Lecture-5

Chapter-4

Computer Architecture and Organization- Jhon P. Hayes

Datapath Design

Datapath Design

- This lecture will addresses the register-level design of the datapath (data processing unit).

Fixed-Point Arithmetic:

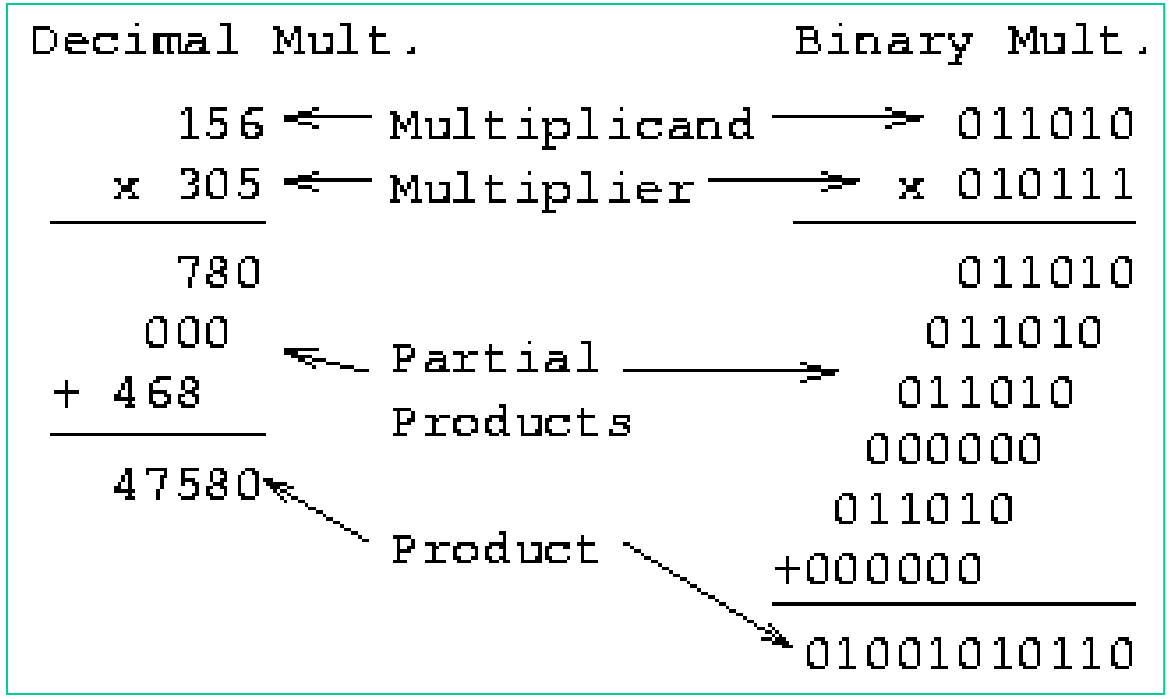
- Fixed point arithmetic is a technique which can be used instead of floating point to perform arithmetic on numbers we functional part.
- It has advantage as well as disadvantage when compared to floating point.
- The design of circuits to implement the four basic arithmetic instructions for fixed-point numbers- addition, subtraction, multiplication and division.

Multiplication

- Fixed point multiplication requires substantially more hardware than fixed point addition, and, as a result, it is not included in the instruction set of some is smaller processor.
- Multiplication is usually implemented by some form of repeated addition.
- A simple but slow method to compute X * Y is to add the multiplicand to Y to itself X times, where X is the multiplier.

Multiplication H/W

- Based on paper-and-pencil method of repeated shift-and-add operations



1010	Multiplicand Y
1101	Multiplier $X = x_3x_2x_1x_0$
1010	x_0Y
0000	$x_1 2Y$
1010	$x_2 2^2 Y$
1010	$x_3 2^3 Y$
10000010	Product $P = \sum_{j=0}^{3} x_j 2^j Y^j$

Observations

- Multiplication of single digits in binary multiplication is just an “AND” operation
- Multiplication of two n-bit numbers can be accomplished with (n-1) additions
- Can use array of AND gates, HA’s, and FA’s

