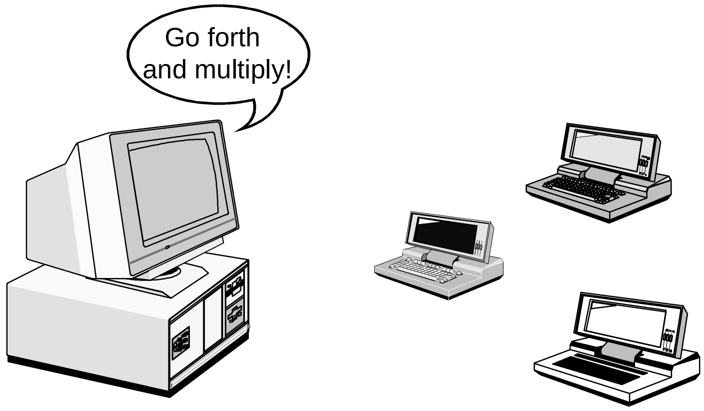
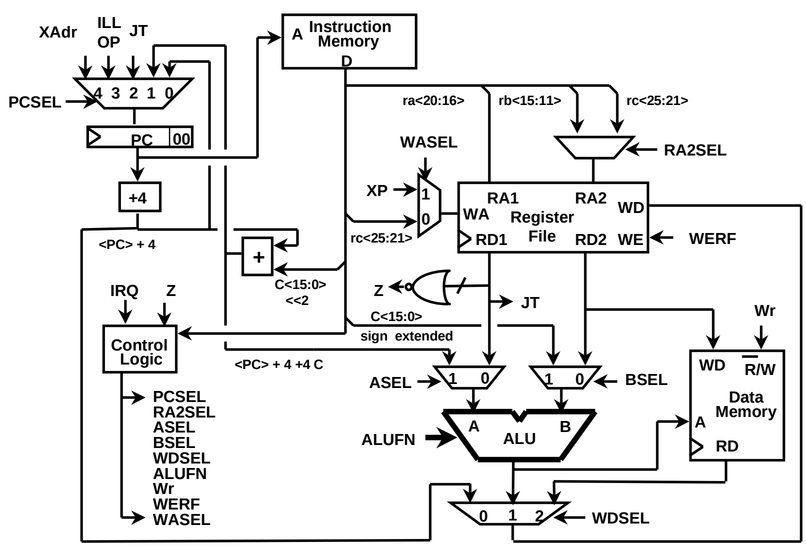
6004 Spring 1998: L34

Computer Arithmetic
(1/3 x 3 = 0.999999…)



1

β Implementation



2

Arithmetic Operations

- Integer add, multiply and divide
- Floating point and double precision multiply
- Floating point and double precision divide
- Square root, sine, cosine
- Do we implement above in hardware or software? Implement as combinational, sequential (serial), or pipelined circuit?

3

Integer Representations

Most significant bit is sign bit, remaining bits represent magnitude in some manner

– Sign-magnitude representation:

2_{10}	0 010	0_{10}	0 000
-2_{10}	1 010	-0_{10}	1 000

– One's complement representation:

2_{10}	0 010	0_{10}	0 000
-2_{10}	1 101	-0_{10}	1 111

– Two's complement representation: ←

2_{10}	0 010	0_{10}	0 000
-2_{10}	1 110	-0_{10}	0 000

4

IEEE Floating Point Representation (FPR)

← 32 bits →

s	exponent (E)	significand or fraction (F)
1	8	23

$$\text{value} = (-1)^{s} \times (1 + F) \times 2^{E - 127}$$

F is a fraction of the form .xxx … x

2^{-1} 2^{-2} 2^{-3}

0.5_{10} .100 … 0 0.375_{10} .011 … 0

Exception: **000 … 0** *represents zero!*

5

Converting IEEE FPR to Decimal

$$\text{value} = (-1)^{s} \times (1 + F) \times 2^{E - 127}$$

s	E	F		
1	01111110	1000 … 0	$-1 \times 1.5 \times 2^{-1}$	-0.75_{10}
1	10000001	0100 … 0	$-1 \times 1.25 \times 2^{2}$	-5.0_{10}
0	10000000	1100 … 0	____________	______

6

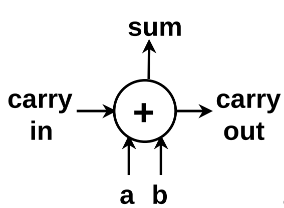
Converting Decimal to IEEE FPR

$$\text{value} = (-1)^s \times (1 + F) \times 2^{E - 127}$$

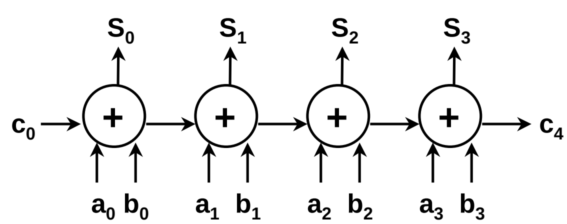
		s	E	F
-0.75_{10}	$-1 \times 1.5 \times 2^{-1}$	1	01111110	1000 … 0
-7.0_{10}	$-1 \times 1.75 \times 2^{2}$	1	10000001	1100 … 0
9.0_{10}	________	________		

