Chapter 9 problems 9.13, 9.16, 9.20, 9.22 — Homework #9 matt kruze

9.13 **[M]** If the product of two n-bit numbers in 2's-complement representation can be represented in n bits, the manual multiplication algorithm shown in Figure 9.6*a* can be used directly, treating the sign bits the same as the other bits. Try this on each of the following pairs of 4-bit signed numbers:

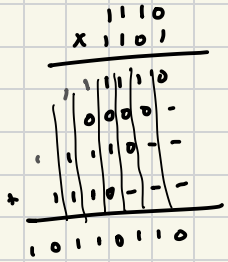
(*a*) Multiplicand $= 1110$ and Multiplier $= 1101$

(*b*) Multiplicand $= 0010$ and Multiplier $= 1110$

Why does this work correctly?

a) M: 1110 $(14)_{10}$ ← unsigned → b) m: 0010 (2)

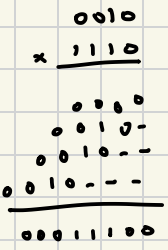
Q: 1101 $(13)_{10}$ Q: 1110 (14)



$= 10111010$ $(186)_{10}$

9-bit 7 6 5 4 3 2 1 0

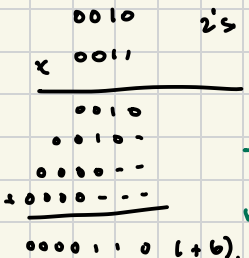
whoops... I think this is asking to multiply n-bit 2's comp to receive n-bit product



00011100 $(28)_{10}$

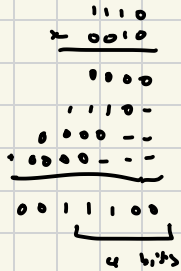
a) M: 1110 $(-2)_{10}$ signed b) m: 0010 (2)

Q: 1101 $(-3)_{10}$ Q: 1110 (-2)



$-2 \times -3 = +6$ ✓

$(+6)_{10}$



$2 \times -2 = -4$ ✓

$1100 = (-4)_{10}$

This works as the algorithm handles signage:
- multiplication operates identically on 2's complement sign encoding.

as opposed to $n \times n = 2n$ in unsigned.

Additionally, signed numbers have half the span of unsigned, thus an $n \times n$ results in n

Chapter 9 problems 9.13, 9.16, 9.20, 9.22

Homework #9 matt kreye

9.16 [M] Tree depth for carry-save reduction is analyzed in this problem.

(*a*) How many 3-2 reduction levels are needed to reduce 16 summands to 2 using a pattern similar to that shown in Figure 9.19?

(*b*) Repeat part (*a*) for reducing 32 summands to 2 to show that the claim of 8 levels in Section 9.5.3 is correct.

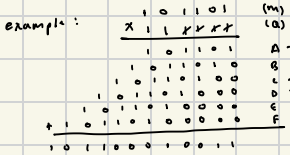
(*c*) Compare the exact answers in parts (*a*) and (*b*) to the results obtained by using the approximation developed in Example 9.3 in Section 9.10.

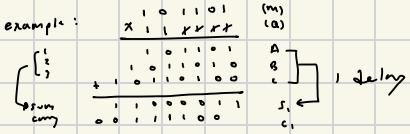
Textbook:

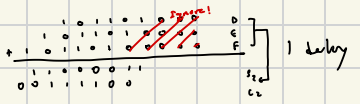
- performed on parallel groups

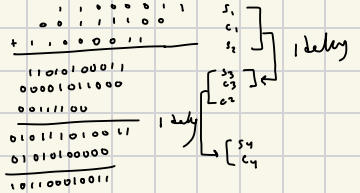
1. [generate s and c vectors
   1 full adder delay

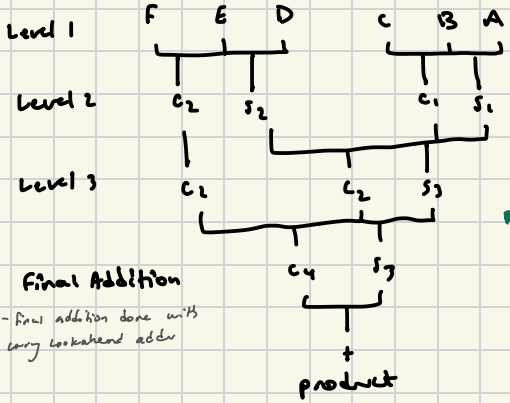
2. [group all s and c vectors into threes
   perform carry save addition on them
   1 full adder delay
   ... continue until 2 remaining vectors











- final addition done with carry lookahead adder

This yields a 1.7 log k - 1.7 CSA tree

Other Reducers:

- 4x2
- 7x3

See next page for problem

9.16 **[M]** Tree depth for carry-save reduction is analyzed in this problem.

