M3 – ALU Design

Module Outline

- Integer Arithmetic
  - Adder, Subtractor, Multiplier, Divider
- Arithmetic and Logical Unit Design
  - ALU Design in SystemC

Multiplication

- Start with long-multiplication approach

```
    1000
×   1001
------
    1000
   0000
  0000
 1000
------
 1001000
```

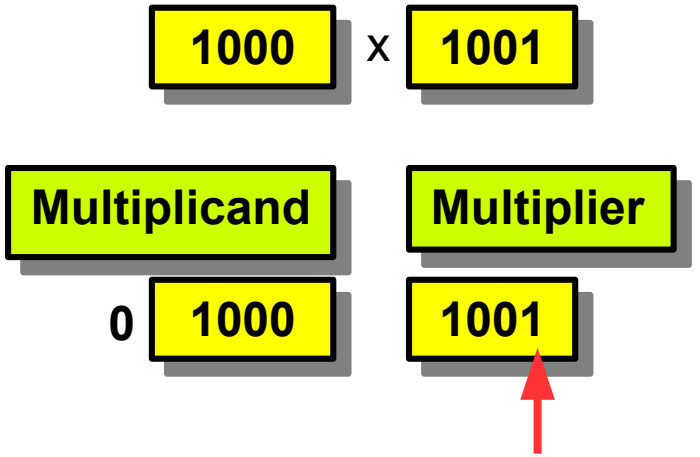
Multiplication

- Start with long-multiplication approach

1000 x **1001**

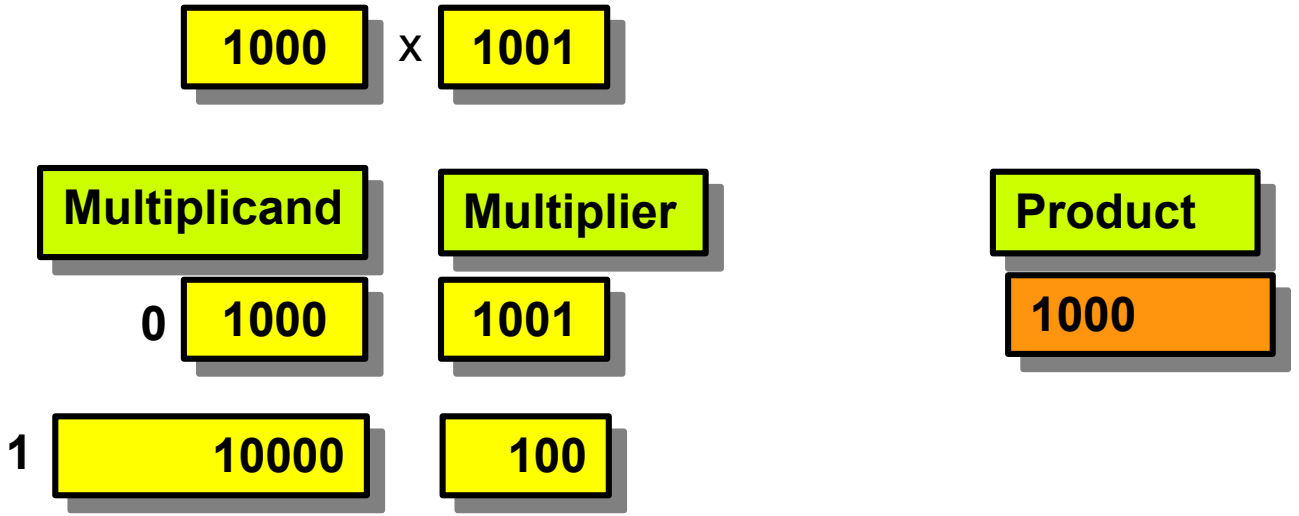
Multiplication

- Start with long-multiplication approach



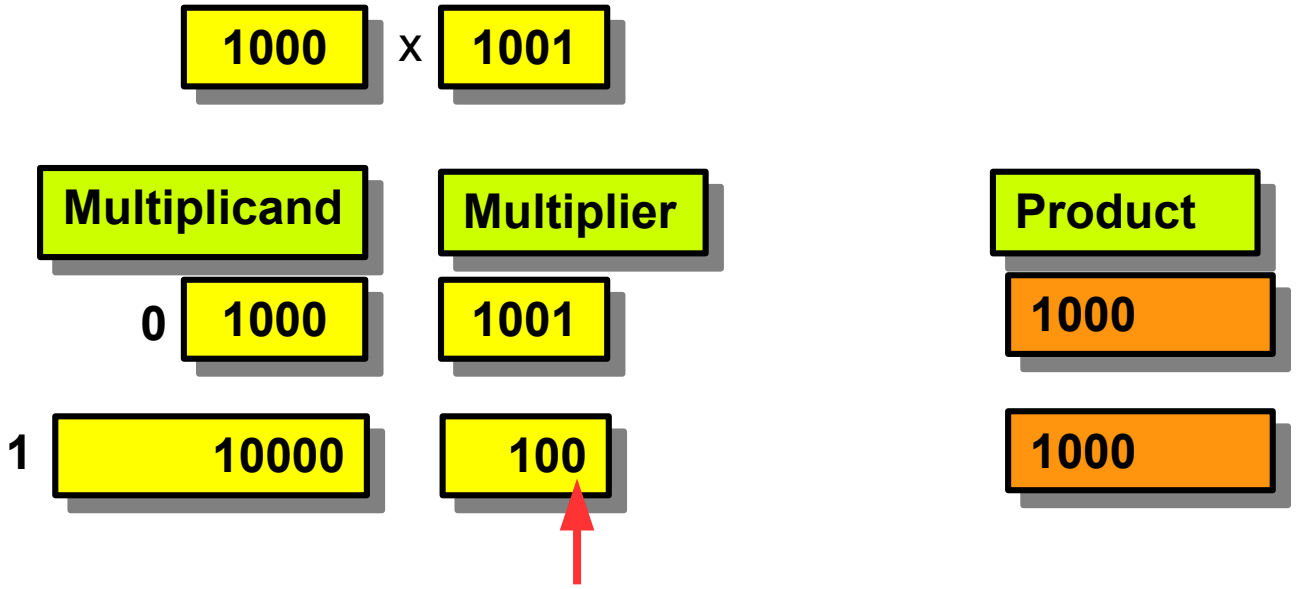
Multiplication

- Start with long-multiplication approach



