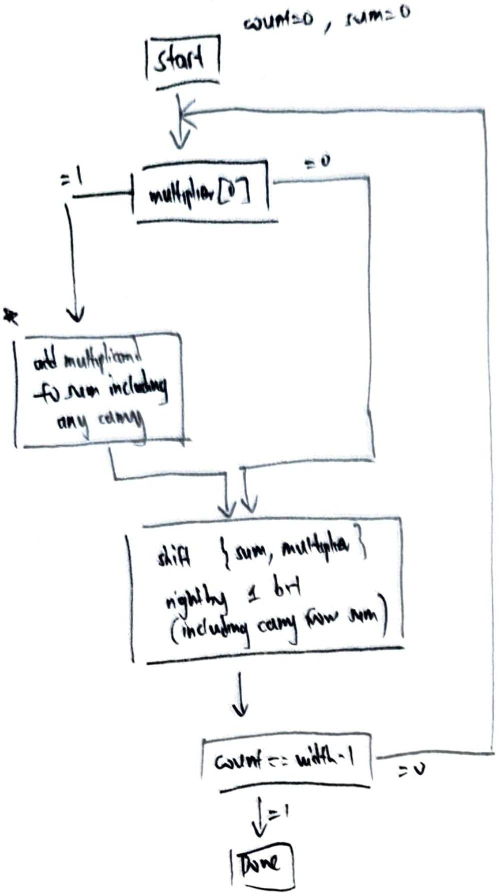
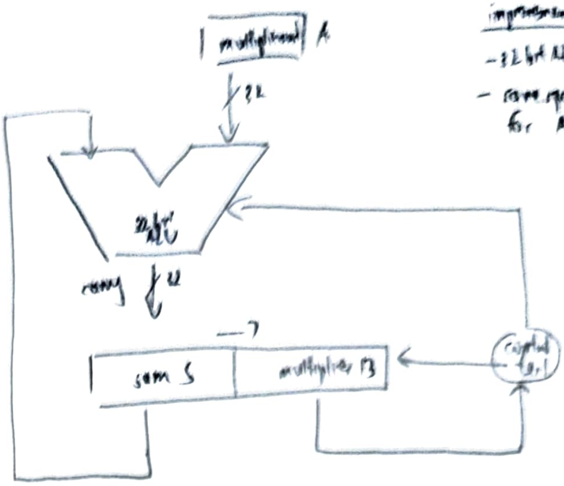
Improved Unsigned Sequential Multiplier

diagram on ch4 pg 20





improvements

- 32 bit ALU vs 64 bit
- same space on registers for A, B, sum

* why carry bit needed after adding multiplicand to sum

multiplicand A = 1111 (15)

multiplier B = 1111 (15)

initial S = 0000 , B = 1111, A = 1111

count

0 — B[0]==1 ⟹ S = S+A = 1111

{carry, S, B} >> 1 ⟹ 0111 | 1111

1 — B[0]==1 ⟹ S = S+A = 0111 + 1111 = $\boxed{1}$0110

{carry, S, B} >> 1 ⟹ 1011 | 0111

2 — B[0]==1 ⟹ S = S+A = 1011 + 1111 = $\boxed{1}$1010

{carry, S, B} >> 1 ⟹ 1101 | 0011

3 — B[0]==1 ⟹ S = S+A = 1101 + 1111 = $\boxed{1}$1100

{carry, S, B} >> 1 ⟹ 11100001

= E1 (225)