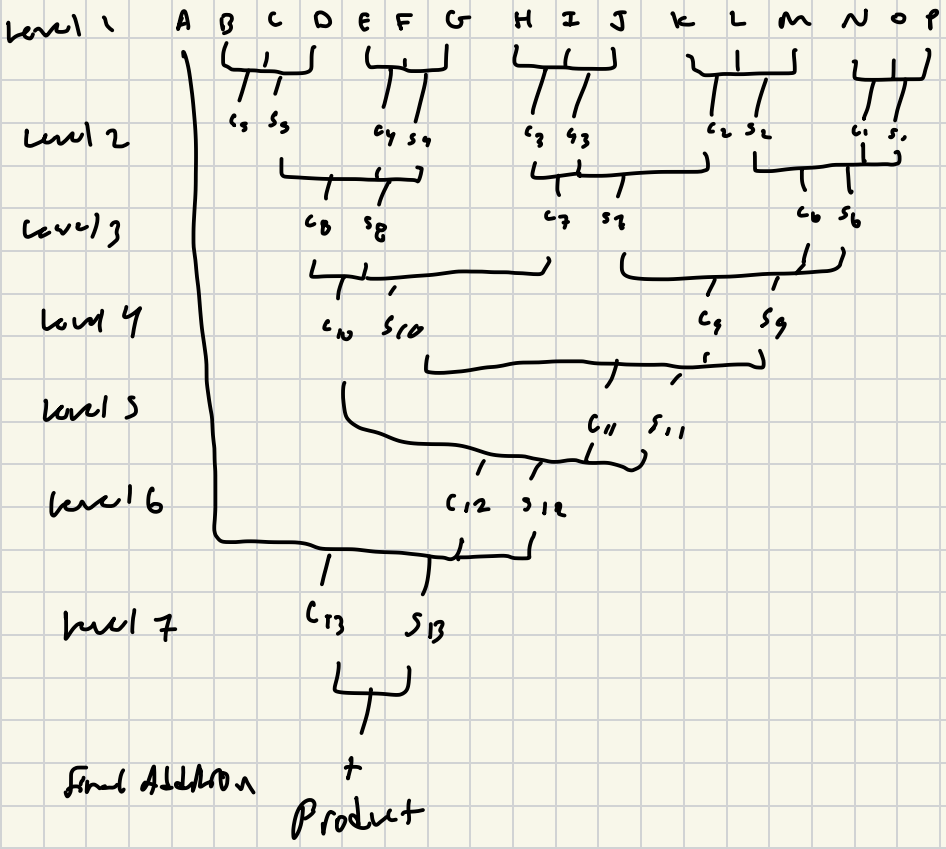
(*a*) How many 3-2 reduction levels are needed to reduce 16 summands to 2 using a pattern similar to that shown in Figure 9.19?

(*b*) Repeat part (*a*) for reducing 32 summands to 2 to show that the claim of 8 levels in Section 9.5.3 is correct.

(*c*) Compare the exact answers in parts (*a*) and (*b*) to the results obtained by using the approximation developed in Example 9.3 in Section 9.10.

$16 = 2^n = 4$

a) $1.7 \log_2 K - 1.7 \approx 1.7(\log_2 16) - 1.7 = 6.8 - 1.7 \approx 5.1$ levels



7 reducting levels are required before final addition can be reached

9.16 **[M]** Tree depth for carry-save reduction is analyzed in this problem.