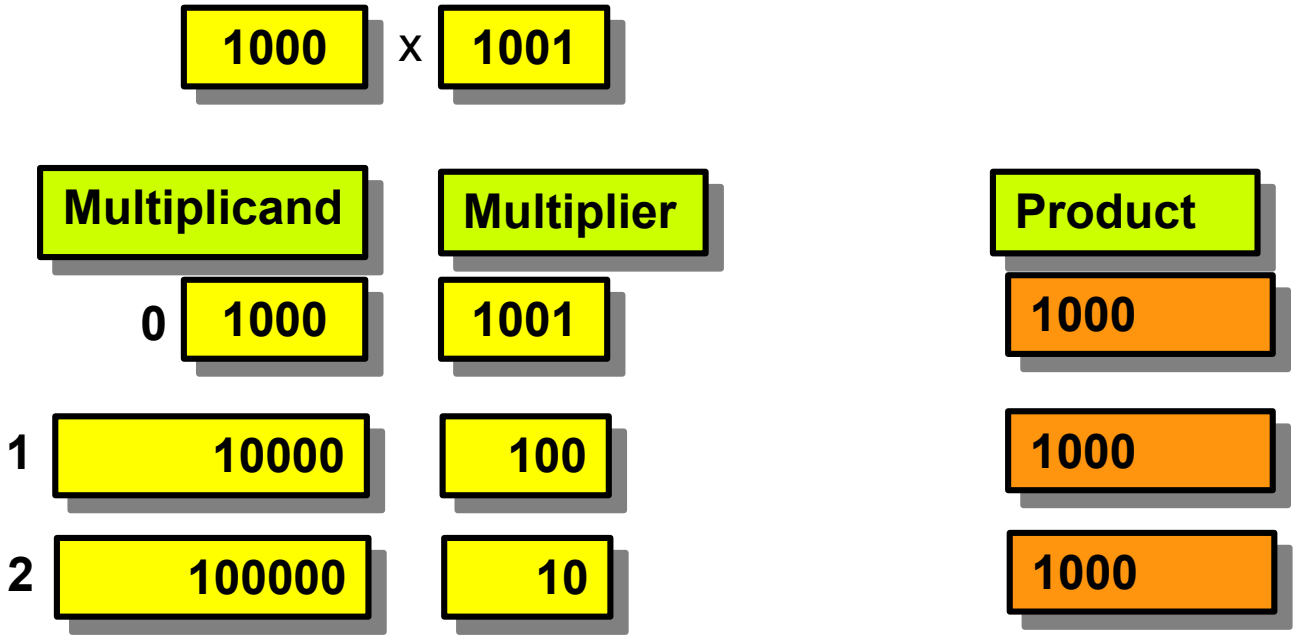
Multiplication

- Start with long-multiplication approach



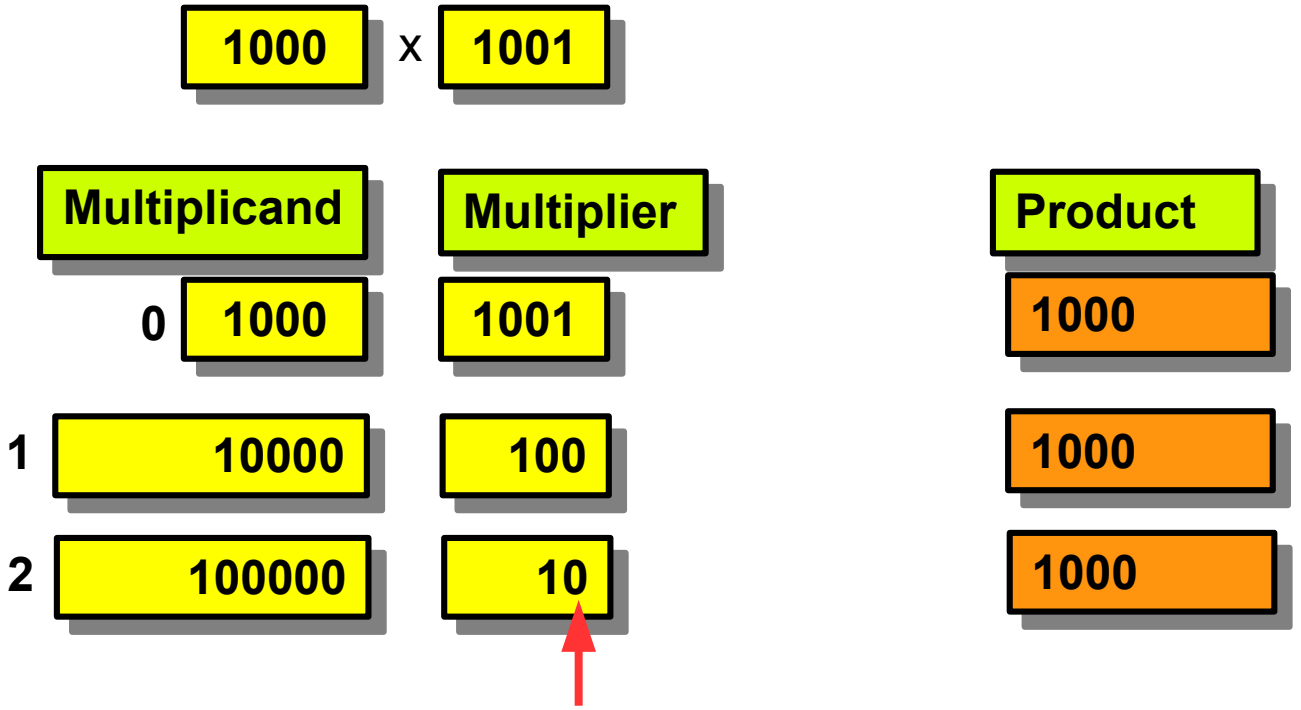
Multiplication

- Start with long-multiplication approach



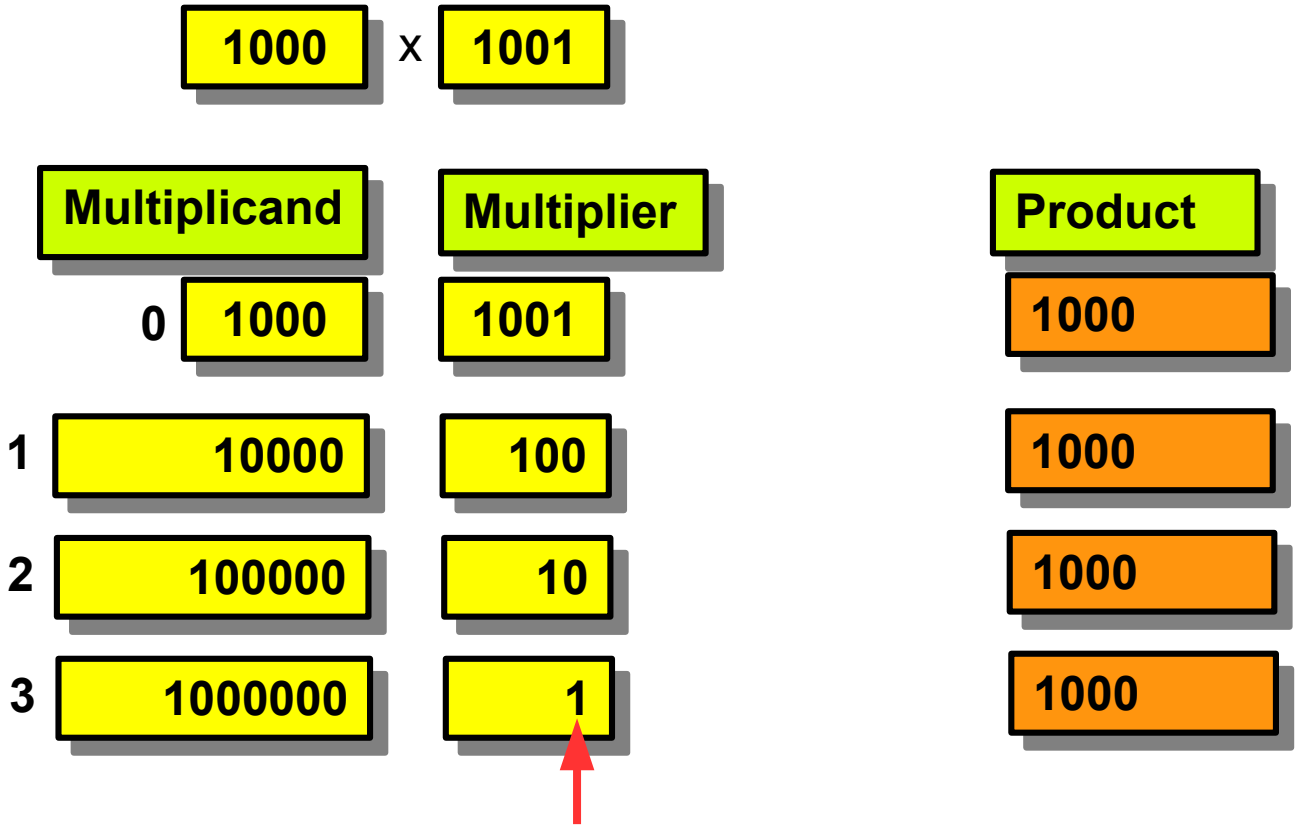
Multiplication

- Start with long-multiplication approach



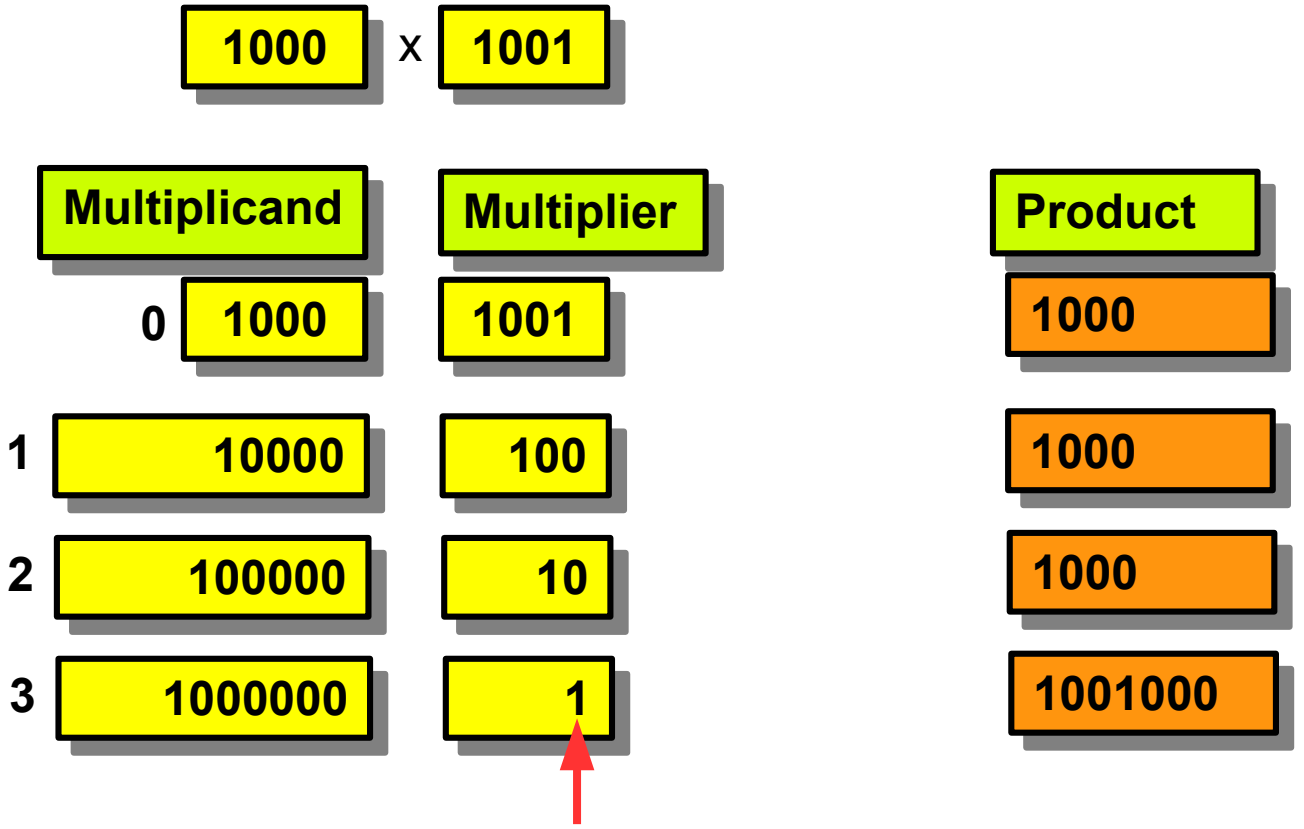
Multiplication

- Start with long-multiplication approach



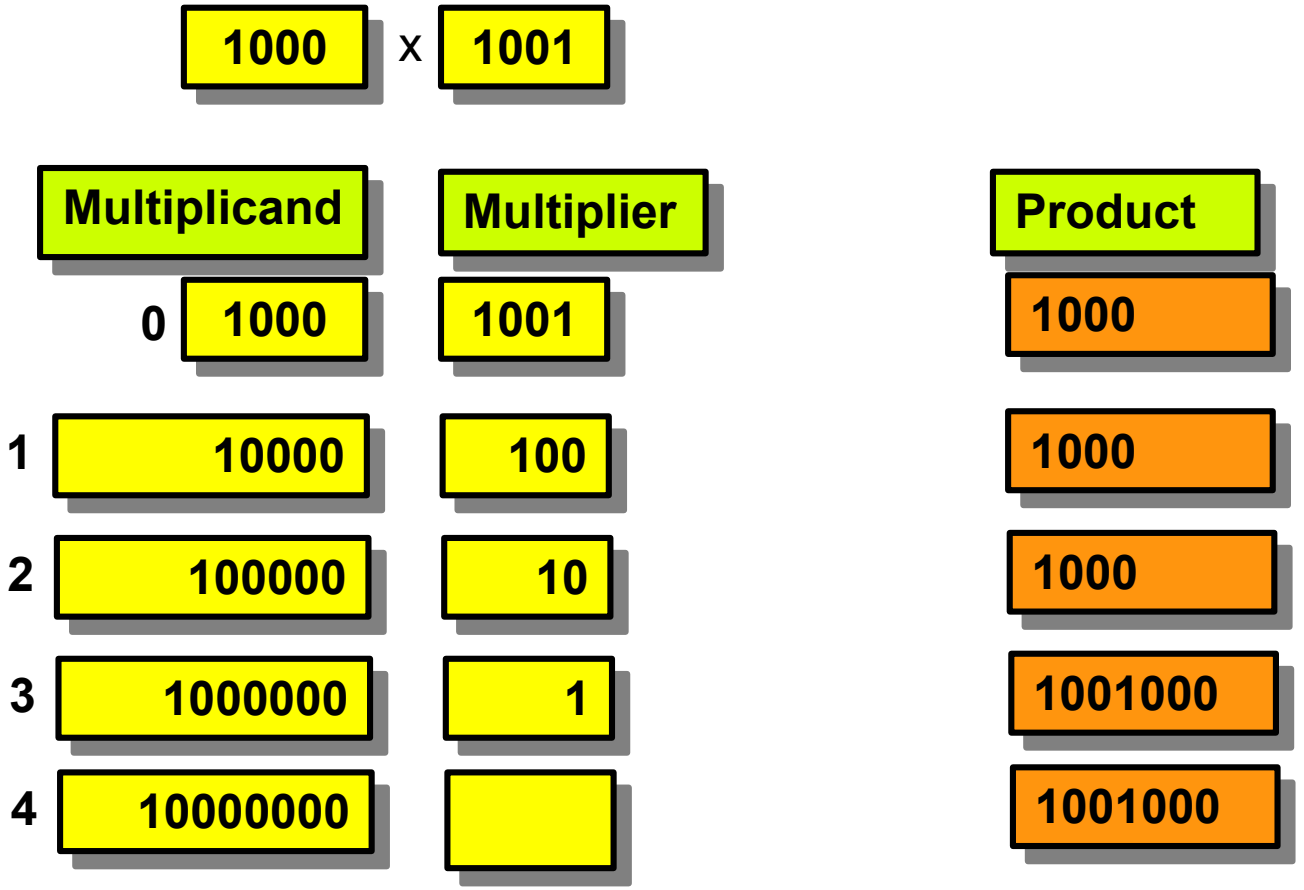
Multiplication

- Start with long-multiplication approach



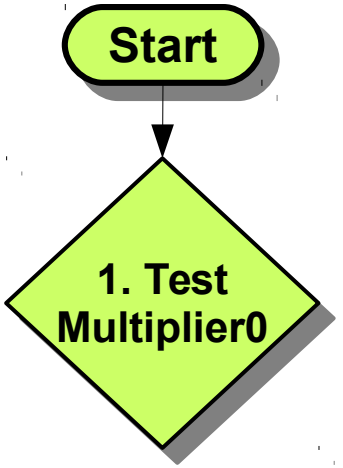
Multiplication

- Start with long-multiplication approach