* no carry

count

0 — B[0]==1 ⟹ S = S+A = 1111

{S, B} >> 1 ⟹ 0111 | 1111

1 — B[0]==1, ⟹ S = S+A = 0111 + 1111 = 10110

{S, B} >> 1 ⟹ 0011 | 0111

2 — B[0]==1 ⟹ S = S+A = 0011 + 1111 = 10010

{S, B} >> 1 ⟹ 0001 | 0011

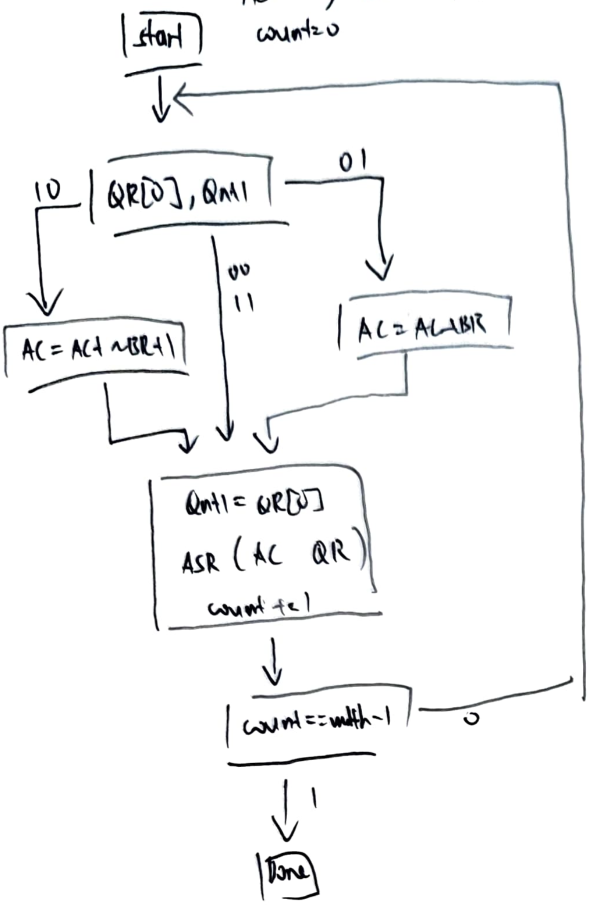
3 — B[0]==1 ⟹ S = S+A = 0001 + 1111 = 10000

{S, B} >> 1 ⟹ 0000 | 0001

= 1 × (not correct)

Booth's Signed Multiplication Algorithm

AC = 0 , BR = { Op1 [width-1] , op1 } , QR = op2 Qm1 = 0
count = 0



* why sign extend BR (multiplicand) & extra bit for AC [width : 0] → still terminate at width - 1

* using [width-1 : 0] fails for most negative number, e.g. -8 when 4 bit width

-8 X 7 = 1100 1000 (-56)

1000 x 0111

AC = 00000 BR = 11000 QR = 0111 Qm1 = 0

count			
0	10	AC = 00000 + 01000 = 01000	Qm1 = 1 QR = 0011 AC = 00100
1	11		Qm1 = 1 QR = 0001 AC = 00010
2	11		Qm1 = 1 QR = 0000 AC = 00001
3	01	AC = 00001 + 11000 = 11001	Qm1 = 0 QR = 1000 AC = 11100