7

Ripple Carry Adder



sum = a XOR b XOR cin
cout = a b + a cin + b cin

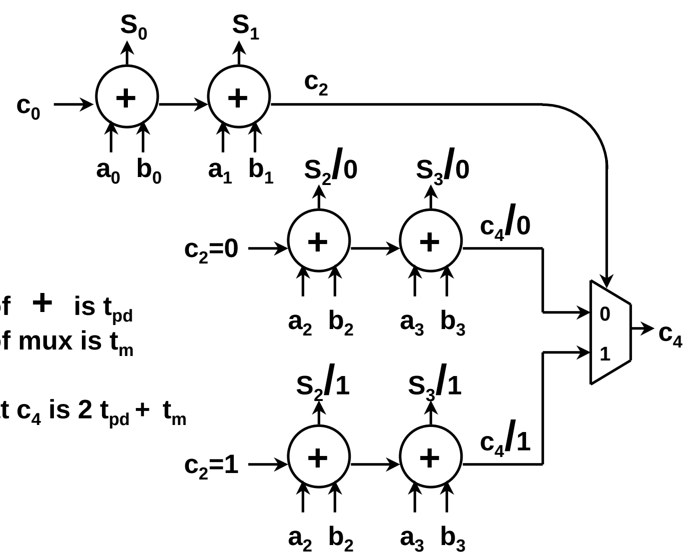


If delay of + is t_{pd}, then delay at c_4 is 4 t_{pd}

8

6004 Spring 1998: L34

Carry Bypass Adder

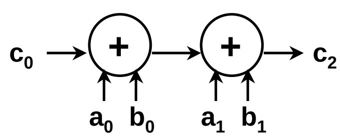


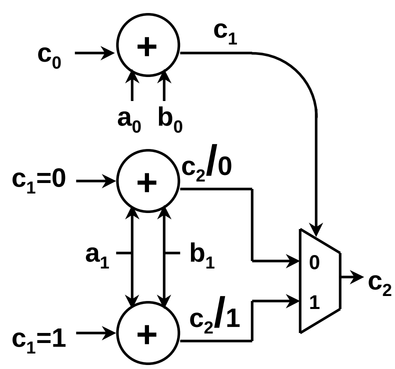
Delay of + is t_{pd}
Delay of mux is t_m

Delay at c_4 is $2\ t_{pd} + t_m$

9

Recursive Application





Delay at c_2 is $2\ t_{pd}$
for 2-bit ripple carry

Delay at c_2 is $t_{pd} + t_m$
for 2-bit carry bypass

Recursive 4-bit carry bypass delay at c_4 is $t_{pd} + 2\ t_m$

Recursive N-bit ($N = 2^k$) carry bypass delay at c_N is $t_{pd} + k\ t_m$

10

Floating Point Addition

- Match exponents by shifting number with smaller exponent to the right
- Add the significands (fractions)
- Normalize the sum
- Check for underflow and overflow in sum

$0.5_{10} = 1.000 \times 2^{-1}$ $-0.4375_{10} = -1.110 \times 2^{-2}$

-1.110×2^{-2} is smaller, convert to -0.111×2^{-1}

Add significands 1.000 and -0.111 to get 0.001

Sum is 0.001×2^{-1} , normalize to 1.000×2^{-4}

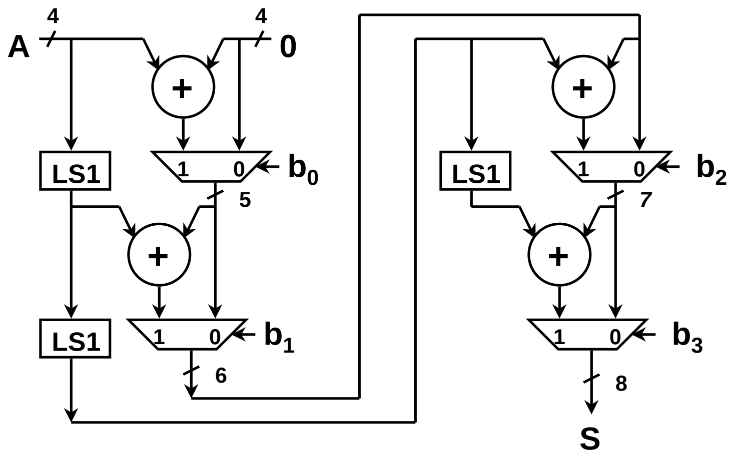
11

Multiplication

	A x B			a_3	a_2	a_1	a_0
				b_3	b_2	b_1	b_0
				a_3b_0	a_2b_0	a_1b_0	a_0b_0
			a_3b_1	a_2b_1	a_1b_1	a_0b_1	
		a_3b_2	a_2b_2	a_1b_2	a_0b_2		
	a_3b_3	a_2b_3	a_1b_3	a_0b_3			
s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0