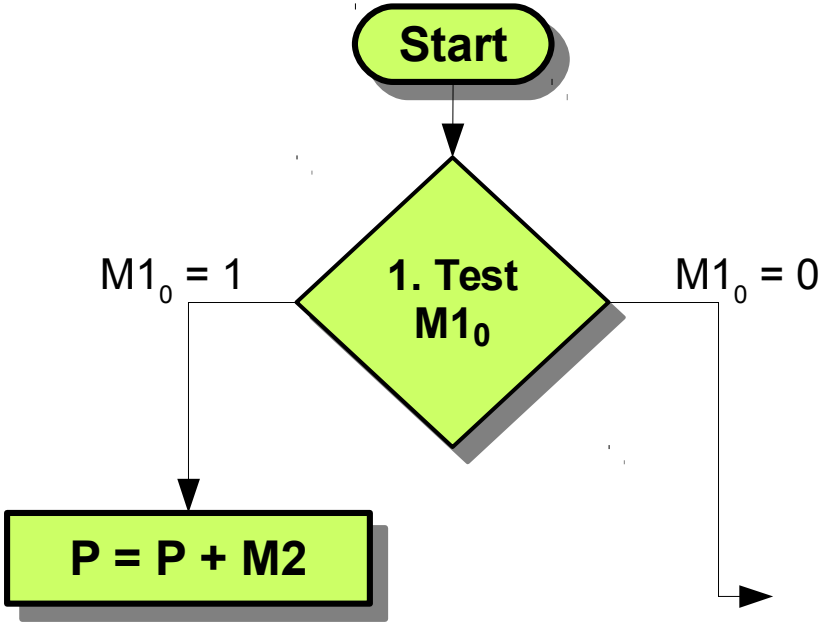


Multiplication Hardware

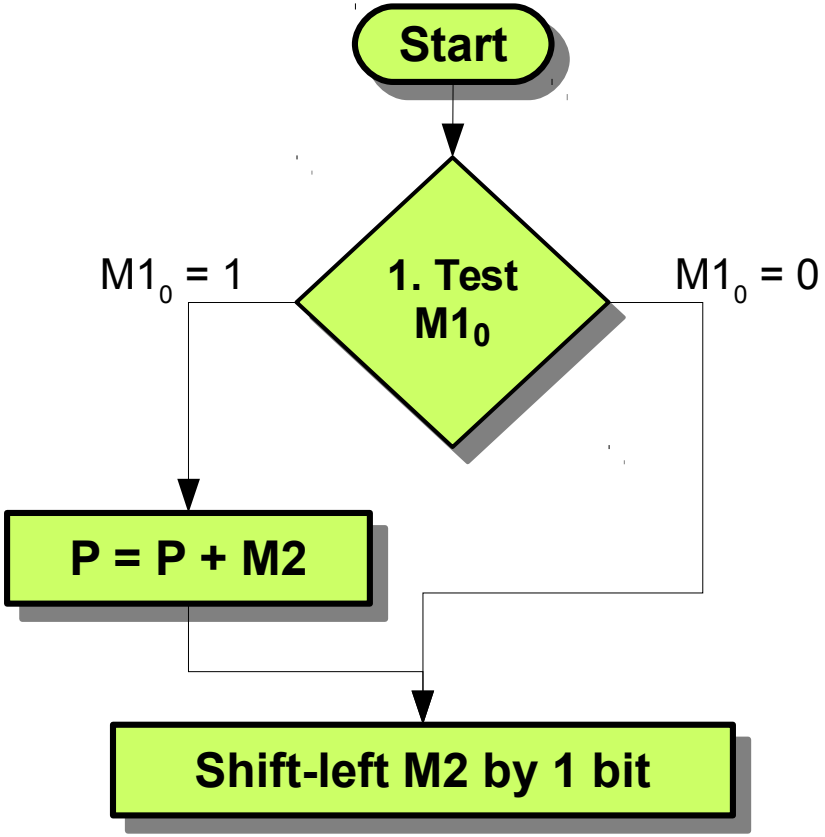
Multiplication Hardware



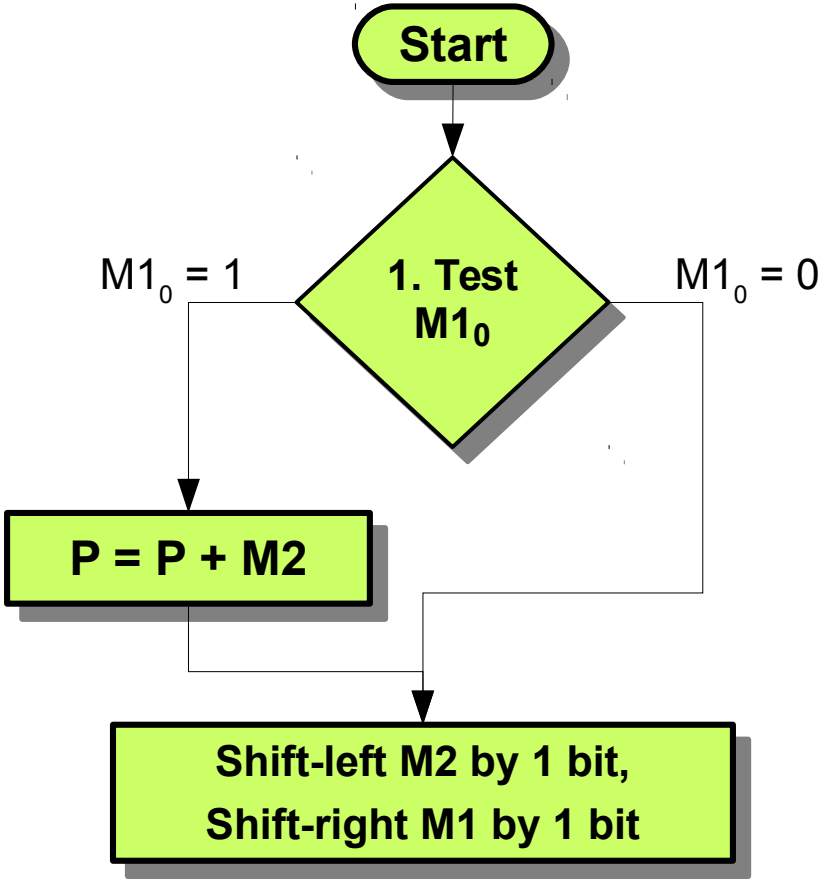
Multiplication Hardware



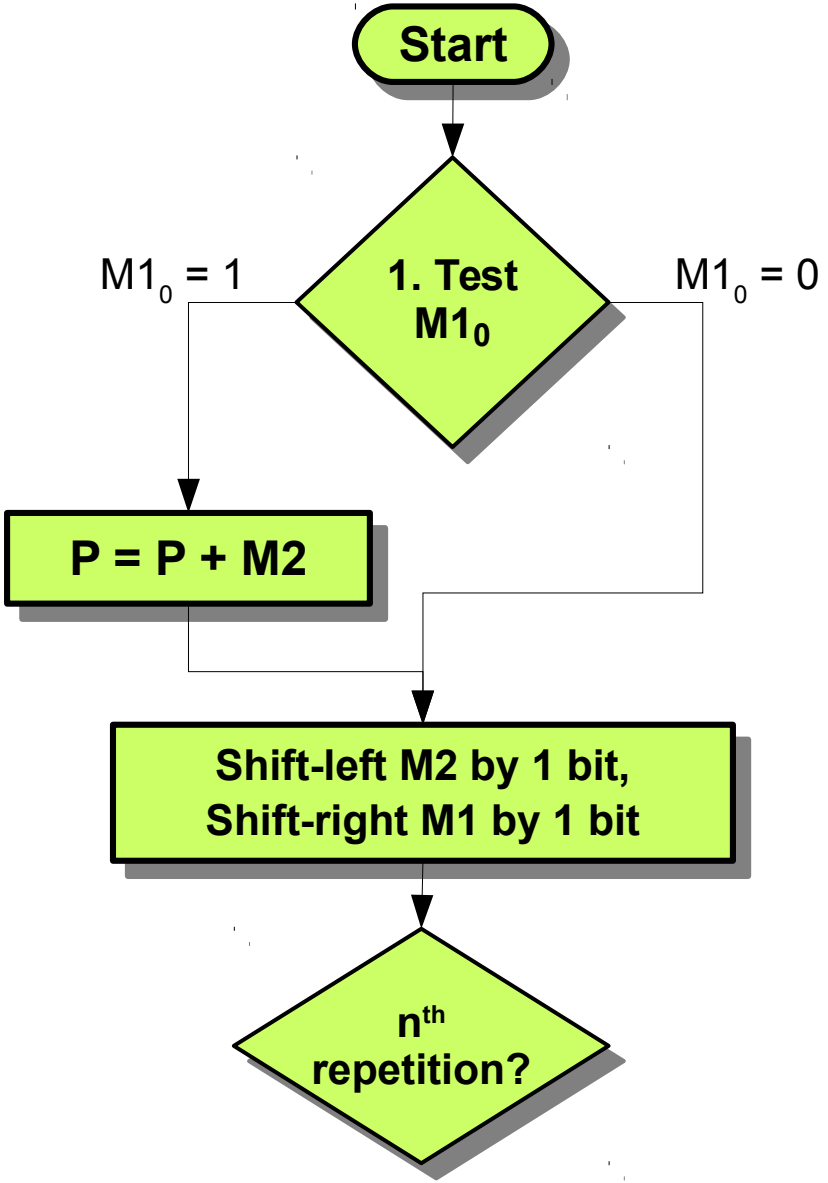
Multiplication Hardware



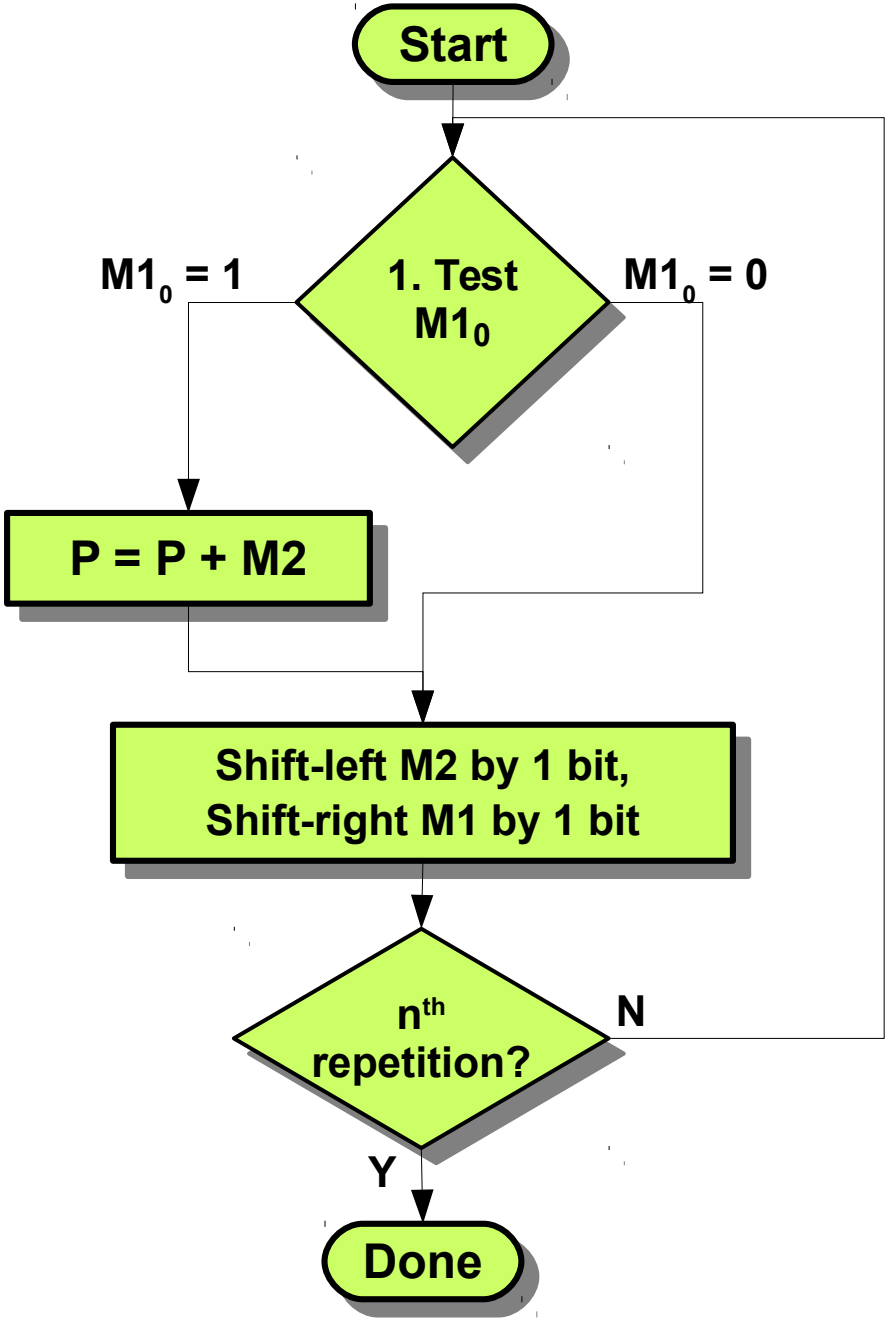
Multiplication Hardware



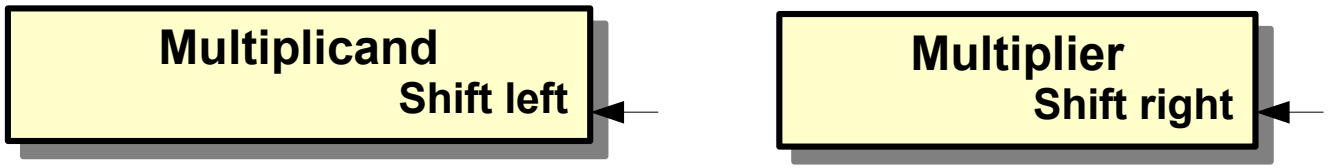
Multiplication Hardware



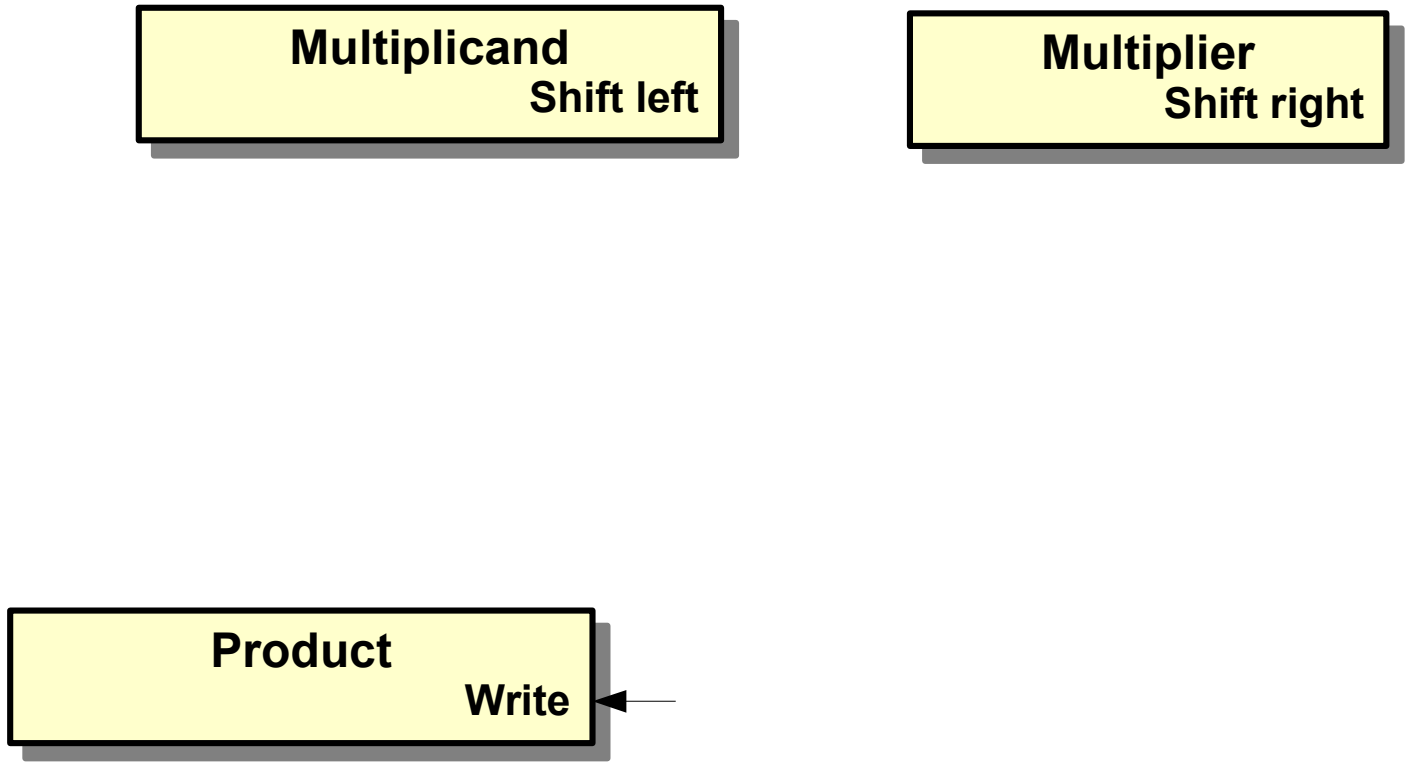
Multiplication Hardware



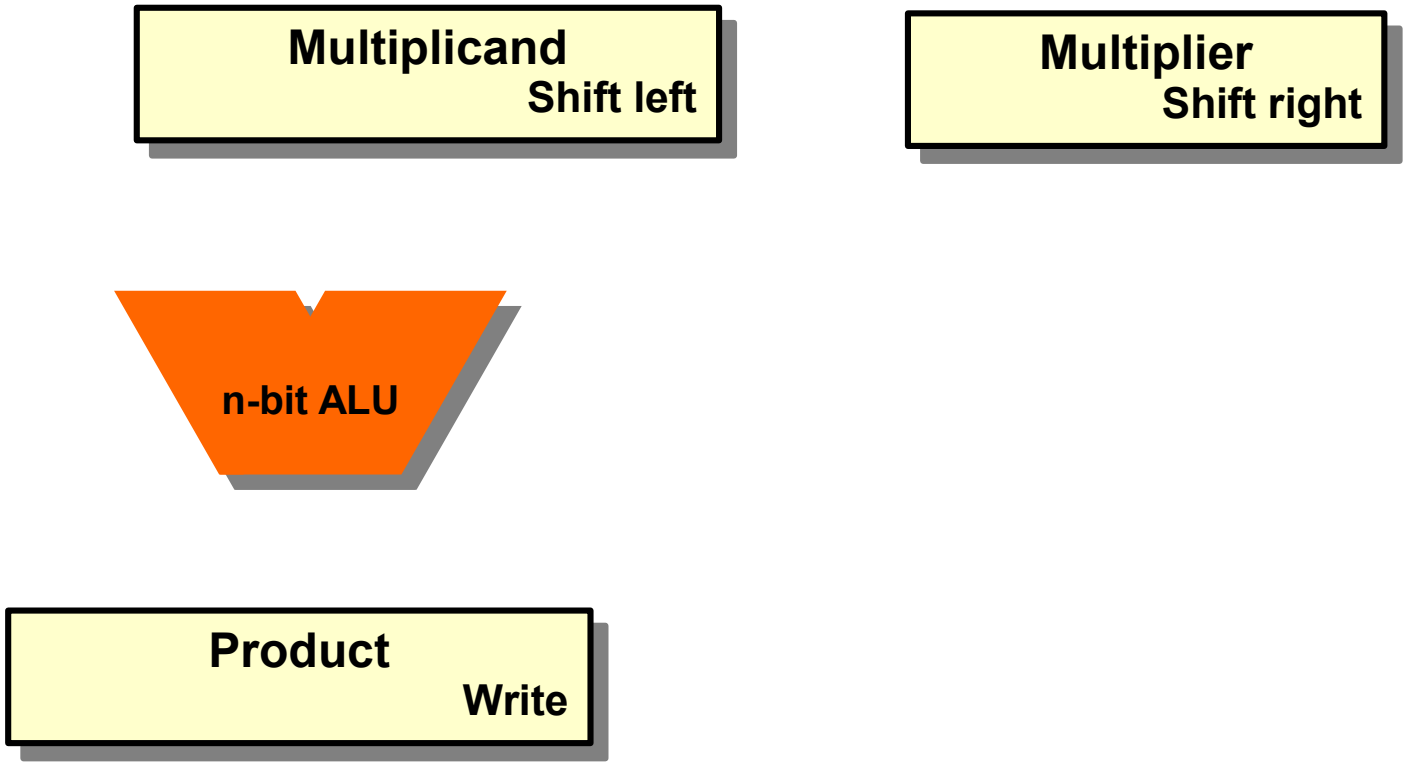
Multiplication Hardware



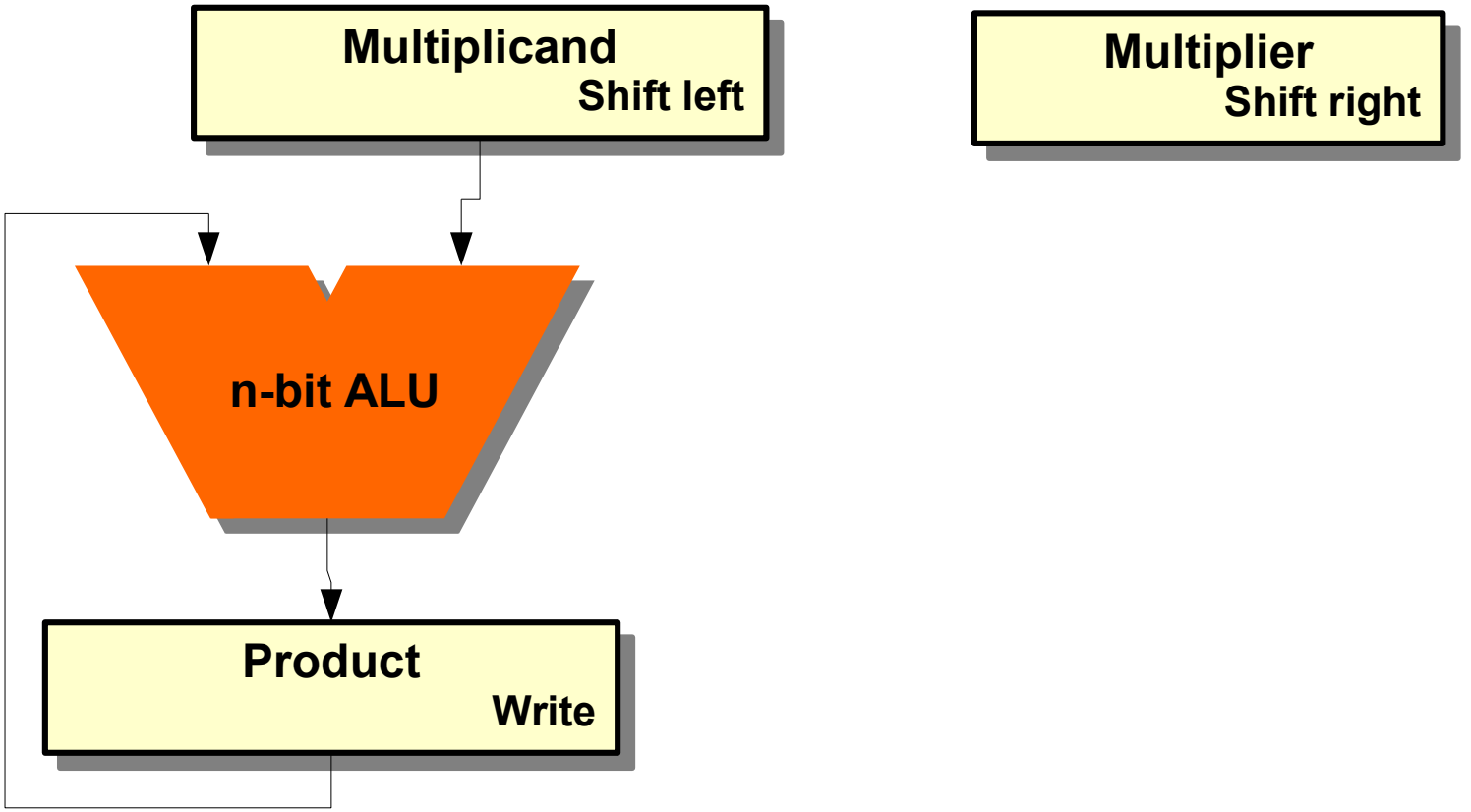
Multiplication Hardware



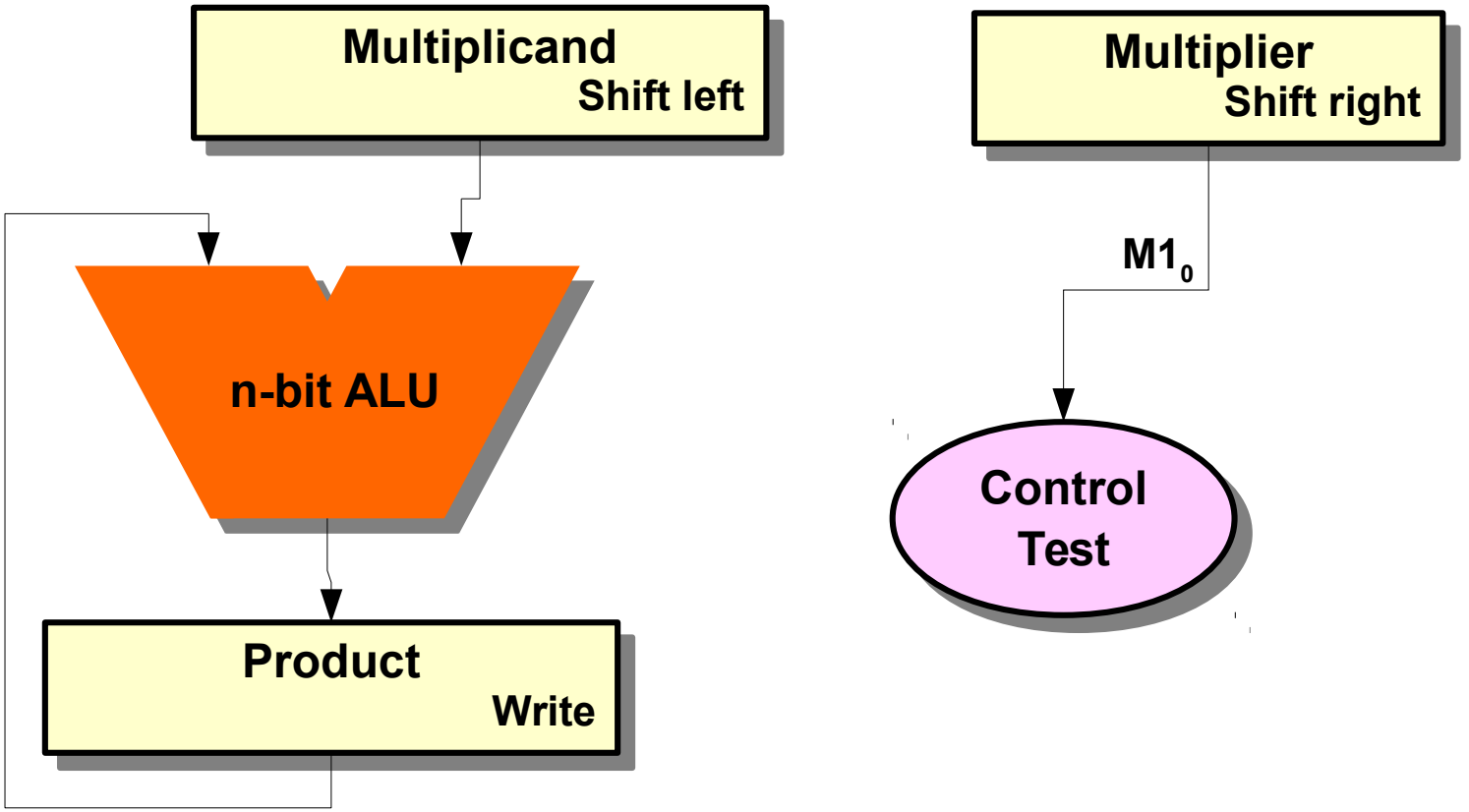