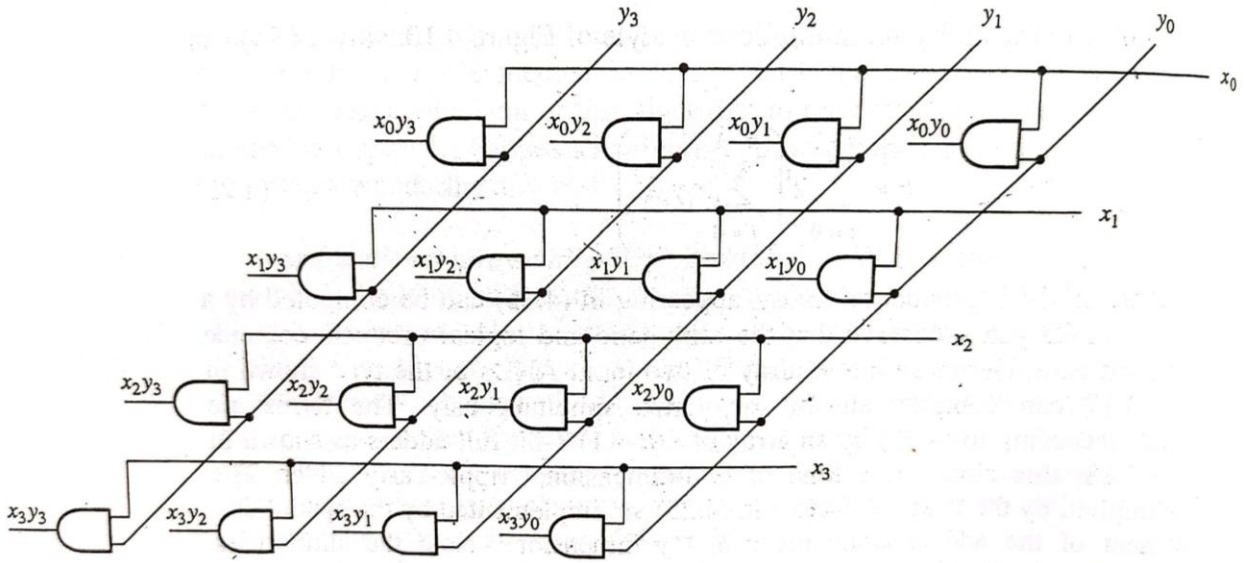


Figure 4.17
AND array for 4 × 4-bit unsigned multiplication.

- Question: Where is most of the “delay” in this design?

Division

- In fixed point division two numbers a divisor **V** and a dividend **D**, are given.
- The objective is to compute a third number **Q**, the quotient, such that **Q * V** equals or is very close to **D**.
- For example, if unsigned integer format are being used, **Q** is compute so that

  D = Q * V + R

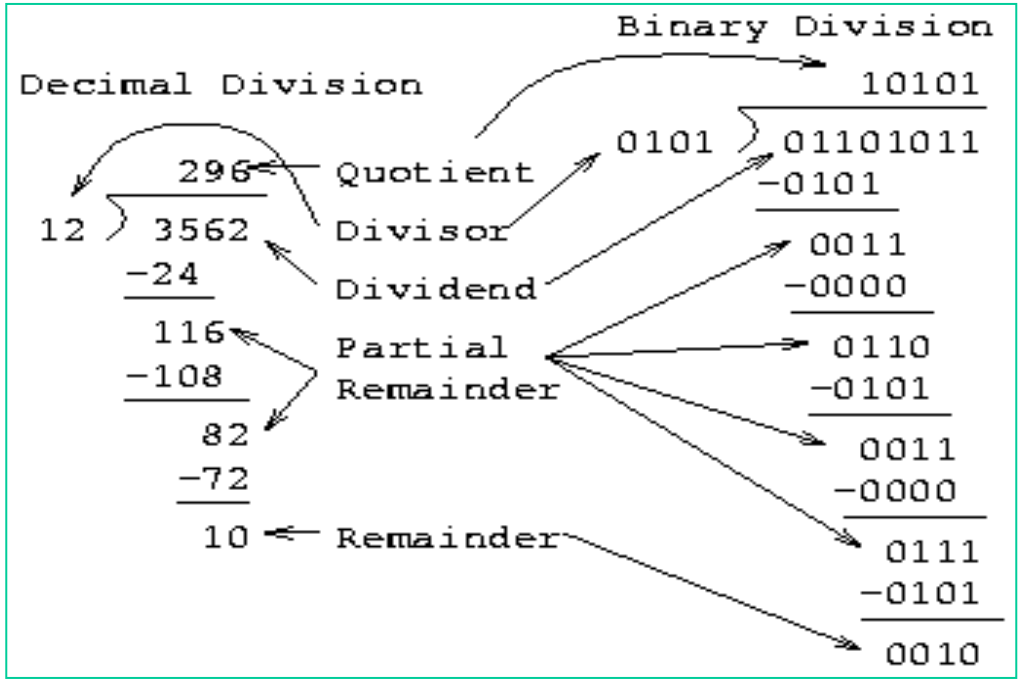
- Where **R**, the reminder is required to be less than **V**, that is **0 <= R <= V**. We can then write:

  D / V = Q + R / V

- Here **R/V** is a small quantity representing the error in using **Q** alone to represent **D/V**; this error is **zero** if **R = 0**.