12

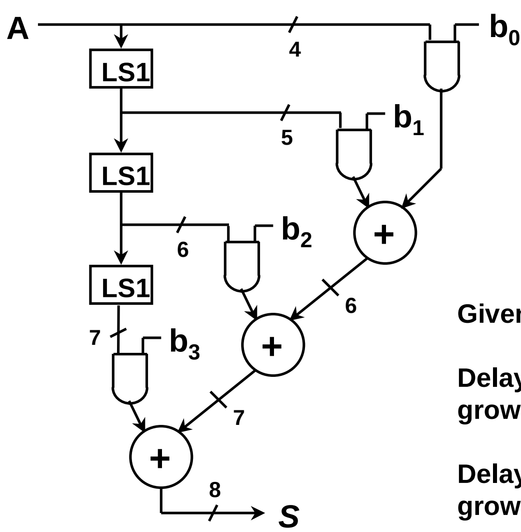
Naïve 4-bit Multiplier



LS1 is a Left Shift 1 module

13

4-bit Multiplier - II



Given N-bit inputs:

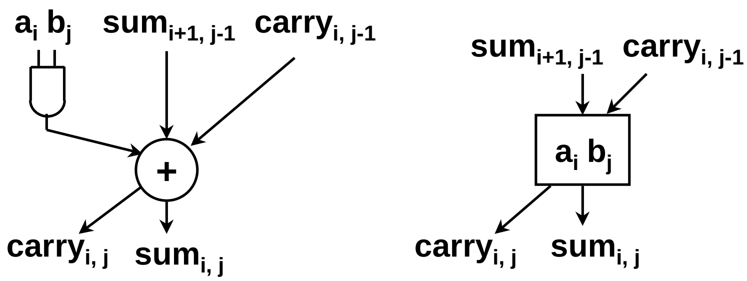
Delay of ripple adder grows as N

Delay of multiplier grows as _______

14

Carry-Save Array Multiplier

- Uses full adders and AND gates

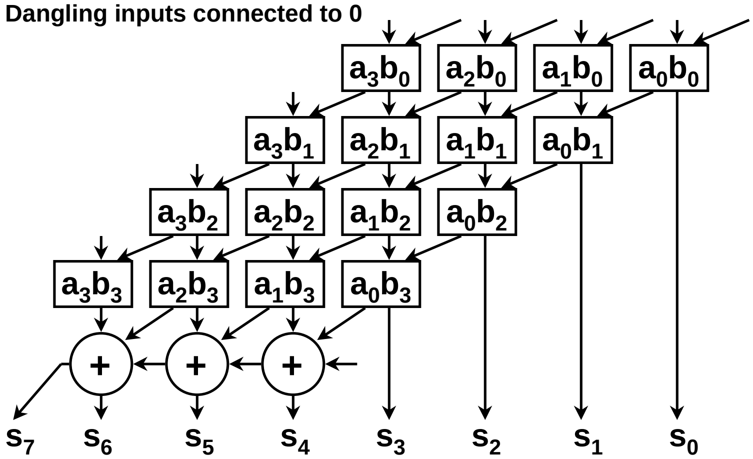


- Homogeneous array with N^2 cells shown above

15

Carry-Save Array Implementation

Dangling inputs connected to 0



Given N-bit inputs, delay of multiplier grows as ______
Can improve throughput using pipelining.

16

Floating Point Multiplication

- Floating point multiplication is easy!
  - Just add exponents using an integer adder
  - Multiply significands using integer multiplier
  - Sign bit computed as XOR of input sign bits
- Normalize product

$0.5_{10} = 1.000 \times 2^{-1}$ $-0.4375_{10} = -1.110 \times 2^{-2}$

Add exponents -1 + (-2) to get -3

Multiply 1.000 and 1.110 to get 1.110

Product is -1.110×2^{-3} , already normalized

17

Division

Dividend = Quotient x Divisor + Remainder

- Accomplished using repeated subtraction
- One quotient digit generated on each subtract

```
                    1001     Quotient
Divisor 1000 |   1001010     Dividend
               -1000
                    10
                    101
                    1010
                   -1000
                      10     Remainder
```

18

Latencies for Typical Modern Processor

- **Central Processing Unit (CPU):**
  - **Integer add: 1 cycle**
  - **32-bit integer multiply: 10 cycles**
  - **64-bit integer multiply: 20 cycles**
  - **32-bit integer divide: 69 cycles**
  - **64-bit integer divide: 133 cycles**
- **On-chip Floating Point Unit (FPU):**
  - **Floating point add: 4 cycles**
  - **Floating point multiply: 7 cycles**
  - **Double precision multiply: 8 cycles**
  - **Floating point divide: 23 cycles**
  - **Double precision divide: 36 cycles**

19