Multiplication Hardware



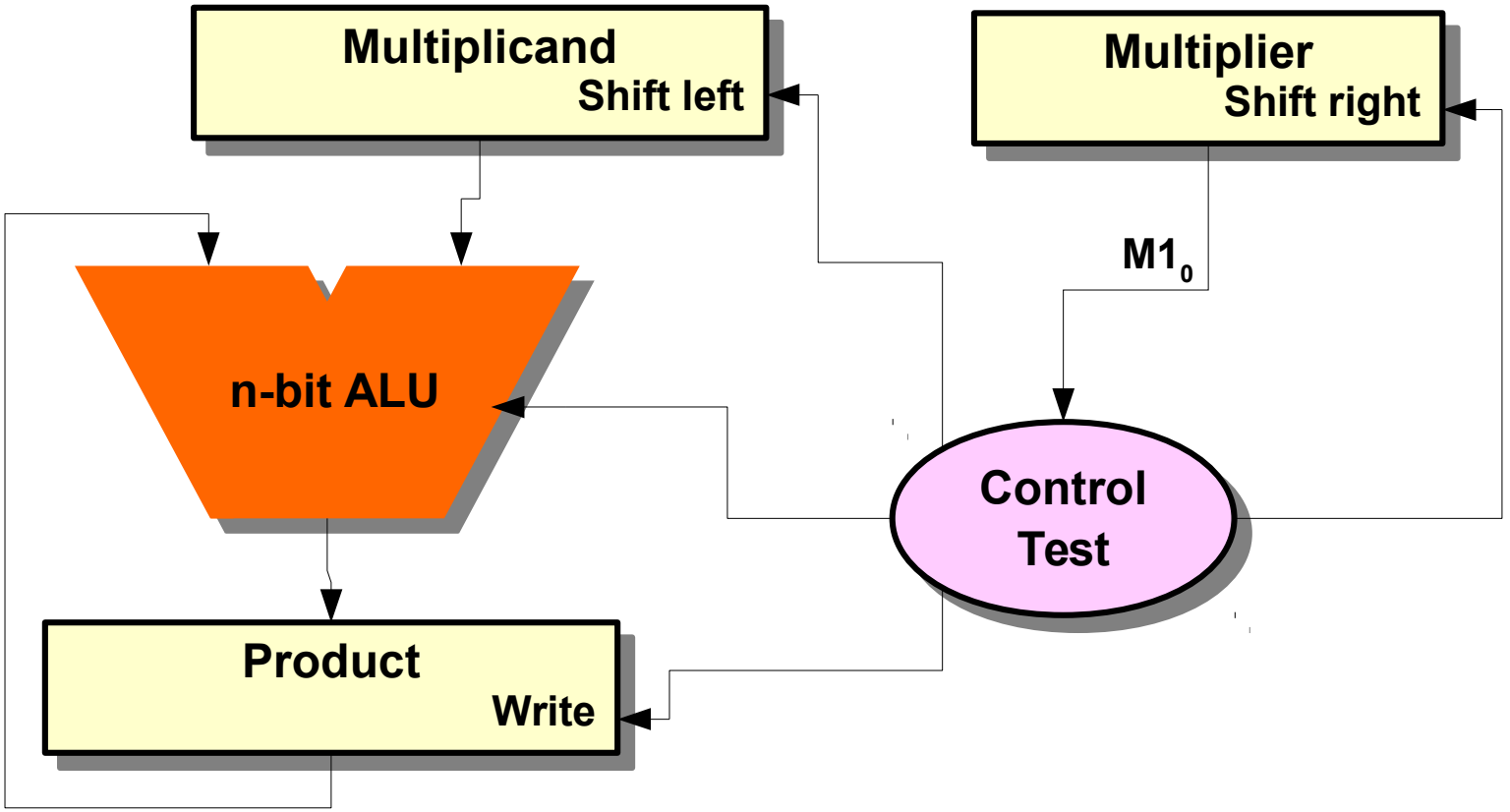
Multiplication Hardware



Multiplication Hardware



Multiplication Hardware



MIPS Multiplication

- Two 32-bit registers for product
  - HI: most-significant 32 bits
  - LO: least-significant 32-bits

MIPS Multiplication

- Two 32-bit registers for product
  - HI: most-significant 32 bits
  - LO: least-significant 32-bits
- Instructions
  - `mult rs, rt  /  multu rs, rt`
    - 64-bit product in HI/LO

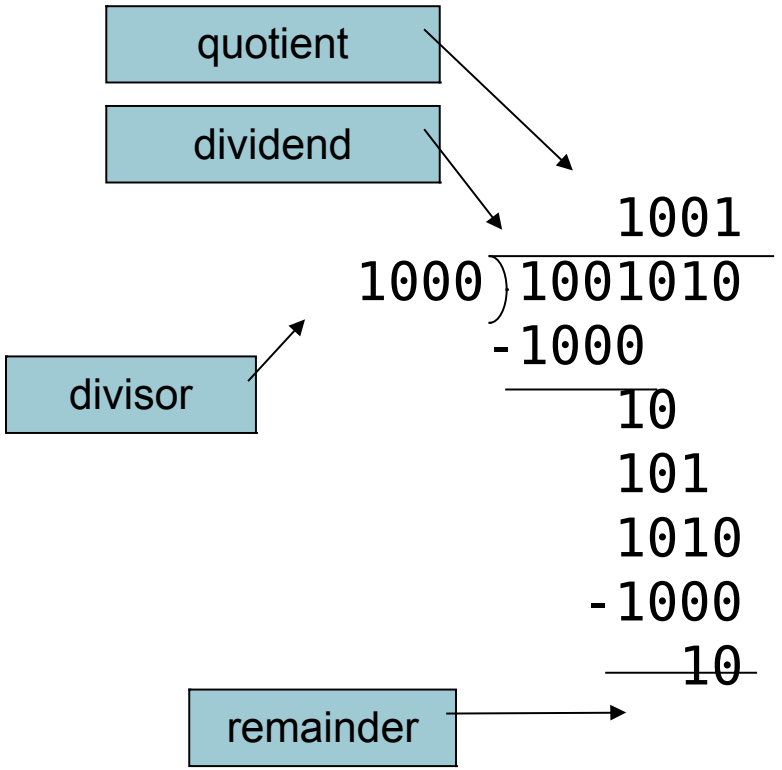
MIPS Multiplication

- Two 32-bit registers for product
  - HI: most-significant 32 bits
  - LO: least-significant 32-bits
- Instructions
  - `mult rs, rt  /  multu rs, rt`
    - 64-bit product in HI/LO
  - `mfhi rd  /  mflo rd`
    - Move from HI/LO to rd
    - Can test HI value to see if product overflows 32 bits

MIPS Multiplication

- Two 32-bit registers for product
  - HI: most-significant 32 bits
  - LO: least-significant 32-bits
- Instructions
  - `mult rs, rt  /  multu rs, rt`
    - 64-bit product in HI/LO
  - `mfhi rd  /  mflo rd`
    - Move from HI/LO to rd
    - Can test HI value to see if product overflows 32 bits
  - `mul rd, rs, rt`
    - Least-significant 32 bits of product –> rd

