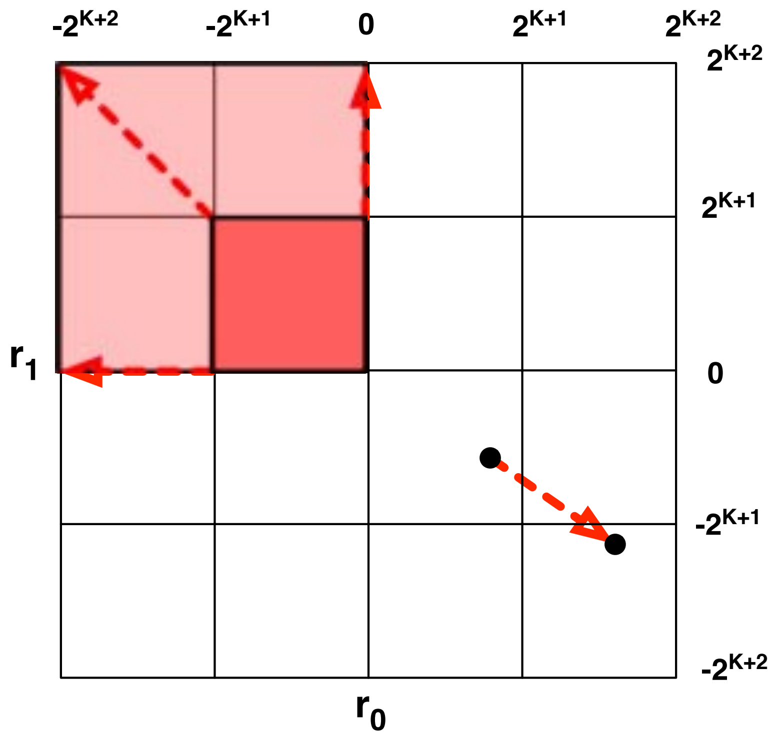
Carry-Save Addition

- Speeding up additions with carry-save addition
- Split remainder r in r0 and r1
  - r = r0 + r1
  - r0 = “sum bits”
  - r1 = “carry bits”
- Addition/subtraction implemented by full adder
  - Example: subtract D from r

```
r0,r1 := add(r0, r1, -D)
```

Operations in (r0,r1) Plane

- Use 2's complement representation
- K+3 nonfractional bits
- Operations:
  - Double
  - Translate over (t,-t) or (-t, t) with $t = 2^{K+1}$
  - Carry-save add with -2D, -D, D, or 2D



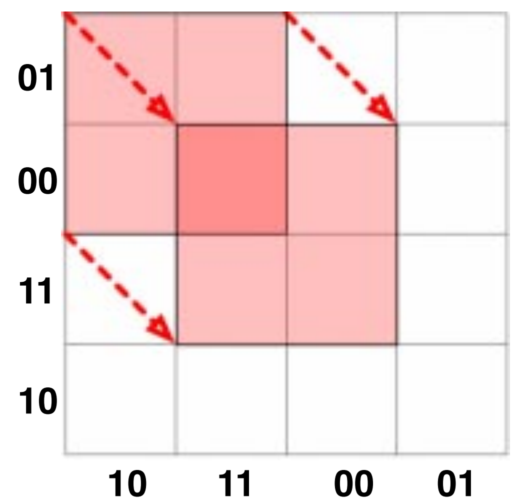
Double: r0,r1 := 2*r0, 2*r1
n:=n+1

Translations

- **Translation over (t,-t) or (-t,t) with $t=2^{K+1}$ and K+3 non-fractional bits**
- **Maintains invariants**

