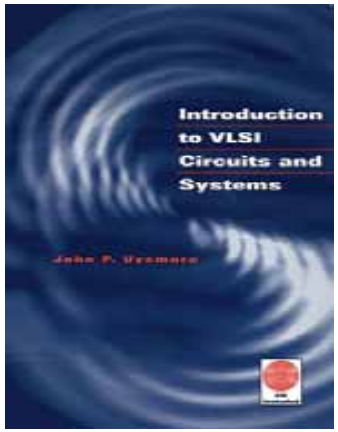
國立勤益科技大學
National Chin-Yi University of Technology

Introduction to VLSI Circuits and Systems
積體電路概論

Chapter 15
System-Level Physical Design



賴秉樑
Dept. of Electronic Engineering
National Chin-Yi University of Technology
Fall 2007

Outline

- Clocked Flip-flops
- CMOS Clocking Styles
- Pipelined Systems
- Clock Generation and Distribution
- System Design Considerations

Clocked Flip-flops

- *Synchronous* design employs clocking signals to coordinate the movement of data through the system
- In Figure 15.2, the data bit D is loaded into the DFF only on a rising clock edge

$$Q(t_0 + t_{ff}) = D(t_0) \qquad (15.1)$$

Where t_o is the rise edge, and t_{ff} is the time delay when the output has this value

- In high speed design, the limiting circuit factor is the DFF delay time t_{ff} that is determined by the electronics and the load
  - » Decreasing t_{ff} allows for a higher frequency clock
  - » The tradeoff of speed and power

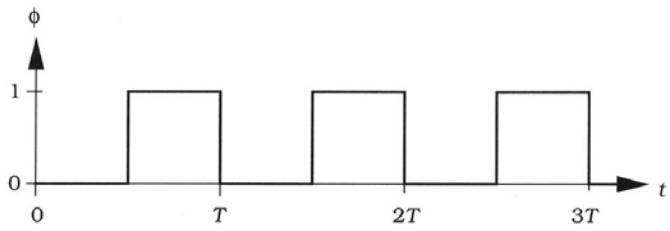


Figure 15.1 Ideal clocking signal

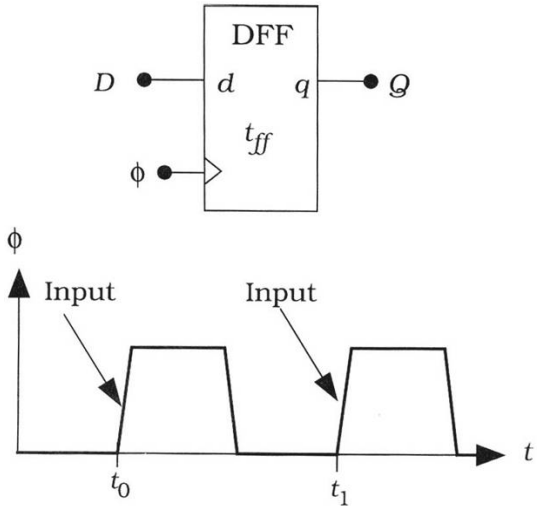
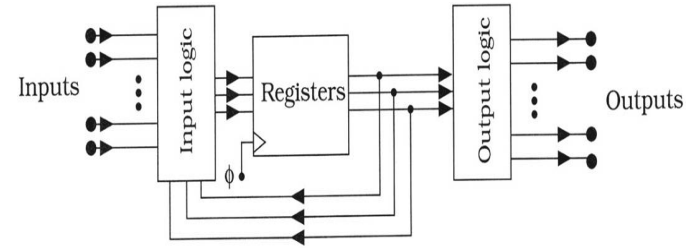


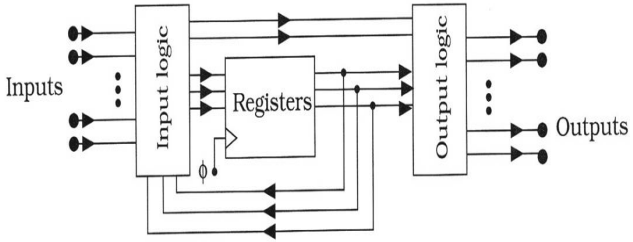
Figure 15.2 Timing in a DFF

Classical State Machines (1 / 2)

- Two models for state machines that use single-clock timing are shown in Figure 15.3
  - » Moore machine and Mealy machine
  - » Huffman model (Figure 15.4): contains both the Moore and Mealy models