Division

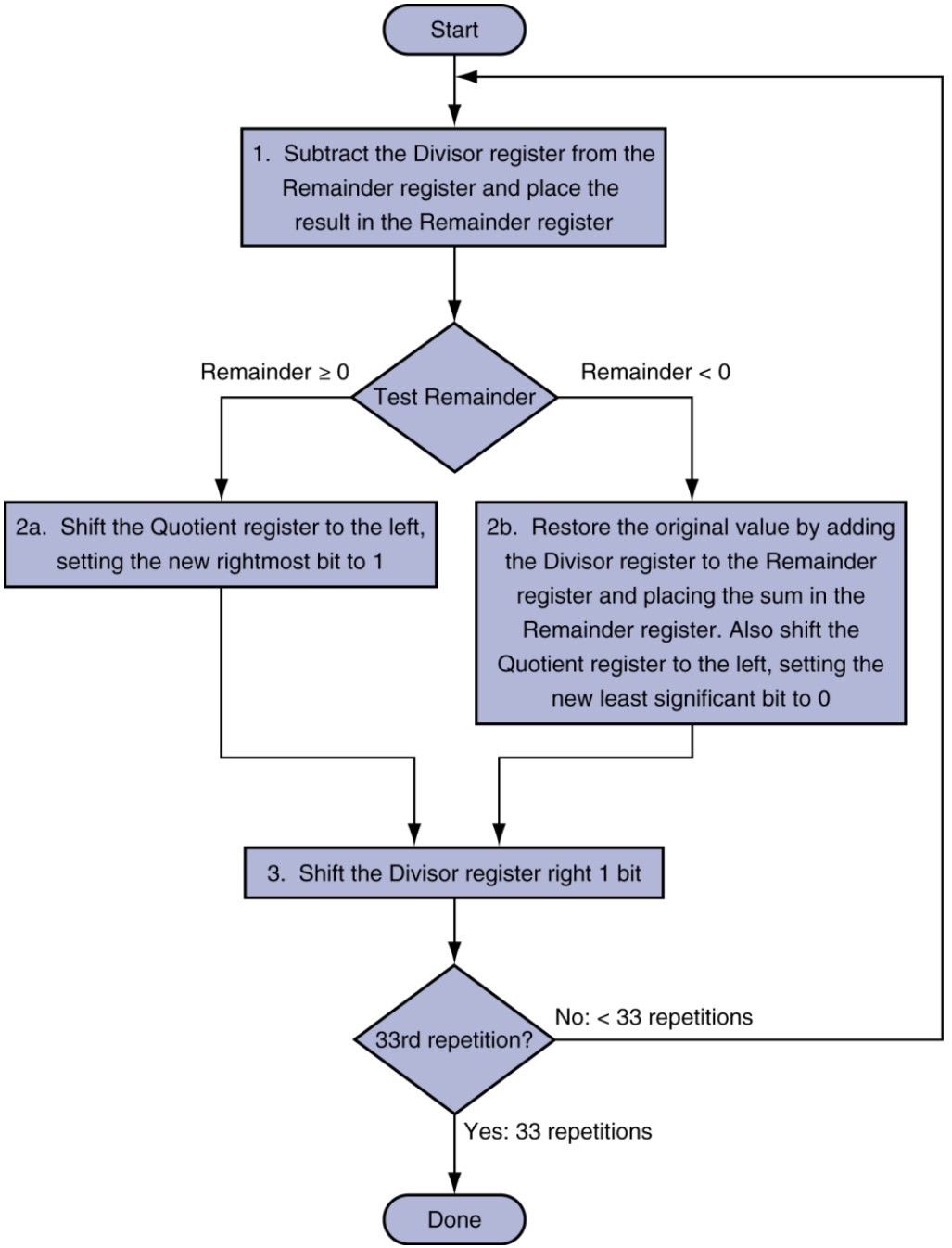


n-bit operands yield *n*-bit quotient and remainder

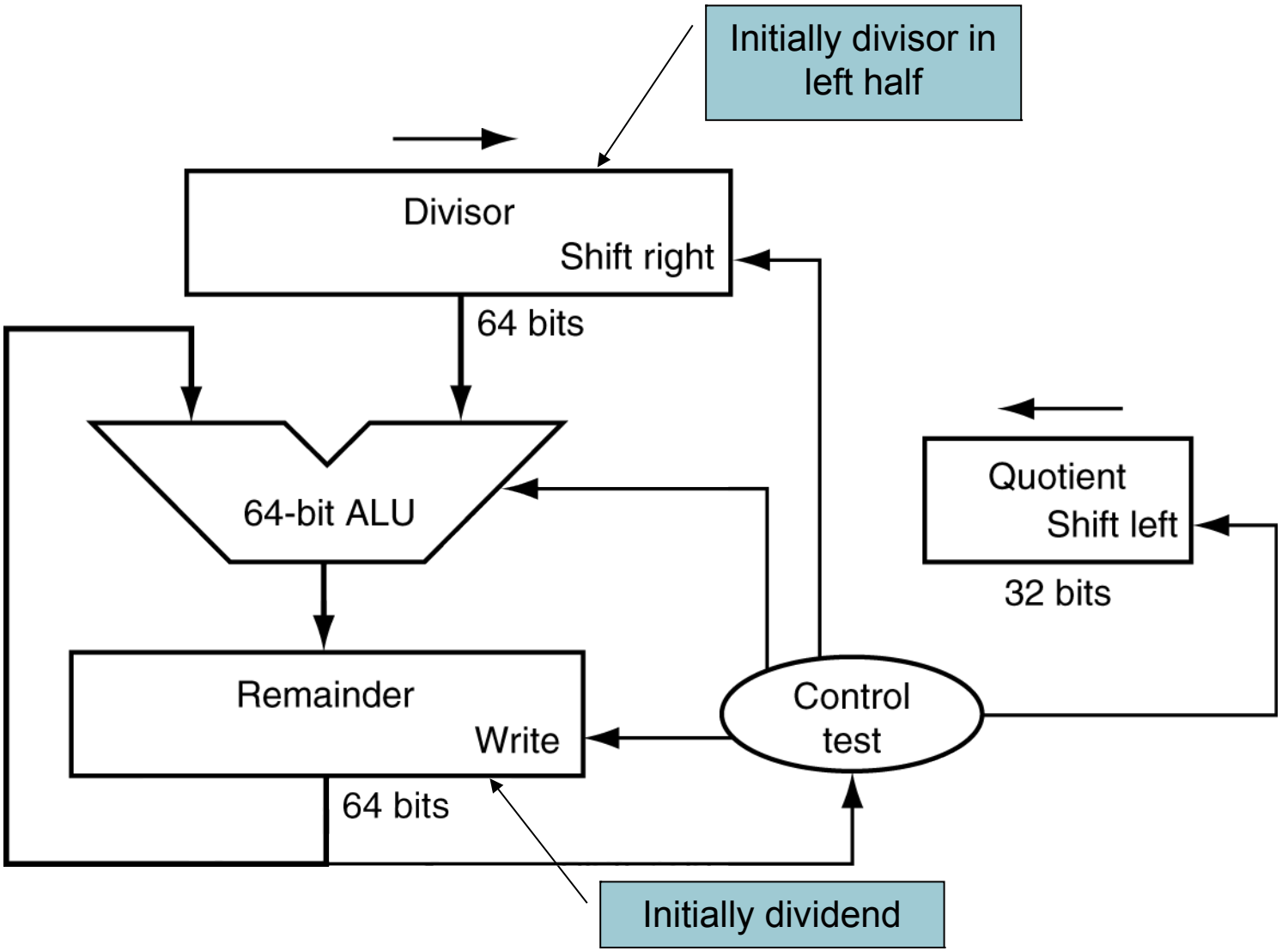
Division

- Check for 0 divisor
- Long division approach
  - If divisor ≤ dividend bits
    - 1 bit in quotient, subtract
  - Otherwise
    - 0 bit in quotient, bring down next dividend bit
- Restoring division
  - Do the subtract, and if remainder goes < 0, add divisor back
- Signed division
  - Divide using absolute values
  - Adjust sign of quotient and remainder as required

Division Hardware



Division Hardware



MIPS Division

- Use HI/LO registers for result
  - HI: 32-bit remainder
  - LO: 32-bit quotient
- Instructions
  - `div rs, rt  /  divu rs, rt`
  - No overflow or divide-by-0 checking
    - Software must perform checks if required
  - Use `mfhi`, `mflo` to access result

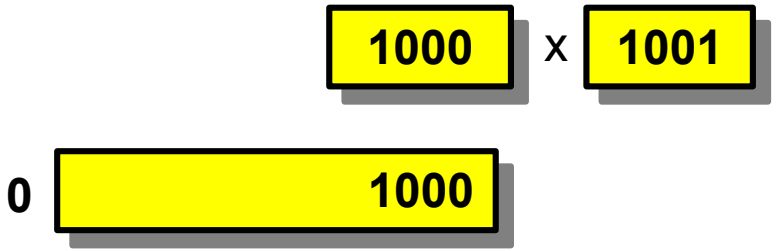
Module Outline

- Integer Arithmetic
  - Adder, Subtractor, Multiplier, Divider
- Arithmetic and Logical Unit Design
  - ALU Design in SystemC