bit K+2 bit K+1

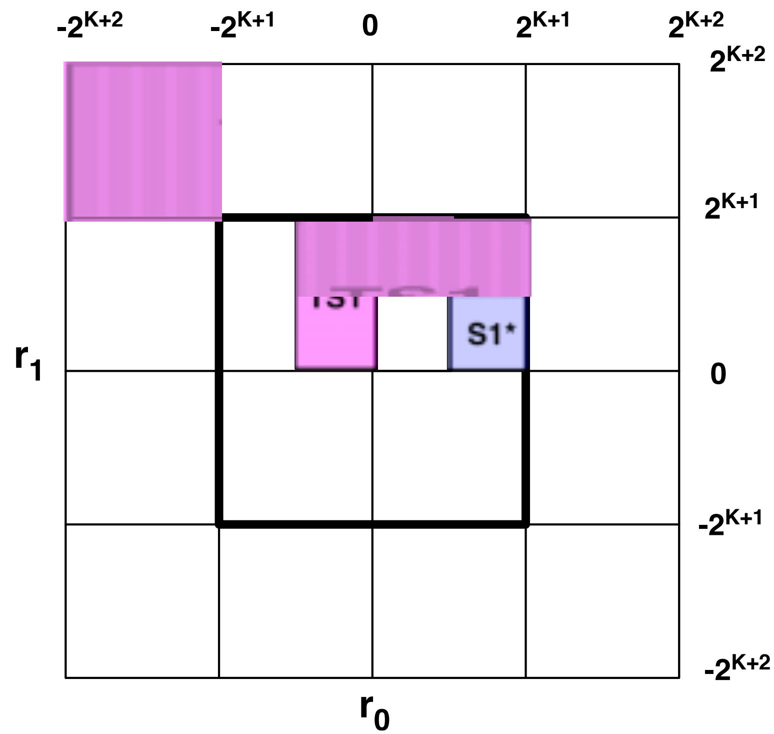
$$\begin{array}{ccc} 10 & \rightarrow & 11 \\ 11 & \rightarrow & 00 \\ 01 & \rightarrow & 00 \\ 00 & \rightarrow & 11 \end{array}$$



- **Bit K+1 changes and bit K+2 = bit K+1**
- **If you ignore bit K+2, translation = “invert MSB”**

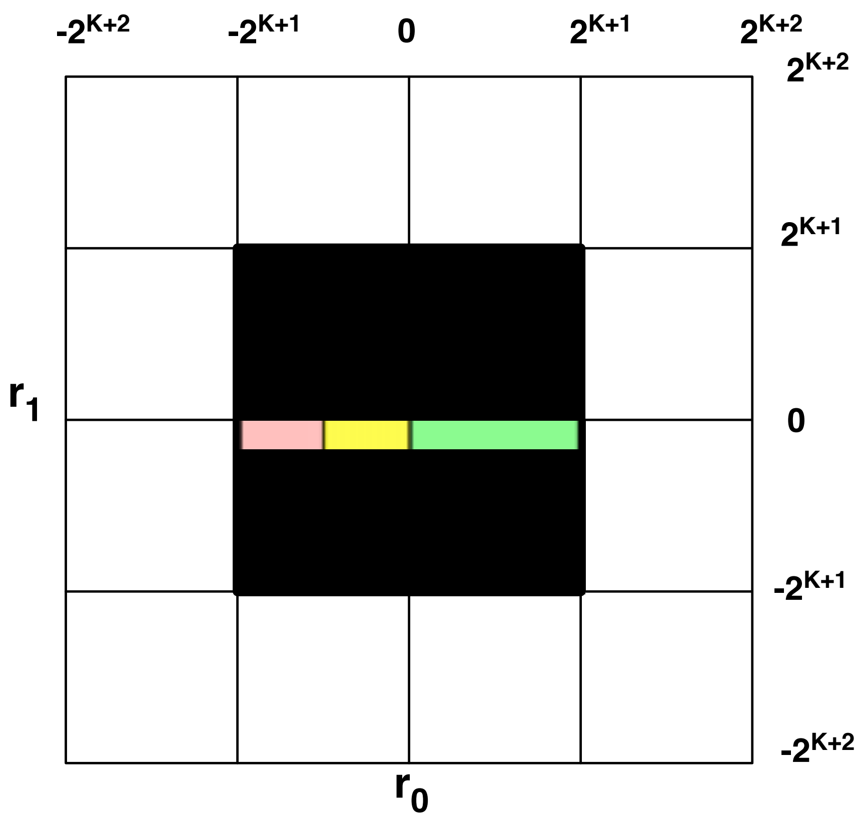
Carry-Save Additions

- Region S0 - 2D yields TS0
- Region S1 - D yields TS1
- Region S1* - D yields TS1
- Additions are done similar