(a) Moore machine



(b) Mealy machine

Figure 15.3 Moore and mealy state machines

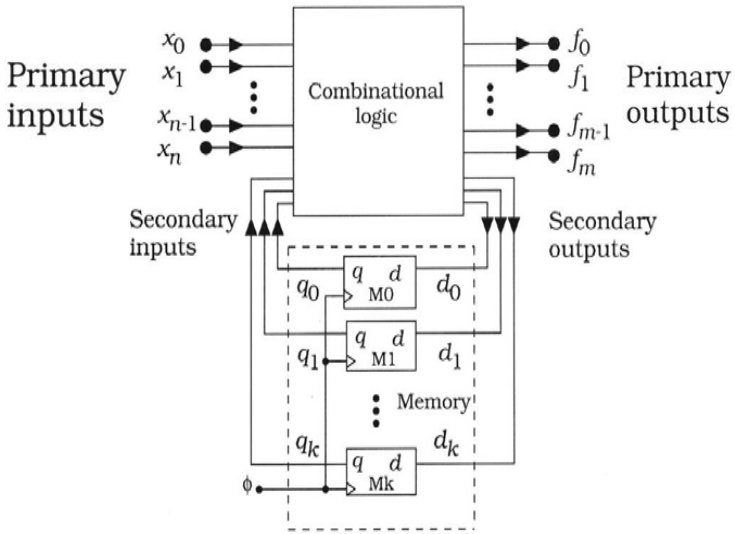


Figure 15.4 Huffman model of a state machine

Classical State Machines (2 / 2)

- FPGA design are also heavily based on classical state machine theory
  - » For examples, in combinational logic, Individual gates, PLAs, Programming Logic Device (PLD), and groups of multiplexors
  - » Programming is achieved with EPROMs, fuses, SRAM arrays, and some contain lookup tables (LUT, e.g. FPGA) to aid in the design
- In Figure 15.5

$$\sum_r m_r \qquad (15.2)$$

(The AND-plane of the PLA can be programmed to produce minterms m_r)

$$T > t_{ff} + t_d + t_{su} \qquad (15.3)$$

(The clock T must be large enough to allow completion)

Where,

- t_{ff} is the delay time form input to output of the flip-flop
- t_d is the logic delay time through the PLA
- t_{su} is the "setup time" of the flip-flip

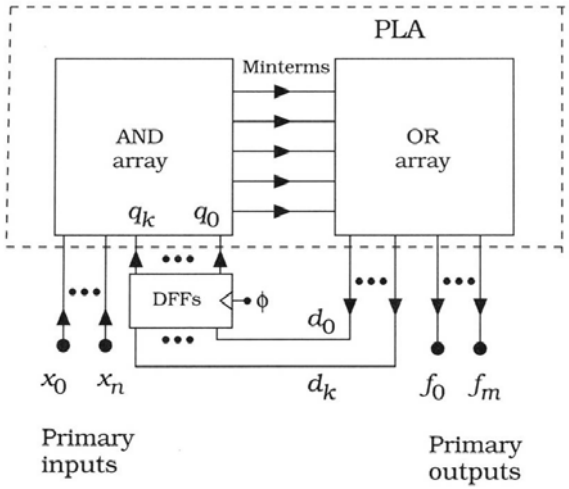


Figure 15.5 Huffman state machine using PLA logic

Outline

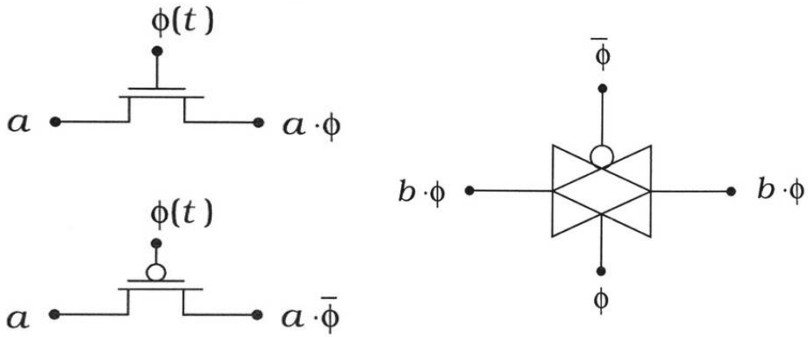
- Clocked Flip-flops
- CMOS Clocking Styles
- Pipelined Systems
- Clock Generation and Distribution
- System Design Considerations