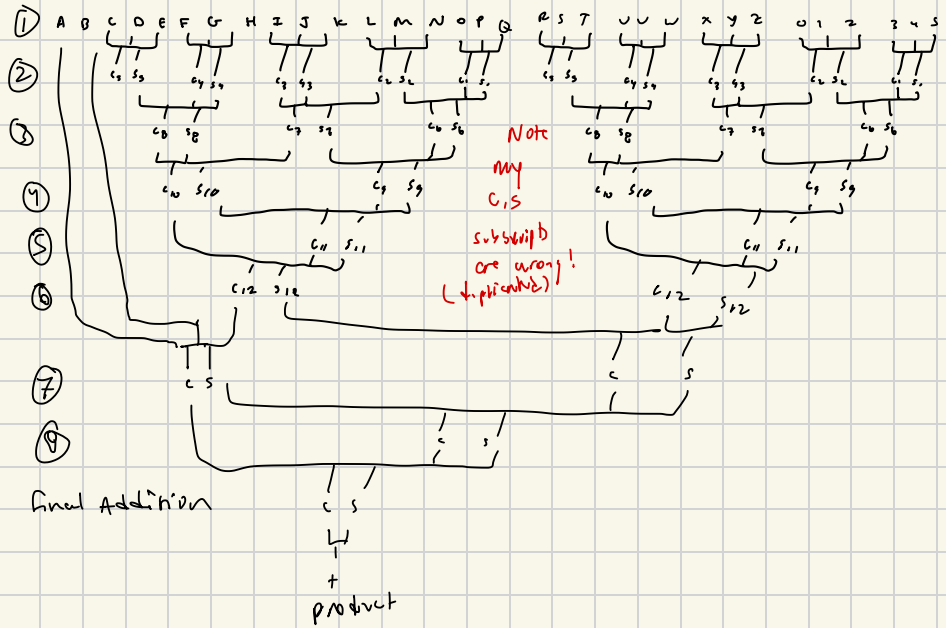
(*a*) How many 3-2 reduction levels are needed to reduce 16 summands to 2 using a pattern similar to that shown in Figure 9.19?

(*b*) Repeat part (*a*) for reducing 32 summands to 2 to show that the claim of 8 levels in Section 9.5.3 is correct.

(*c*) Compare the exact answers in parts (*a*) and (*b*) to the results obtained by using the approximation developed in Example 9.3 in Section 9.10.

b) claim 8 levels, $k = 32$ ✓ $32 = 2^n$ $n = 5$

$1.7 \log_2 k - 1.7 \rightarrow 1.7 \log_2(32) - 1.7 = 1.7 \cdot 5 - 1.7 \approx 6.8$ levels



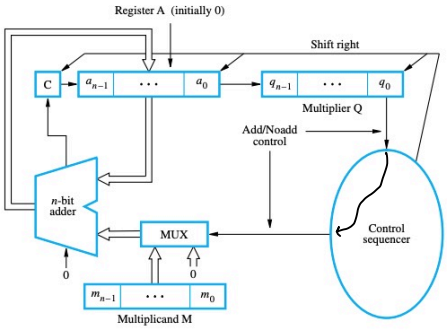
c) approximate is under in both cases for CSA levels

approx	actual
5.1	7
6.8	8

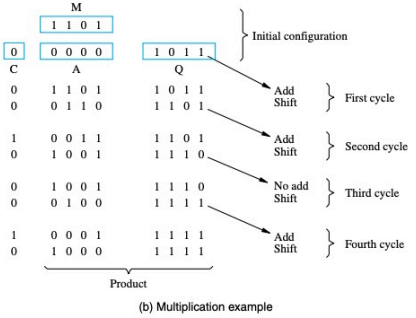
Chapter 9 problems 9.13, 9.16, 9.20, 9.22

Homework #9 matt knype

9.20 **[M]** Show how the multiplication and division operations in Problem 9.19 would be performed by the hardware in Figures 9.7*a* and 9.23, respectively, by constructing charts similar to those in Figures 9.7*b* and 9.25.



(a) Register configuration



(b) Multiplication example

Figure 9.7 Sequential circuit binary multiplier.

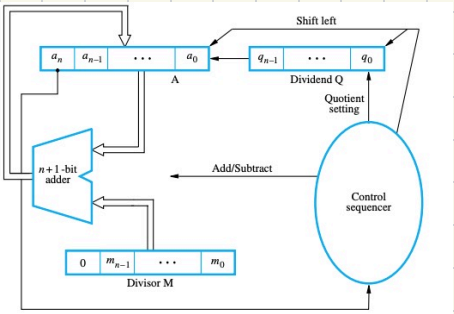


Figure 9.23 Circuit arrangement for binary division.

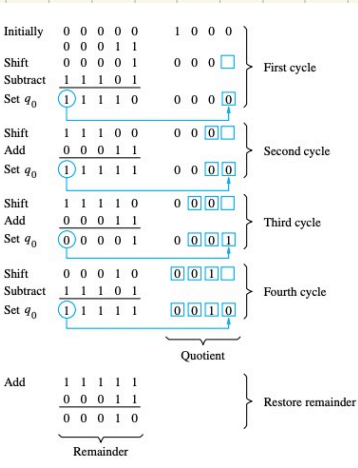


Figure 9.25 A non-restoring division example.

