COL216
Computer Architecture

Divider Design

8th February, 2016

Outline of this lecture

- Division example
- Basic division algorithm
- Proof of correctness
- Circuit design
- Improvements

Division: example

```
              0011   ←  Q
          ________
0100 | 00001101   ←  A
  ↑     0000000   ←———  0 x B x 2^3
  B    00001101
         000000   ←———  0 x B x 2^2
       00001101
          01000   ←———  1 x B x 2^1
       00000101
           0100   ←———  1 x B x 2^0
       00000001   ←  R
```

$0 \times B \times 2^3$, $0 \times B \times 2^2$, $1 \times B \times 2^1$, $1 \times B \times 2^0$

Unsigned Division

$$A = Q \times B + R$$

step1: $i = 0$; $R = A$; $Q = 0$; $D = B$

do {

  step2:

    if ($D \times 2^{n-i-1} \leq R$) $R = R - D \times 2^{n-i-1}$; $Q_{n-i-1} = 1$

    else $Q_{n-i-1} = 0$

    i ++

} while ($i < n$)

Correctness check

$$\mathit{Loop}\ \text{invariant}:\ A = Q \times D + R\ \mathit{and}\ 0 \le R < B \times 2^{n-i}$$

At the end of the program, i has a value = n and D is still = B

For i = n, the loop invariant $\Rightarrow$ $0 \le R < B$

Therefore R is the correct remainder and Q is the correct quotient

Base case

$$Loop\ \text{invariant}:\ A = Q \times D + R\ and\ 0 \le R < B \times 2^{n-i}$$

step1: $i = 0$; $R = A$; $Q = 0$; $D = B$

This ensures the truth of the loop invariant initially, provided $A < B \times 2^n$

This condition is met if we assume that $B \neq 0$ and A is contained within n bits.

Induction over i (case 1)

$Loop\ \text{invariant}: \ A = Q \times D + R\ and\ 0 \leq R < B \times 2^{n-i}$

Assume invariant holds before iteration i and $D \times 2^{n-i-1} \leq R$

New value of Q x D + R is

$(Q + 2^{n-i-1}) \times D + R - D \times 2^{n-i-1} = Q \times D + R$

The condition of subtraction ensures that

$D \times 2^{n-i-1} \leq R < D \times 2^{n-i}$,

$\therefore$ after subtraction, $0 \leq R < D \times 2^{n-i-1} = B \times 2^{n-i-1}$

i now becomes $i' = i+1$, $\therefore\ 0 \leq R < B \times 2^{n-i'}$

Induction over i (case 2)

$$Loop \text{ invariant}: \; A = Q \times D + R \text{ and } 0 \le R < B \times 2^{n-i}$$

Assume invariant holds before iteration i and

$D \times 2^{n-i-1} > R$

New value of Q x D + R is

$(Q + 0) \times D + R = Q \times D + R$

The condition of omitting subtraction ensures that $0 \le R < D \times 2^{n-i-1}$

Therefore, $i' = i+1$, $0 \le R < B \times 2^{n-i'}$

Unsigned Division

$$A = Q \times B + R$$

step1: i = 0; R = A; Q = 0; D = B

do {

step2:

if (D x 2^{n-i-1} ≤ R) R = R - D x 2^{n-i-1}; Q_{n-i-1} = 1

else Q_{n-i-1} = 0

i ++

} while (i < n)

can this be avoided ?

Introducing shift registers

$$A = Q \times B + R$$

step1: $i = 0$; $R = A$; $Q = 0$; $D = B \times 2^{n-1}$

do {

  step2:

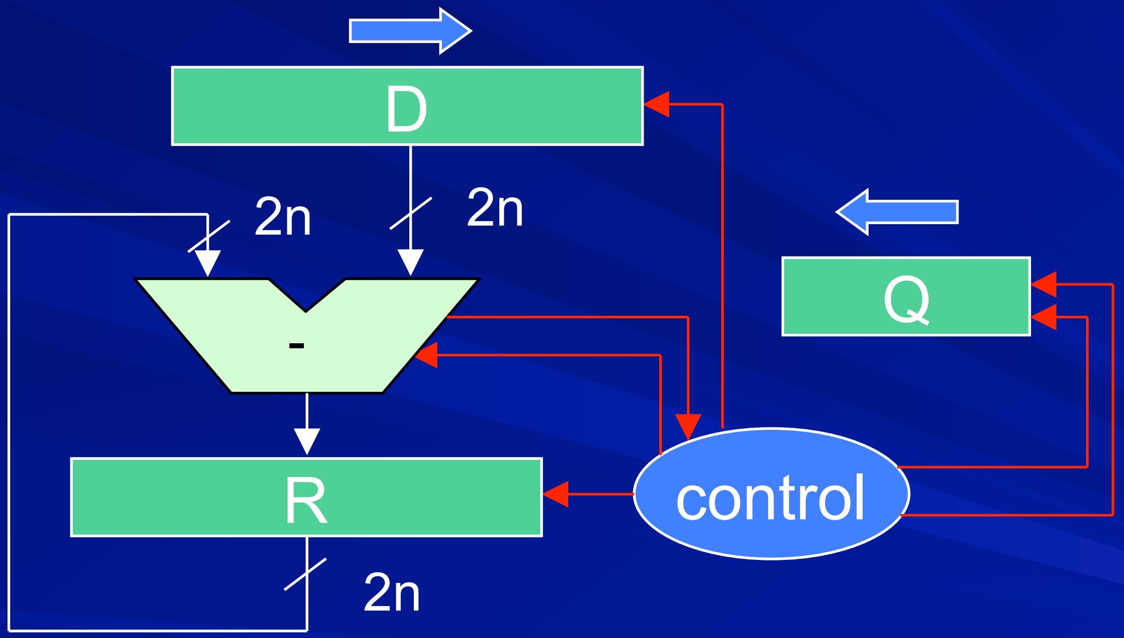
    if ($D \leq R$) $R = R - D$; $Q = 2 \times Q + 1$

    else $Q = 2 \times Q$

    $D = D / 2$; i ++

} while ($i < n$)

Divider design - 1



Reducing subtractor size

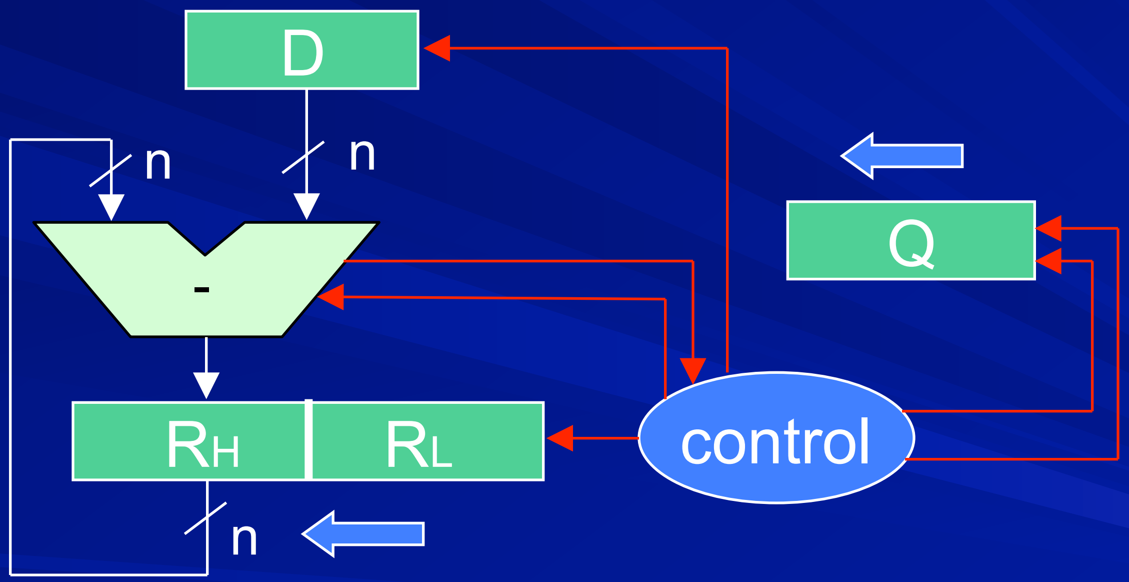
$$A = Q \times B + R$$

```
step1: i = 0; R = A; Q = 0; D = B
do {
  step2:
    R = 2 x R
    if (D ≤ R_H) R_H = R_H - D; Q = 2 x Q + 1
   else Q = 2 x Q
    i ++
} while (i < n)
```

Division: example

```
          0011   ← Q
     ____________
0100 | 00001101  ← A
  ↑    00011010
  B    0000      ← 0 x B
       00011010
       00110100
       0000      ← 0 x B
       00110100
       01101000
       0100      ← 1 x B
       00101000
       01010000
       0100      ← 1 x B
       00010000  ← R
```

Divider design - 2



Reducing registers

$$A = Q \times B + R$$

step1: i = 0; R = A; D = B

do {

  step2:

    R = 2 x R

    if ($D \leq R_H$) $R_H = R_H - D$; R = R + 1

    i ++

} while (i < n)   # R_H = remainder, R_L = quotient

Divider design - 3