Clocked Logic Cascades

- Ideal waveforms for the complimentary signal

$$\phi \cdot \bar{\phi} = 0 \quad (15.4) \qquad \text{(Figure 15.6)}$$

$$V_{\max} = V_{DD} - V_{Tn} \quad (15.5) \qquad \text{(Figure 15.7)}$$



(a) FETs (b) Transmission gate

Figure 15.7 Clock-controlled transistors

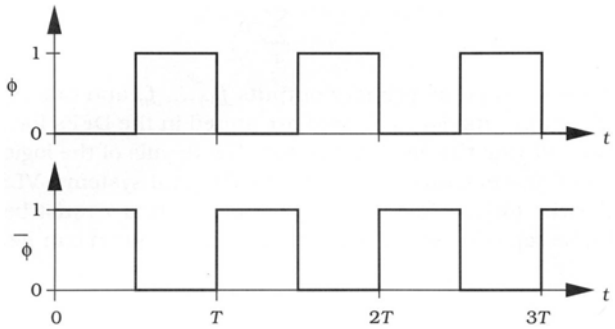


Figure 15.6 Complementary clocks

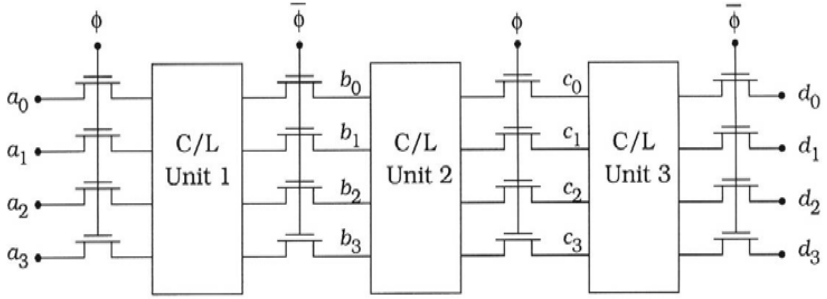


Figure 15.8 A clocked cascade

Timing Circles and Clock Skew

- *Timing circles*: are simple constructs that can be useful for visualizing data transfer
  - » In Figure 15.9, this defines what is known as a 50% duty cycle
- *Clock skew*: the timing of a clock is out of phase with the system reference

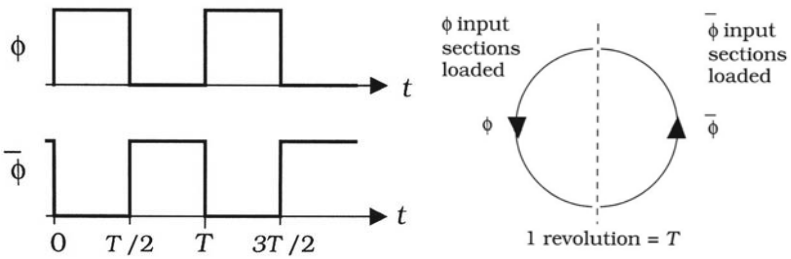


Figure 15.9 Timing circle for a single-clock, dual-phase cascade

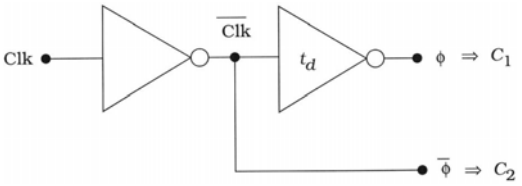


Figure 15.10 Clock generation circuit

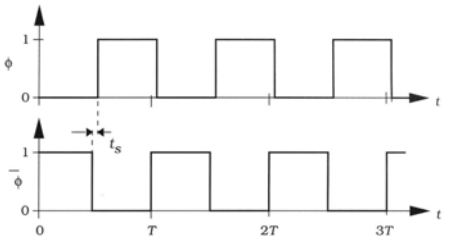


Figure 15.11 Clock skew

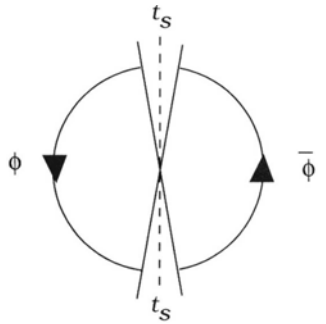


Figure 15.12 Timing circle with clock skew

Circuit Effects and Clock Frequency (1/2)

- The logic-level description of the clocked cascade masks the circuit characteristics that determine the ultimate speed

$$\left(\frac{T}{2}\right)_{\min} = t_{FET} + t_{NOT} \qquad (15.7)$$

$$\left(\frac{T}{2}\right)_{\min} = t_{r,FET} + t_{HL,NOT} \qquad (15.8)$$

$$f_{\max} = \frac{1}{T_{\min}} = \frac{1}{2(t_{r,FET} + t_{HL,NOT})} \qquad (15.9)$$

$$f_{\max} = \frac{1}{T_{\min}} = \frac{1}{2(t_{r,FET} + t_{CL})} \qquad (15.10)$$

$$I_{leak} = -C_{in}\frac{dV_{in}}{dt} \qquad (15.11)$$

$$\left(\frac{T}{2}\right)_{\max} = t_h \qquad (15.12)$$

$$f_{\min} = \frac{1}{T_{\max}} = \frac{1}{2t_h} \qquad (15.12)$$

$$V_M = \frac{V_{DD} - |V_{Tp}| + \sqrt{\frac{\beta_n}{\beta_p}}V_{Tn}}{1 + \sqrt{\frac{\beta_n}{\beta_p}}} \qquad (15.13)$$

$$\frac{\beta_n}{\beta_p} = \frac{\kappa_n'\left(\frac{W}{L}\right)_n}{\kappa_p'\left(\frac{W}{L:}\right)_p} \qquad (15.14)$$