Sequential circuit binary multiplier:

- multiplication of 2 n-bit #
- sequential → n-bit adder

uses n-bit adder n times for serial addition
performed by ripple carry adders

* A & Q are shift registers holding PPi
multiplier bit generates q_i signal for add/no add

$q_i = 0$: select 0
$q_i = 1$: select multiplicand M to add to P(i-1)

start: load multiplier into register Q
- M, C, A cleared (0)

@ end of each cycle,
C, A, Q shifted right to generate PPi
causes q_i @ LSB.

problem solved on next page

9.20 [M] Show how the multiplication and division operations in Problem 9.19 would be performed by the hardware in Figures 9.7*a* and 9.23, respectively, by constructing charts similar to those in Figures 9.7*b* and 9.25.

9.19: using manual methods, perform the operations

$A \times B$

$A \div B$

on the 5 bit unsigned numbers

A: 10101

B: 00101

A x B

		C	A	Q	M	
			10101			initial config
	0		00000	00101		
Cycles						
1	0		10101	00101		ADD
	0		01010	10010		shift
2	0		01010	10010		no add
	0		00101	01001		shift
3	0		00101	01001		no add
	0		00010	00100		shift
4	0		10111	00100		Add
	0		01011	10010		shift
5	0		01011	10010		no add
	0		00101	11001		shift

product

A÷B

initial 00000 10101 / 00101

shift 0 00010 01010 □ } 1

subtract 11101

set q_0 11110 01010

shift 00010 10100 } 2

add 00101

set q_0 00001

shift 00010 01001 } 3

sub 11101

set q_0 11101 01001

shift 11101 10010 } 4

add 00101

set q_0 00010 10010

shift 00010 00101 } 5

sub 11011

set q_0 11111 00101

quotient

Add 11111 } restore

+ 00101 remainder

00100

remainder

Chapter 9 problems 9.13, 9.16, 9.20, 9.22 — Homework #9 matt kruze

9.22 [D] Consider a 16-bit, floating-point number in a format similar to that discussed in Problem 9.21, with a 6-bit exponent and a 9-bit mantissa fraction. The base of the scale factor is 2 and the exponent is represented in excess-31 format.

(*a*) Add the numbers *A* and *B*, formatted as follows:

A =	0	100001	111111110
B =	0	011111	001010101

Give the answer in normalized form. Remember that an implicit 1 is to the left of the binary point but is not included in the *A* and *B* formats. Use rounding as the truncation method when producing the final mantissa.

(*b*) Using decimal numbers *w*, *x*, *y*, and *z*, express the magnitude of the largest and smallest (nonzero) values representable in the preceding normalized floating-point format. Use the following form:

$$\text{Largest} = w \times 2^{x}$$
$$\text{Smallest} = y \times 2^{-z}$$

Add/Subtract Rule

1. Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.
2. Set the exponent of the result equal to the larger exponent.
3. Perform addition/subtraction on the mantissas and determine the sign of the result.
4. Normalize the resulting value, if necessary.

Multiplication and division are somewhat easier than addition and subtraction, in that no alignment of mantissas is needed.

15-bit mantissa: $B = 1.M = 1.b_{-1}\, b_{-2}\, b_{-3} \ldots b_{-15}$ (normalized!)

val: $V(B) = 1 + 2^{1}(b_{-1}) + 2^{2}(b_{-2}) + \ldots + 2^{16}(b_{-16})$

range 11 bit exces: $1 \le E \le 2046$ (0 special, 2047 special)

A: 0 | 100001 | 1.111111110 (sign | (33)$_{10}$ | implicit 1.)

B: 0 | 011111 | 1.001010101 → shift ×2

a) Calculation

① B exp ≤ A exp

difference: 33 − 31 = 2

B mantissa shifts right twice

→ 0.001001010 1 = 0.291015625

② larger exponent is 2

A: 1.111111110 ≈ 1.996093750

③ + B: 0.001001010 1 ≈ 0.291015625

≈ 2.287109375 > 2

thus needing normalization

④ shift left ÷ by 2 ≈ 1.143554687 5

correct back → (1).001001001$_2$

real exponent: 31+3 = 34 → 100010$_2$

A+B = 0 100010 001001001

b) value represented: 1.frac × 2^{exp-31}

Largest:

63 − 31 = 32 → 1.111111111

mantissa = $1 + \frac{511}{512} = \frac{1023}{512} \approx 1.998$

Largest = 1.998×2^{32}

Smallest:

0 − 31 = −31 → 1.000000000

Smallest ≈ 1.0×2^{-31}