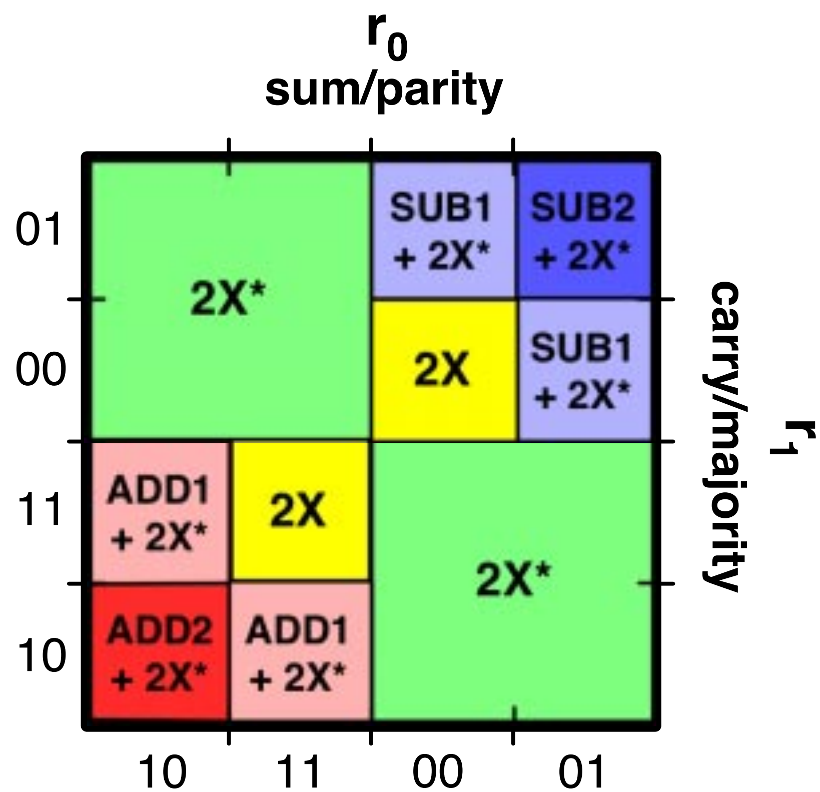
Range Invariant for Divine Division

- $r0 \in [-2^{K+1}, 2^{K+1})$ and $r1 \in [-2^{K+1}, 2^{K+1})$
- "Stay in bold square"
- Each region in bold square leads to different operations, but result lands again in bold square
- Each operation maintains invariants
- We may ignore bit K+2



"Divine" Division

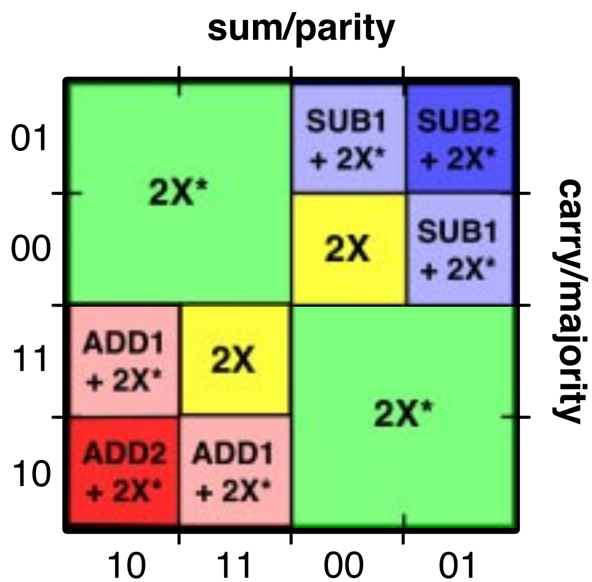
- **Represented graphically in (r0,r1) plane**