$$\frac{\beta_n}{\beta_p} = \frac{\kappa_n'}{\kappa_p'} \qquad (15.15)$$

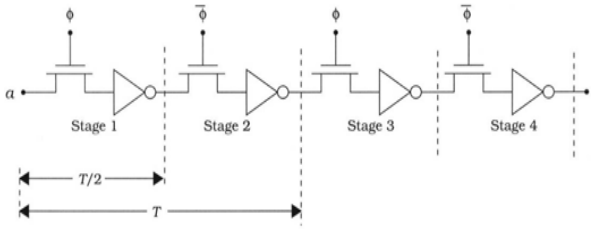
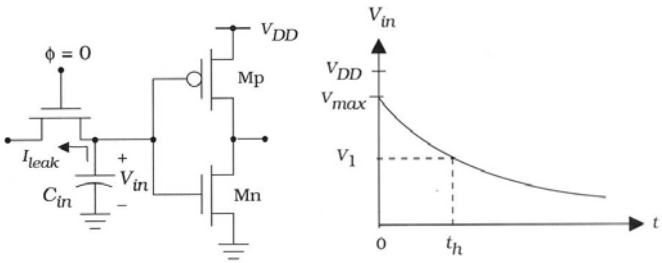


Figure 15.13 Shift register circuit



(a) Circuit (b) Voltage decay

Figure 15.14 Charge leakage in the shift register

Circuit Effects and Clock Frequency (2 / 2)

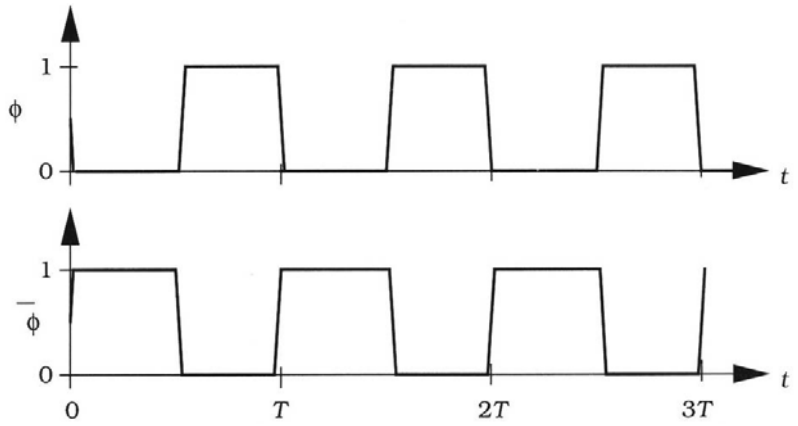


Figure 15.15 Clocking waveforms with finite rise and fall times

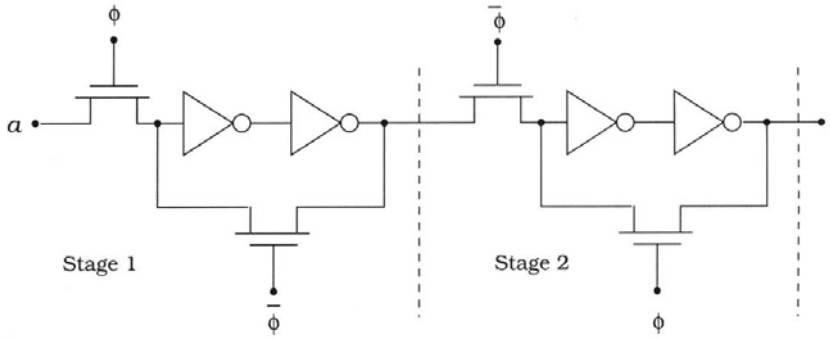


Figure 15.16 Static shift register design

Dual Non-overlapping Clocks

- In this technique, two distinct non-overlapping clocks ψ_1 and ψ_2 are used such that (Fig. 15.17 and 15.18)

$$\phi_1(t) \cdot \phi_2(t) = 0 \qquad (15.17)$$

- Finite-state machines that are based on dual-clock schemes can provide powerful interactive capabilities (Fig. 15.19)