Extra

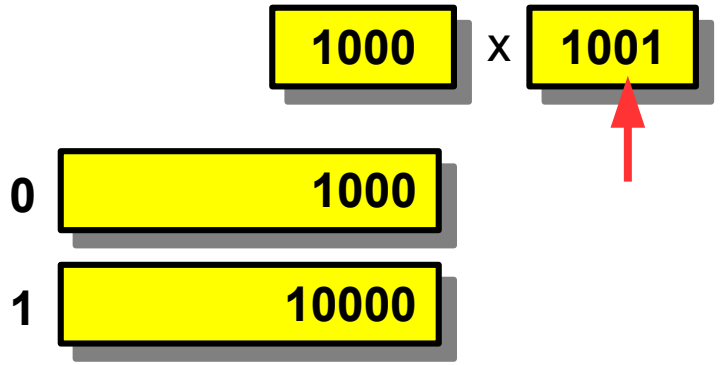
Multiplication

- Start with long-multiplication approach



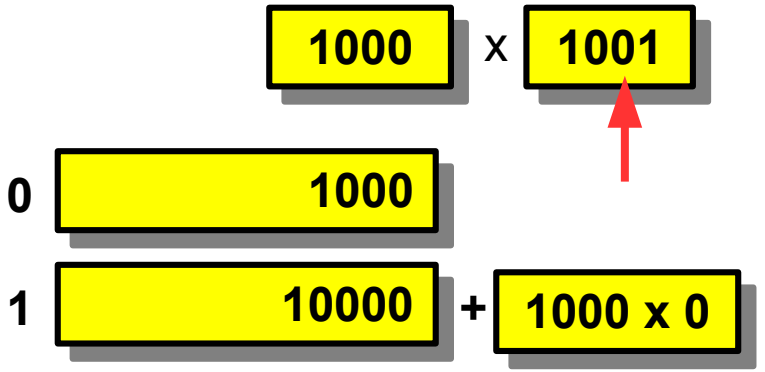
Multiplication

- Start with long-multiplication approach



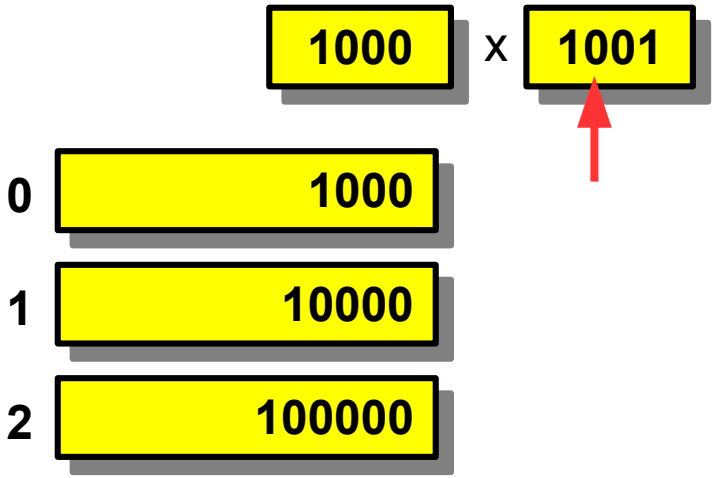
Multiplication

- Start with long-multiplication approach



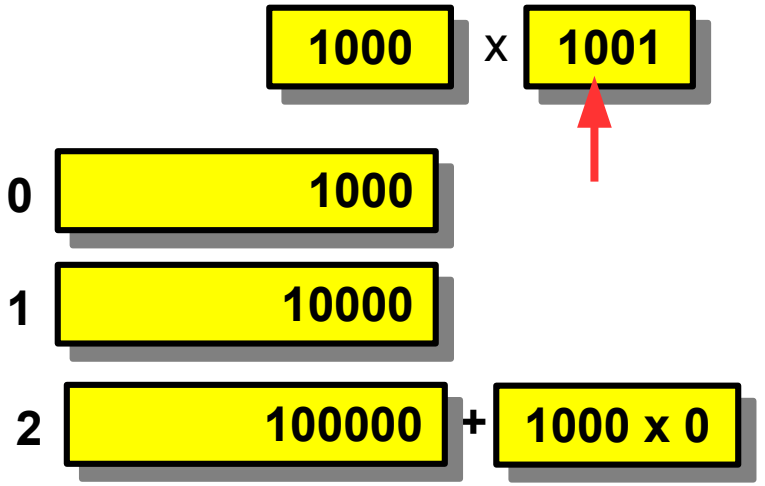
Multiplication

- Start with long-multiplication approach



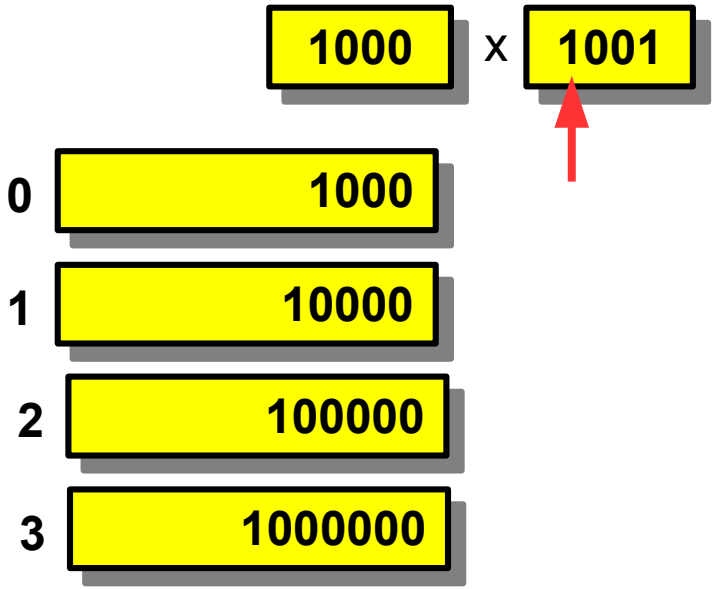
Multiplication

- Start with long-multiplication approach



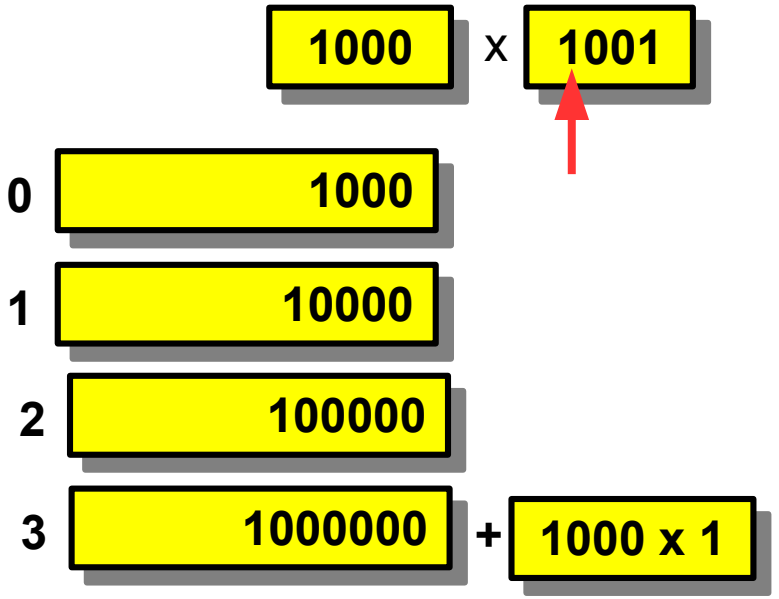
Multiplication

- Start with long-multiplication approach



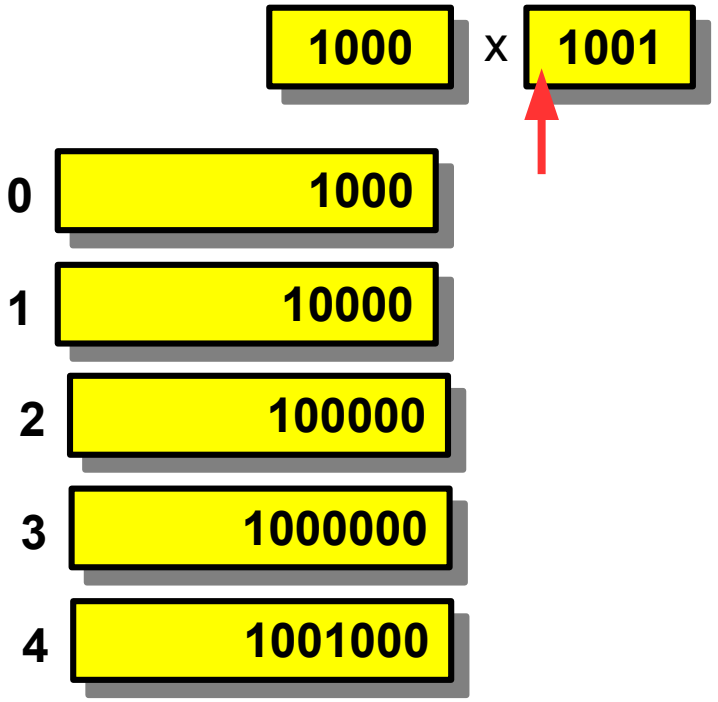
Multiplication

- Start with long-multiplication approach



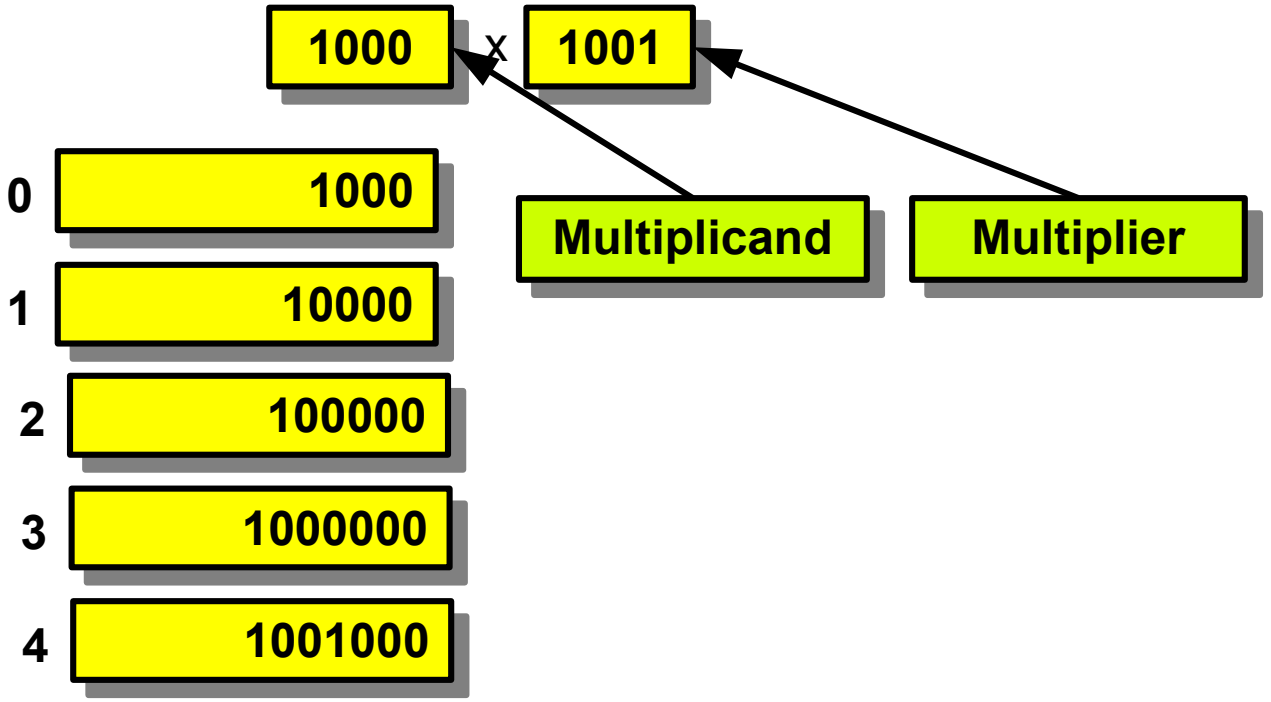
Multiplication

- Start with long-multiplication approach



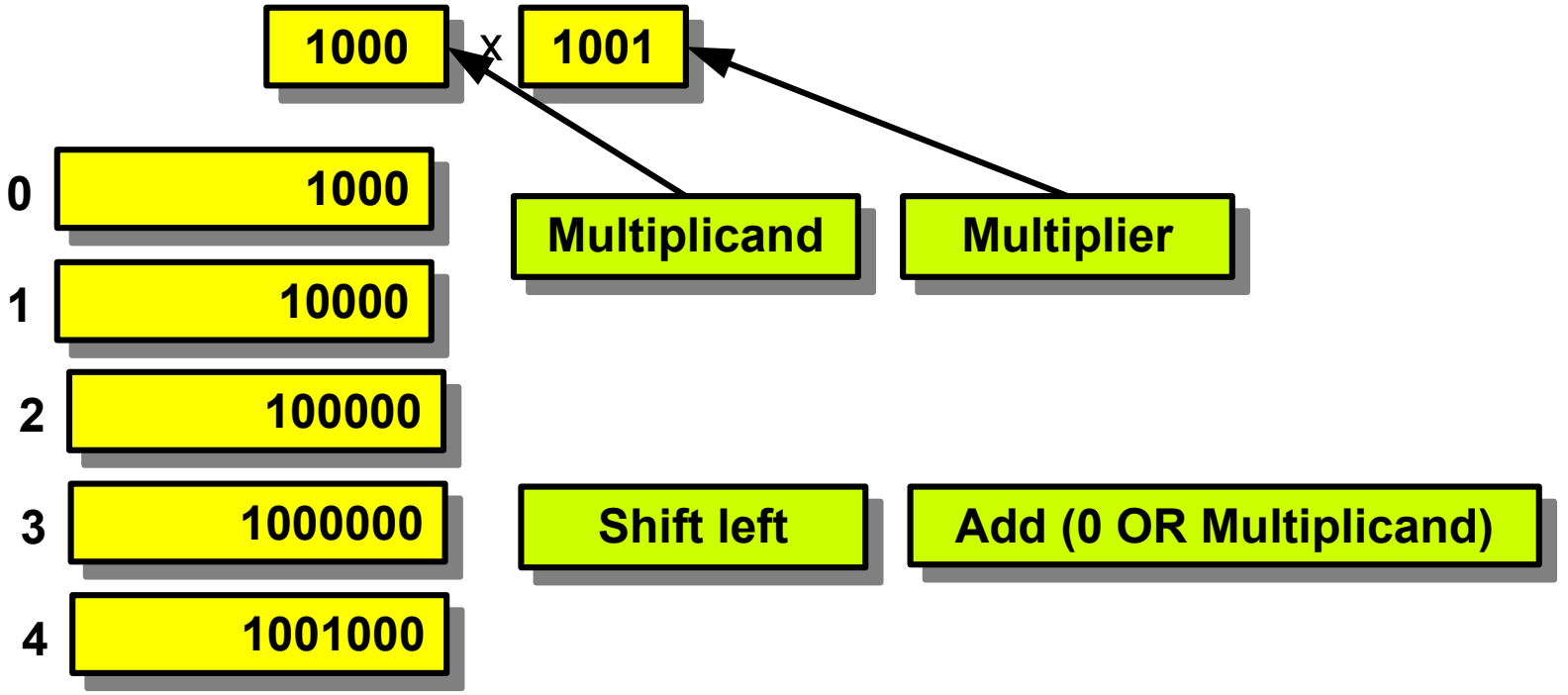