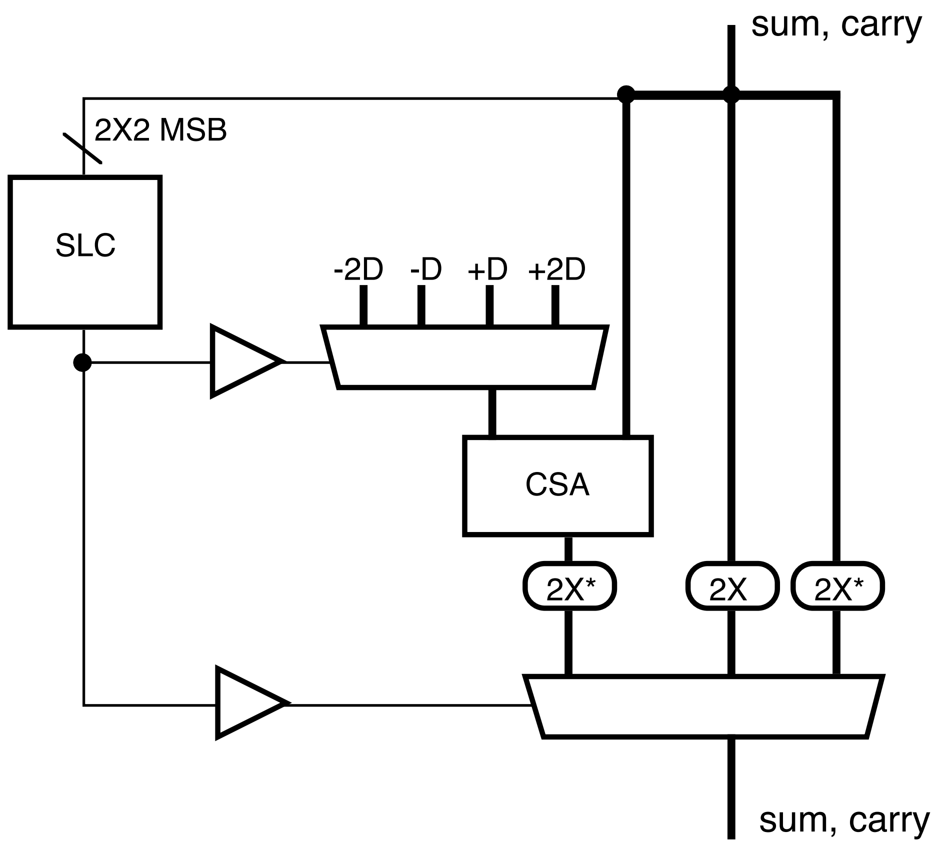


2X = double
2X* = double + invert MSB
SUB1 = subtract D
SUB2 = subtract 2D
ADD1 = add D
ADD2 = add 2D

"invert MSB" = translate

An Implementation





A Comparison

Algorithm	(Add+Sub) /Shift	# Rep's	Digit set	Bits to inspect for SLC	Carry-save addition
Carry-Save Division	0.66	K+L+1	-1,0,1	2x4	Yes
"Divine" Division	0.42	K+L+2	-2,-1,0,1,2	2x2	Yes

Concluding Remarks

- A novel division algorithm
- Compared to radix-2 division with carry-save addition:
  - Simpler selection logic, only 2x2 bits to inspect
  - Larger digit set {-2,-1,0,1,2}
  - One extra repetition step
  - On average 36% fewer additions
- Novel graphical representation of operations on remainder in carry-save format