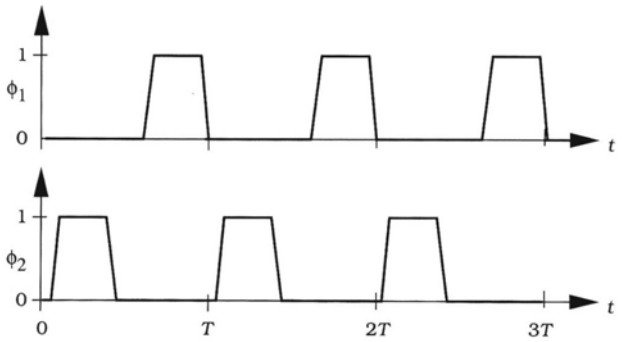


Figure 15.17 Dual non-overlapping clocks

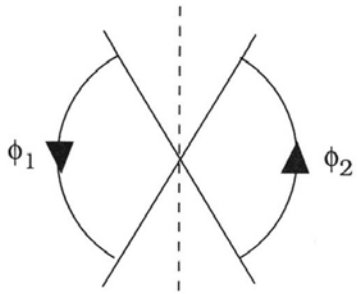


Figure 15.18 Timing circuit for a 2-clock network

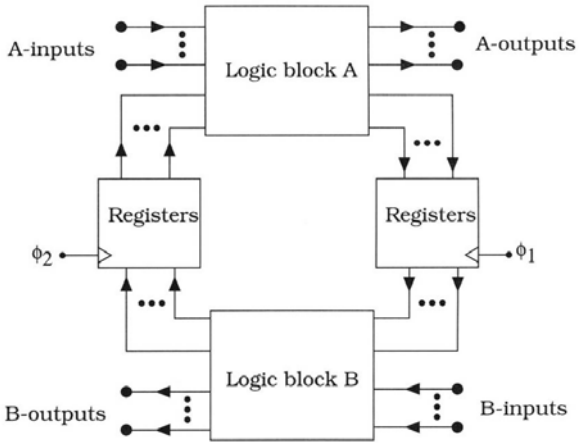


Figure 15.19 A dual-clock finite-state machine design

Other Multiple-clock Schemes

- It is possible to create different mutliple-clock schemes to control clocked logic cascades and state machine
  - » For example, a triple, non-overlapping clock set
- However, in modern high-speed VLSI, complicated clocking schemes introduce too many problems
  - » Solution: speed gains are accomplished by improved *circuit design*, *processing*, and *architectural modifications*
  - » The most popular approach is to use a *single-clock, dual-phase system*

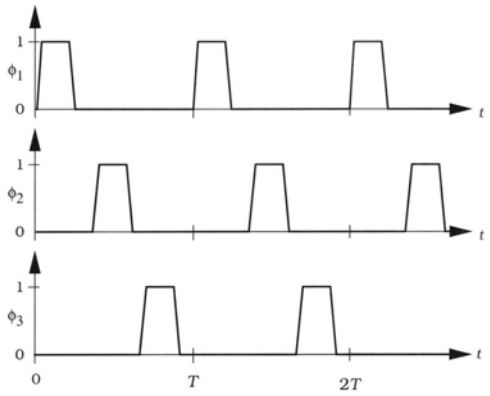


Figure 15.20 Triple, non-overlapping clock signals

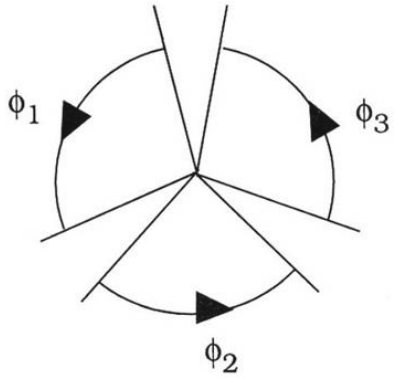


Figure 15.21 Timing circle for a 3-clock non-overlapping network

Dynamic Logic Cascades (1/2)

- Dynamic logic circuits achieve synchronized data flow by controlling the internal operational states of the logic gate circuits
  - » Typical domino logic state: In section 9.5 of Chapter 9
  - » P: per-charge phase
  - » E: Evaluation phase

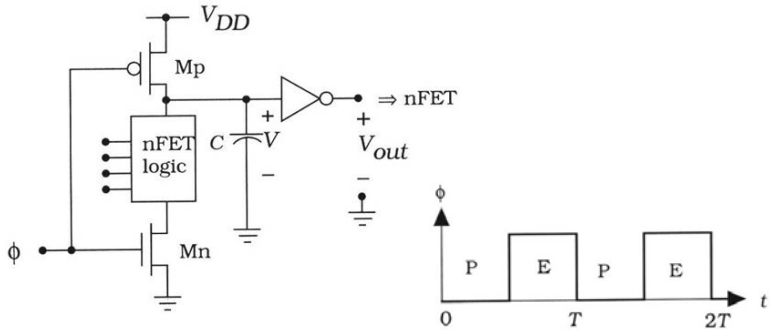


Figure 15.22 Operation of a domino logic state

- In Figure 15.23,

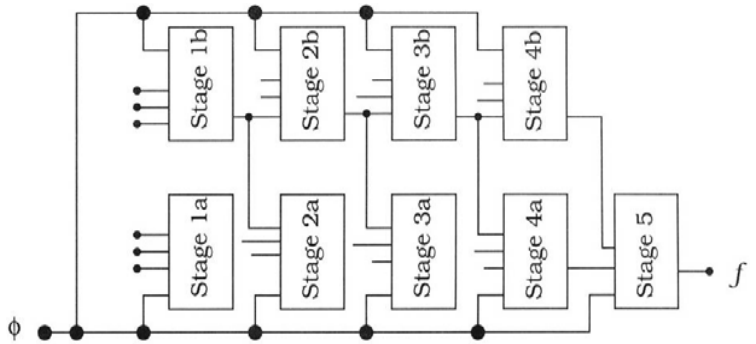


Figure 15.23 A dynamic logic cascade

Dynamic Logic Cascades (2/2)