result = 11001000 = -56

* no sign extend

AC = 0000 BR = 1000 QR = 0111 Qm1 = 0

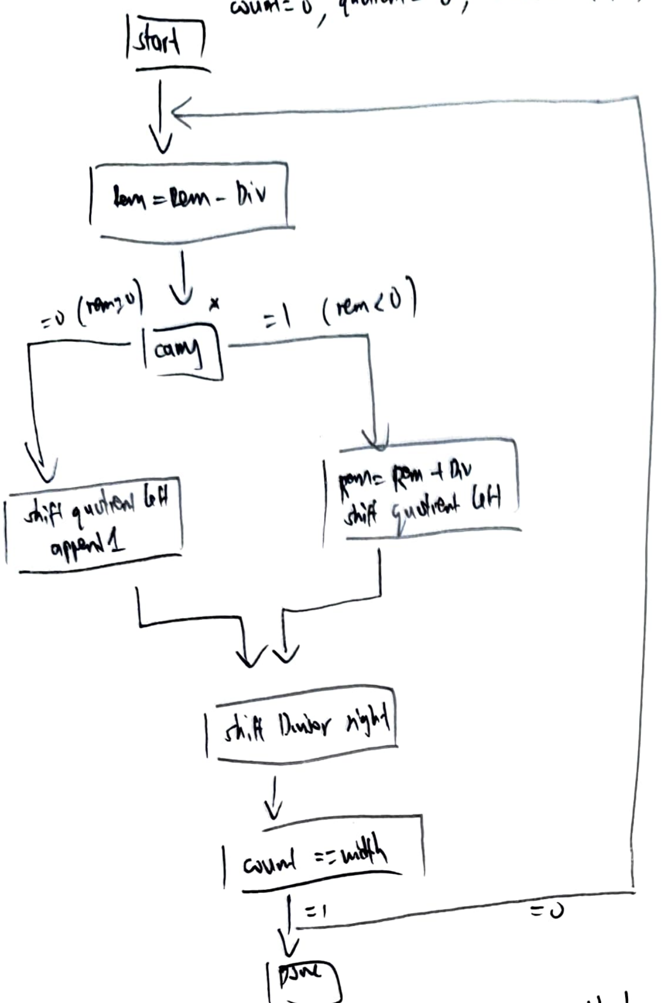
count			
0	10	AC = 0000 + 1000 = 1000	Qm1 = 1 QR = 0011 AC = 1100
1	11		Qm1 = 1 QR = 0001 AC = 1110
2	11		Qm1 = 1 QR = 0000 AC = 1111
3	01	AC = 1111 + 1000 = 0111	Qm1 = 0 QR = 1000 AC = 0011

= 56 X

Restoring Division

algorithm on ch4 pg 23

count = 0, quotient = 0, divisor = {op2, width{0}}, remainder = {width{0}, op1}



* For signed division

If negative, convert to positive using 2's complement

magnitude fits in range of unsigned division, so values for quotient & remainder should have correct magnitude

Finally, correct signs

* Dividend & remainder have same sign
* Quotient is negative if operand have opp sign

→ slides algo misleading testing remainder[2*width-1] for negative...

* why carry bit needed => to test for negative value when divisor is greater than & for unsigned division

15 ÷ 9 = 1 R 6 divisor = 1001 0000 remainder = 0000 1111 quotient = 0000

1111 1001

count

0: 0 00001111 − 0 10010000 => 0 00001111 + 1 01110000 = $\boxed{1}$ 01111111 quotient = 0000 divisor = 01001000

1: 0 00001111 − 0 01001000 => 0 00001111 + 1 10111000 = $\boxed{1}$ 11000111 quotient = 0000 divisor = 00100100

2: 0 00001111 − 0 00100100 => 0 00001111 + 1 11011100 = $\boxed{1}$ 11101011 quotient = 0000 divisor = 00010010

3: 0 00001111 − 0 00010010 => 0 00001111 + 1 11101110 = $\boxed{1}$ 11111101 quotient = 0000 divisor = 00001001

4: 0 00001111 − 0 00001001 => 0 00001111 + 1 11110111 = 1 $\boxed{0}$ 0000 $\boxed{0110}$ quotient = 0001 remainder = 0110

* no carry

count

0: 00001111 − 10010000 => 00001111 + 01110000 = $\boxed{0}$ 1111111 quotient = 0001

1: 01111111 − 01001000 => 01111111 + 10111000 = $\boxed{1}$ 0110111 quotient = 0011

2: 00110111 − 00100100 => 00110111 + 11011100 = $\boxed{1}$ 00010011 quotient = 0111

3: 00010011 − 00010010 => 00010011 + 11101110 = 1 $\boxed{0}$ 0000001 quotient = 1111

4: 00000001 − 00001001 => 00000001 + 11110111 = $\boxed{1}$ 1111000 quotient = 1110 remainder = 1000 ✗