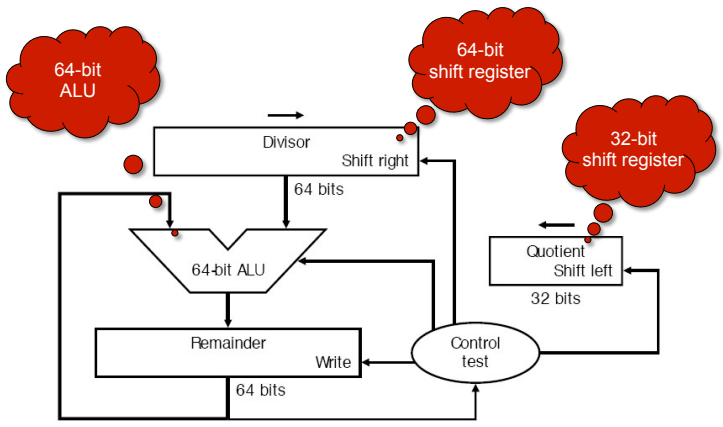
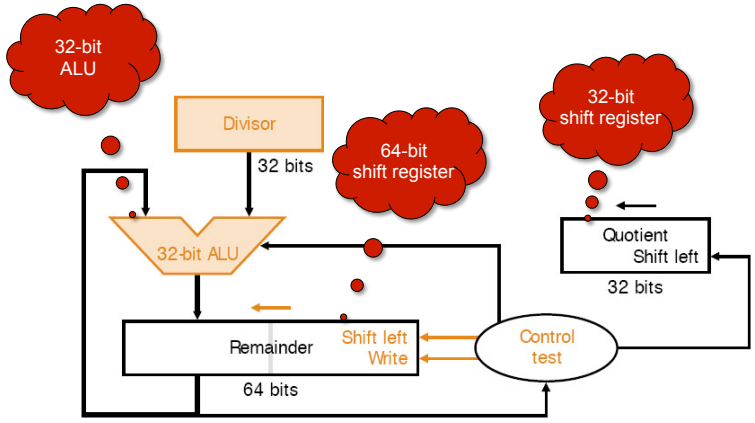
Binary division

- quotient = dividend / divisor, with a remainder
- dividend = divisor × quotient + remainder
- Given dividend and divisor, we want to obtain quotient (Q) and remainder (R)
- We will start from our paper & pencil method

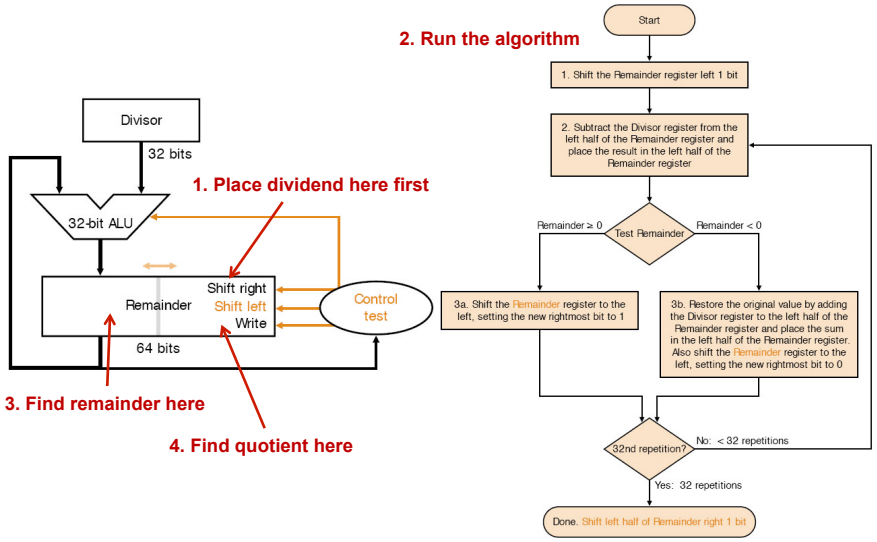
Hardware design 1



Hardware design 2



Hardware design 3



Example

- Let's do 0111/0010 (7/2) – unsigned

Iteration	Divisor	Hardware design 3	
		Step	Remainder
0	0010	initial values	0000 0111
		shift remainder left by 1	0000 1110
1	0010	remainder = remainder – divisor	1110 1110
		(remainder<0) ⇒ +divisor; shift left; r0=0	0001 1100
2	0010	remainder = remainder – divisor	1111 1100
		(remainder<0) ⇒ +divisor; shift left; r0=0	0011 1000
3	0010	remainder = remainder – divisor	0001 1000
		(remainder>0) ⇒ shift left; r0=1	0011 0001
4	0010	remainder = remainder – divisor	0001 0001
		(remainder>0) ⇒ shift left; r0=1	0010 0011
done	0010	shift "left half of remainder" right by 1	0001 0011

Restoring division

- The three hardware designs we saw are based on the notion of "restoring division"
  - At first, attempt to subtract divisor from dividend
  - If the result of subtraction is negative – it rolls back by adding divisor
    - This step is called "restoring"
- It's a "trial-and-error" approach; can we do better?

Non-restoring division

- Let's revisit the restoring division designs
  - Given remainder R (R<0) after subtraction
  - By adding divisor D back, we have (R+D)
  - After shifting the result, we have 2×(R+D)=2×R+2×D
  - If we subtract the divisor in the next step, we have 2×R+2×D–D =2×R+D
- This is equivalent to
  - Left-shifting R by 1 bit and then adding D!
- There's a "special case" when you finish with a negative
  - The negative value must be restored (add D back to R)

Example, non-restoring division

- Let's again do 0111/0010 (7/2) – unsigned

Iteration	Divisor	Hardware design 3, non-restoring	
		Step	Remainder
0	0010	initial values	0000 0111
		shift remainder left by 1	0000 1110
1	0010	remainder = remainder – divisor	1110 1110
		(remainder<0) ⇒ shift left; r0=0	1101 1100
2	0010	remainder = remainder + divisor	1111 1100
		(remainder<0) ⇒ shift left; r0=0	1111 1000
3	0010	remainder = remainder + divisor	0001 1000
		(remainder>0) ⇒ shift left; r0=1	0011 0001
4	0010	remainder = remainder – divisor	0001 0001
		(remainder>0) ⇒ shift left; r0=1	0010 0011
done	0010	shift "left half of remainder" right by 1	0001 0011