- In Figure 15.24, the data transfer into and out of a dynamic logic cascade is sequenced with the clock
  - » The number of stages that can be included in the chain is determined by the delay for the case where every stage switches
  - » However, this introduces *charge leakage problem*
  - » Solution: charge keeper circuits
- True Single-Phase Clock (TSPC): use only a single clock ψ throughout
  - » Two TSPC latches are shown in Figure 15.25

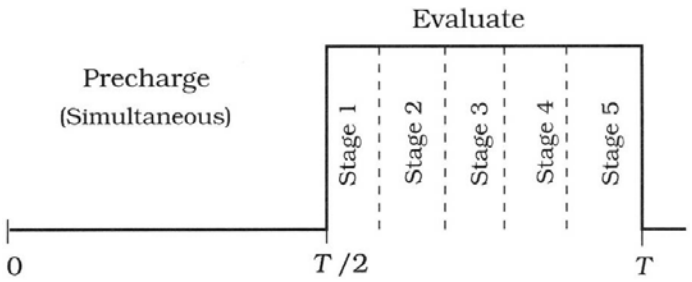
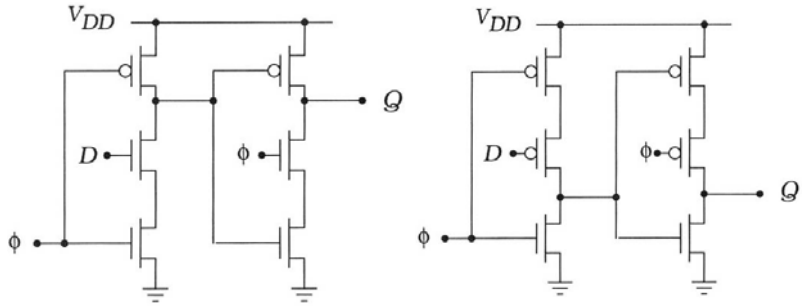


Figure 15.24 Timing sequence in the domino cascade



(a) n-block (b) p-block

Figure 15.25 True single-phase clock latches

Outline

- Clocked Flip-flops
- CMOS Clocking Styles
- Pipelined Systems
- Clock Generation and Distribution
- System Design Considerations

Basic Concept of Pipelining

- Pipelining is a tech. that is used to increase the throughput of a sequential set of distinct data inputs through a synchronous logic cascade

$$T > t_{ff} + t_d + t_{su} + t_s \quad (15.18)$$

$$t_{hold} < PW \quad (15.19)$$ (PW: pulse width of the clock)

$$f < \frac{1}{t_{ff} + t_d + t_{su} + t_s} \quad (15.20)$$

Where,
- t_{ff}: flip-flop delay time
- t_d: logic delay time
- t_{su} : setup time of the flip-flip
- t_{hold} : hold time of the flip-flip

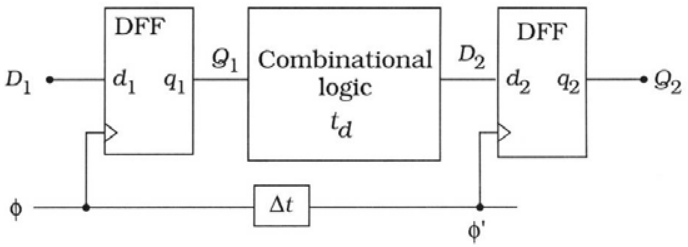


Figure 15.26 Basic pipelined stage for timing analysis

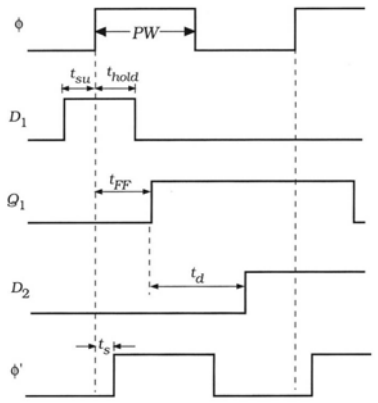


Figure 15.27 Waveform quantities for timing analysis

Pipelining (1/2)

- Pipelined systems are designed to increase the overall throughput of a set of sequential input states by dividing the cascade into small visualization of the problem (Figure 15.28)
- Once a circuit completes a calculation and passes the result on to the next stage, it remain idle for the rest of the clock cycle (Figure 15.29)