Multiplication

- Start with long-multiplication approach



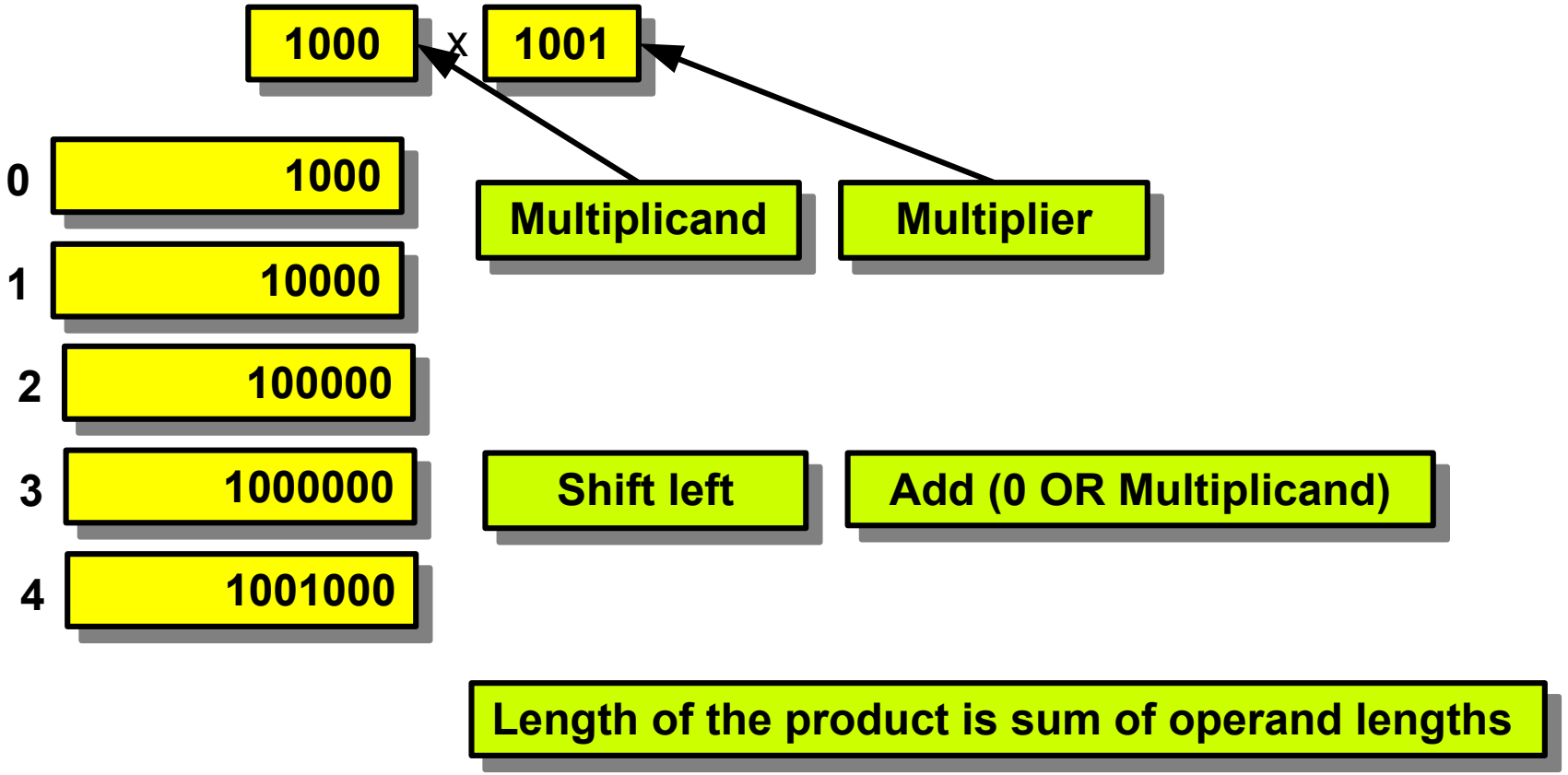
Multiplication

- Start with long-multiplication approach



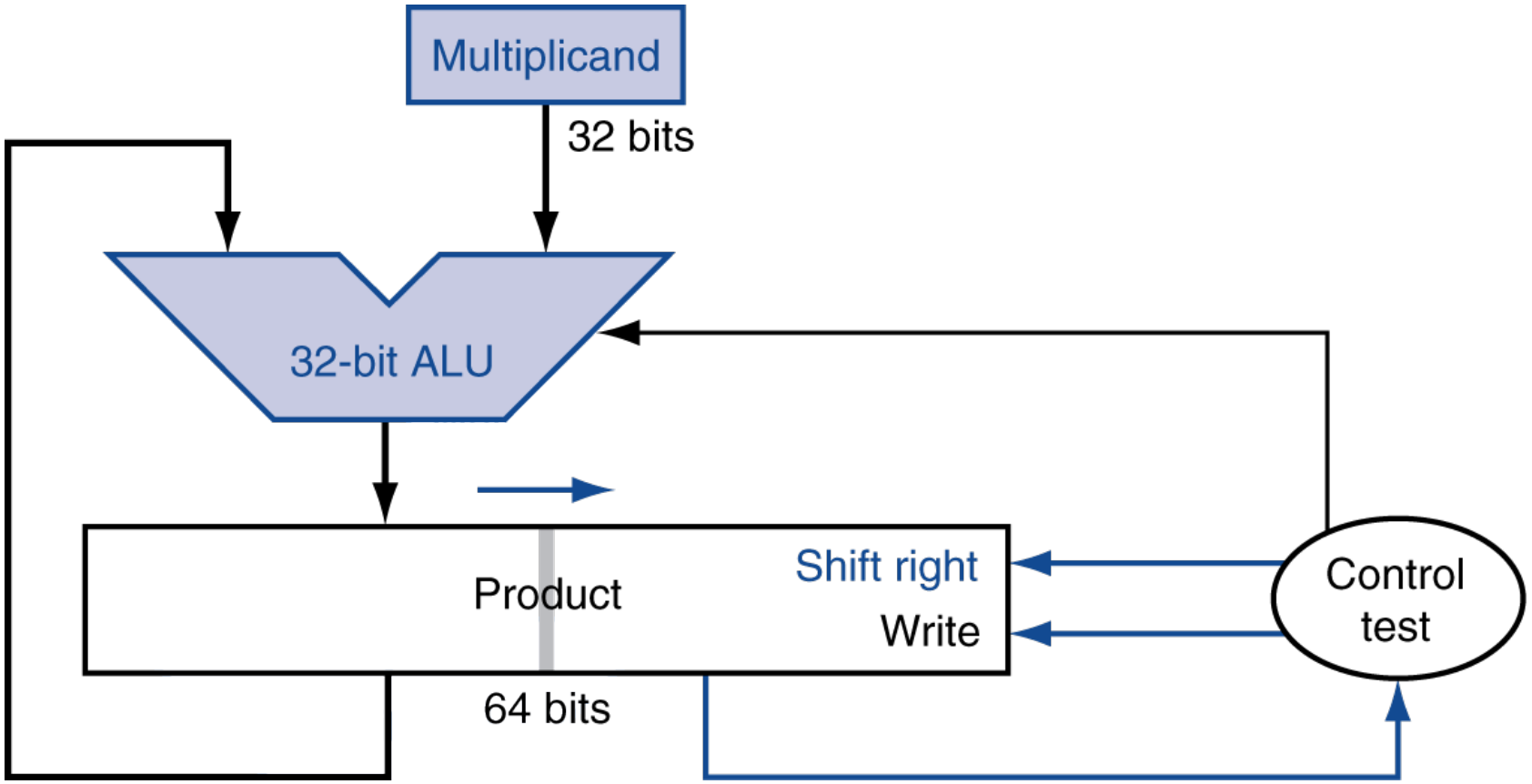
Multiplication

- Start with long-multiplication approach



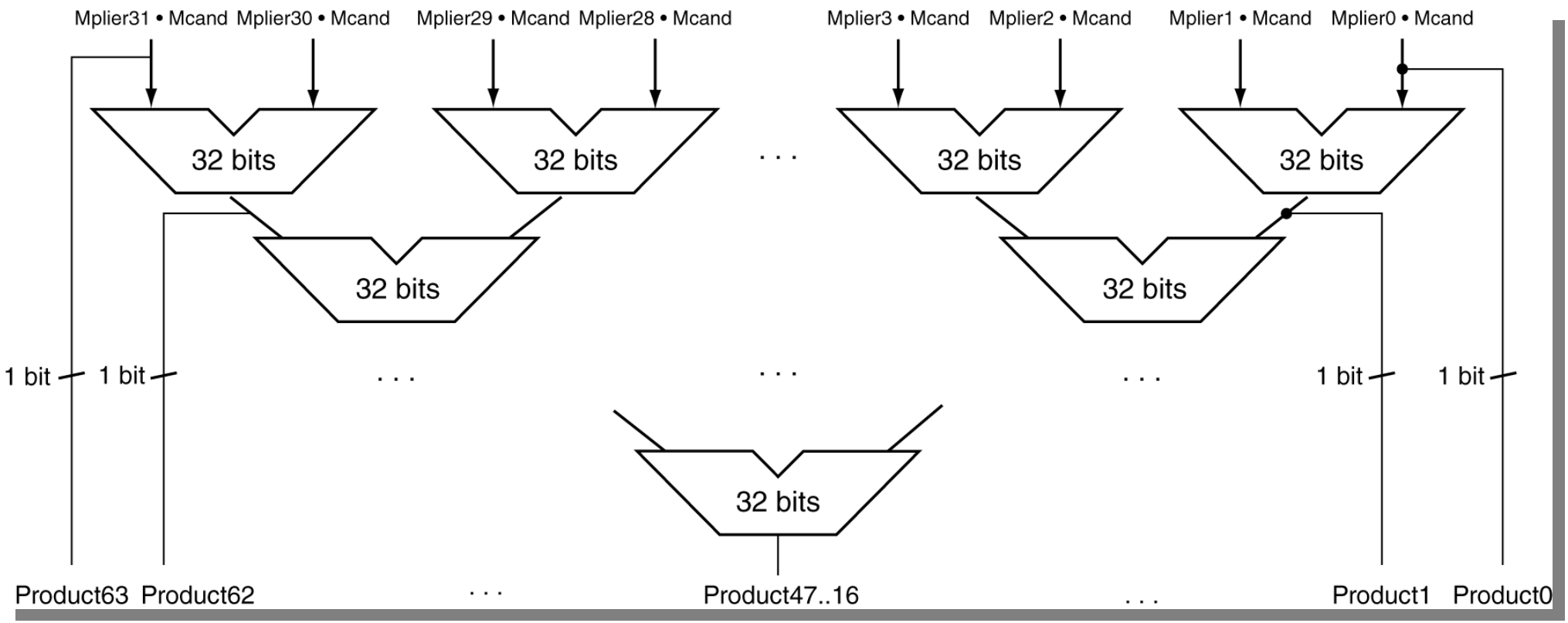
Optimized Multiplication Hardware

- Perform steps in parallel: add/shift

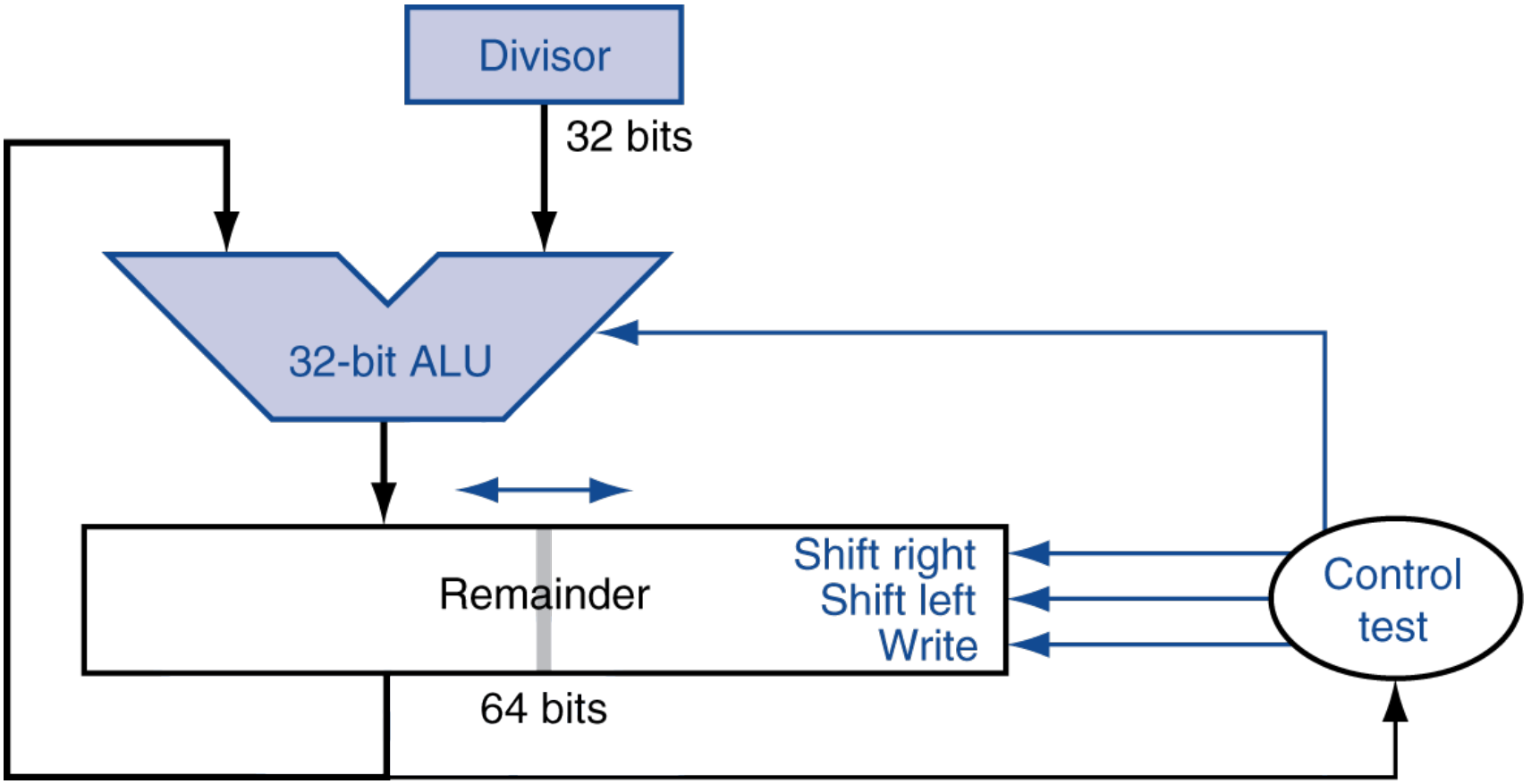


Fast Multiplication

- Using multiple adders



Optimized Divider



- One cycle per partial-remainder subtraction
- Looks a lot like a multiplier!
  - Same hardware can be used for both

Faster Division

- Can’t use parallel hardware as in multiplier
  - Subtraction is conditional on sign of remainder
- Faster dividers (e.g. SRT division) generate multiple quotient bits per step
  - Still require multiple steps