Division H/W

Paper-and-pencil Division Method



	0111	Quotient $Q = q_3q_2q_1q_0$
Divisor $V = 101$	100110	Dividend $D = R_0$
	000	q_3V
	100110	R_1
	101	$q_2 2^{-1} V$
	10010	R_2
	101	$q_1 2^{-2} V$
	1000	R_3
	101	$q_0 2^{-3} V$
	011	R_4 = remainder R

Arithmetic Logic Unit (ALU)

- Uses of the ALU
  - process arithmetic and logical instructions
  - address calculations
  - act as a data conduit (route data between two points)
- ALU Design Techniques
  - many advanced transistor-level design techniques used to achieve fast ALU designs
  - gate-level designs can be “flattened” for better performance
  - basic ALU design is fairly simple

Design of One Bit of ALU

- ALU can be designed as an adder that can conditionally perform other functions based on the selection of control inputs
- ALU designed as a chain of identical 1-bit adders
  - may not be efficient for large numbers of bits
- Adder functions
  - sum = x EX-OR y EX-OR cin
  - cout = (x AND y) OR (y AND cin) OR (x AND cin)

Coprocessor

- Complicated arithmetic operations like exponentiation and trigonometric functions are costly to implement in CPU hardware, while software implementations of these operations are slow.
- A design alternative is to use auxiliary processors called **arithmetic coprocessors** to provide fast, low-cost hardware implementations of these special functions.
- In general, a coprocessor is a separate instruction set processor that is closely coupled to the CPU and whose instructions and registers are direct extensions of the CPU's. [Fig: **4.45**]

Coprocessor

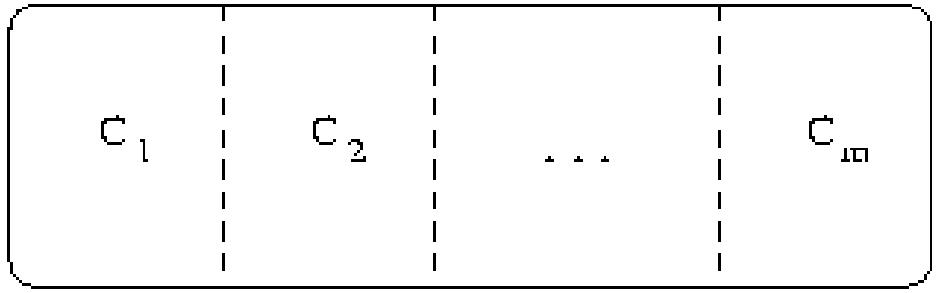
A coprocessor instruction typically contains the following three fields:

- An opcode F_0 that distinguishes coprocessor instructions from other CPU instructions.
- The address F_1 of the particular coprocessor to be used if several coprocessors are allowed.
- The type F_2 of the particular operation to be executed by the coprocessor.