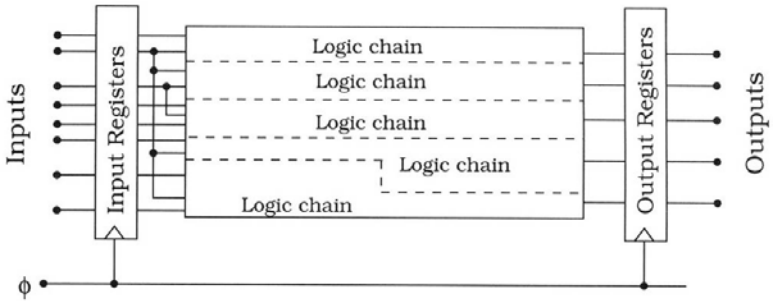


Figure 15.28 Logic chains in a clocked system

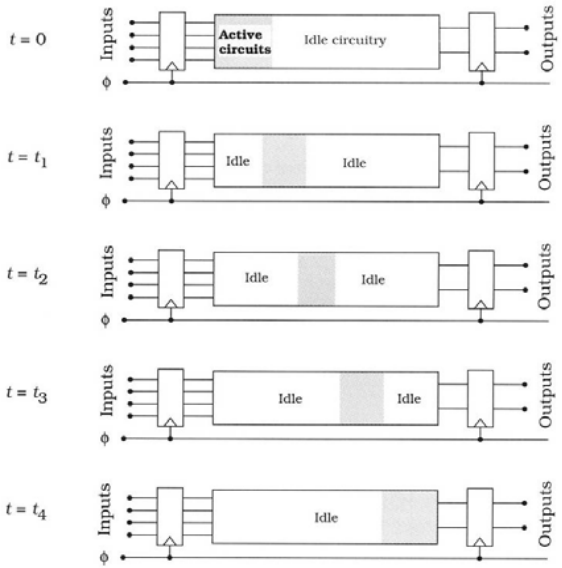


Figure 15.29 Circuit activity in a logic cascade

Pipelining (2 / 2)

- Since the delay through a logic gate varies its complexity and parasitics, the logic propagation rate will not be uniform

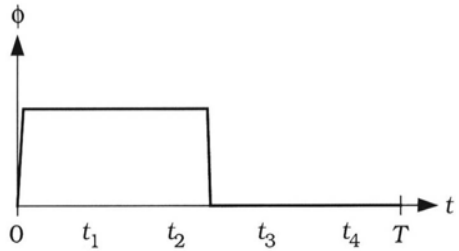


Figure 15.30 Progression times in the logic cascade

- In Figure 15.31, dividing the long logic into small groups, add registers between the sections, and use a faster clock, then most of the circuits will be active at any given time

$$4T_{pipe} + (N-1)T_{pipe} = (N+3)T_{pipe} \quad (15.21)$$

$$T_i > t_{ff} + t_{su} + t_{d,i} + t_{s,i+1} \quad (15.22)$$

$$T_{pipe} = \max\{T_1,...,T_m\} \quad (15.23)$$

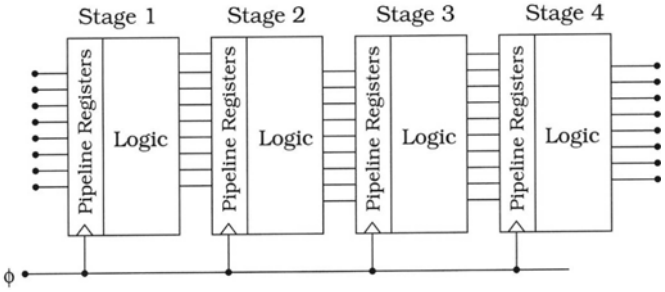


Figure 15.31 A 4-stage pipeline

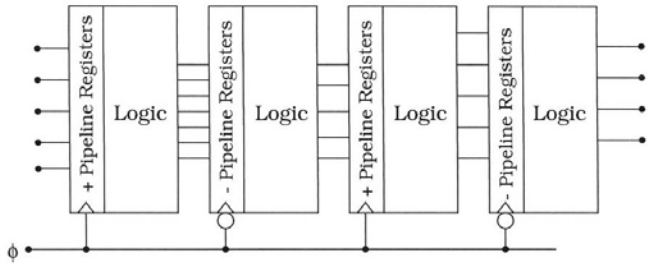


Figure 15.32 Pipeline with positive edge and negative edge-triggering

Outline

- Clocked Flip-flops
- CMOS Clocking Styles
- Pipelined Systems
- Clock Generation and Distribution
- System Design Considerations

Clock Distribution

- When frequencies f reach the 1GHz (10^9 Hz) level corresponding to a clock period of

$$T = \frac{1}{f} = 1 \ ns \qquad (15.24)$$

- However, distribution of the clocking signal to various points of the chip is complicated because the intrinsic RC time delay τ increases as the square of the line length l according to

$$\tau = Bl^2 \qquad (15.25)$$

$$\Delta t_1 = B(l_b^{\ 2} - l_a^{\ 2}) \qquad (15.26)$$

$$\Delta t_2 = B(l_c^{\ 2} - l_b^{\ 2}) \qquad (15.27)$$

- Problems: clock skew and signal distortion will be very difficult to deal with in large chips