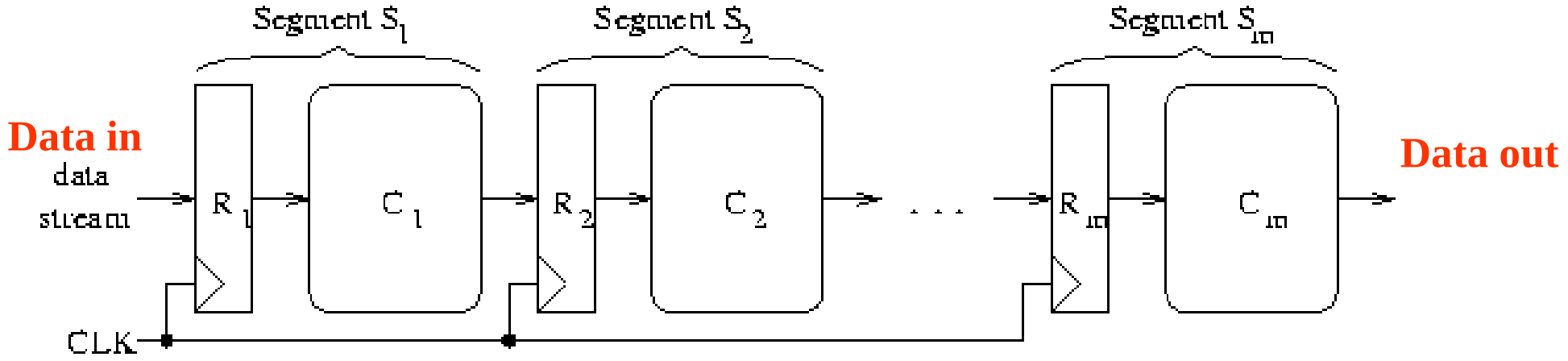
Pipeline Processing

- Pipelining is a general technique for increasing processor throughput without requiring large amounts of extra hardware.
- It is applied to the design of the complex datapath units such as multipliers and floating-point adders.
- It is also used to improve the overall throughput of an instruction set processor [More details in Chap 5].

Pipeline Processing: Basic Structure



(a)



(b)

Pipeline Processing: Basic Structure

- A pipeline processor consists of a sequence of m data-processing circuits, called ***stages or segments***, which collectively perform a single operation on a stream of data operands passing through them.
- Some processing takes place in each stage, but a final result is obtained only after an operand set has passed through the entire pipeline.
- As illustrated in next Fig, a stage S_i contains a multi-word input register or latch R_i, and a datapath circuit C_i that is usually combinational.
- The R_i's hold partially processed results as they move through the pipeline; they also serve as ***buffers*** that prevent neighboring stage from interfering with one another.
- A common clock signal causes the R_i's to change state synchronously.

Pipeline Processing: Basic Structure

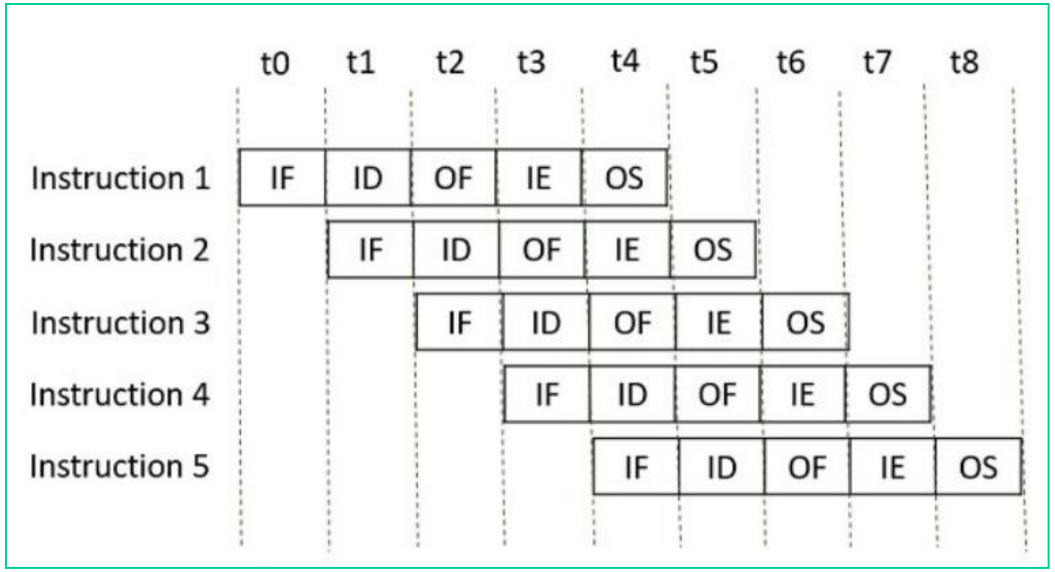
- Each R_i receives a new set of input data D_{i-1} from the preceding stage S_{i-1} except for R_1 whose data is supplied from an external source.
- D_{i-1} represents the results computed by C_{i-1} during the preceding clock period.
- Once D_{i-1} has been loaded into R_i, C_i proceeds to use D_{i-1} to compute a new data set D_i.
- Thus in each clock period, every stage transfers its previous results to the next stage and computes a new set of results.

Pipeline Processing: Basic Structure

An instruction in a process is divided into 5 subtasks likely

Instruction Fetch	Instruction Decode	Operand Fetch	Instruction Execute	Operand Store

1. In the first subtask, the instruction is fetched.
2. The fetched instruction is decoded in the second stage.
3. In the third stage, the operands of the instruction are fetched.
4. In the fourth, arithmetic & logical operation are performed on the operands to execute the instruction.
5. In the fifth stage, the result is stored in memory.



That's All

Thank You