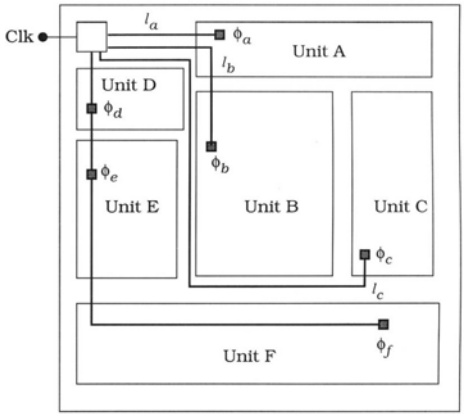


Figure 15.31 A 4-stage pipeline

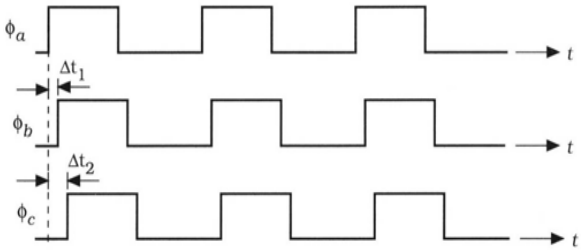


Figure 15.31 A 4-stage pipeline

Clock Stabilization and Generation (1/2)

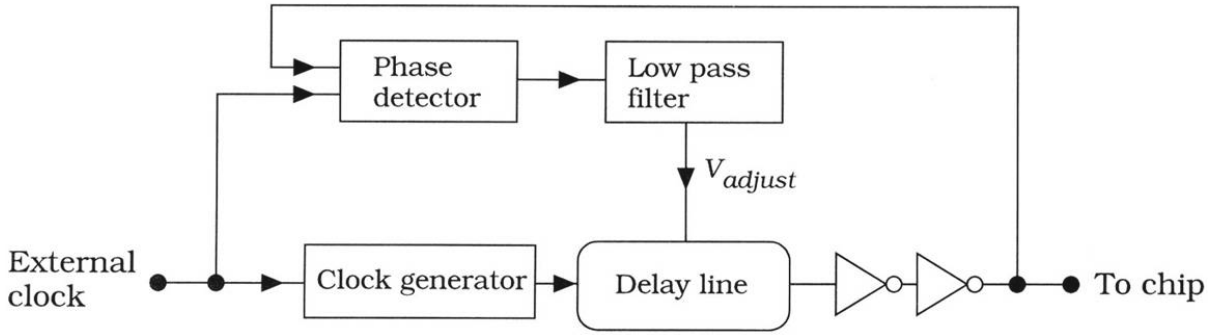


Figure 15.35 A basic clock stabilization network

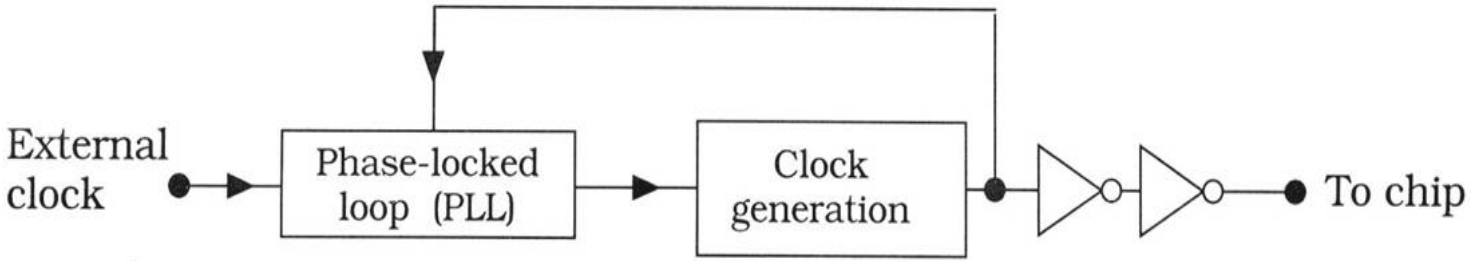


Figure 15.36 Phase-locked loop (PLL) stabilization circuit

Clock Stabilization and Generation (2/2)

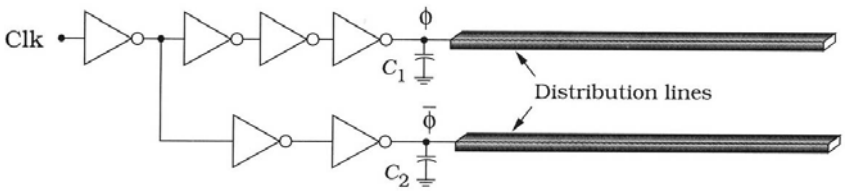


Figure 15.37 Inverter-based clock generation circuit

$$C = C_{line} + \sum C_G \qquad (15.28)$$

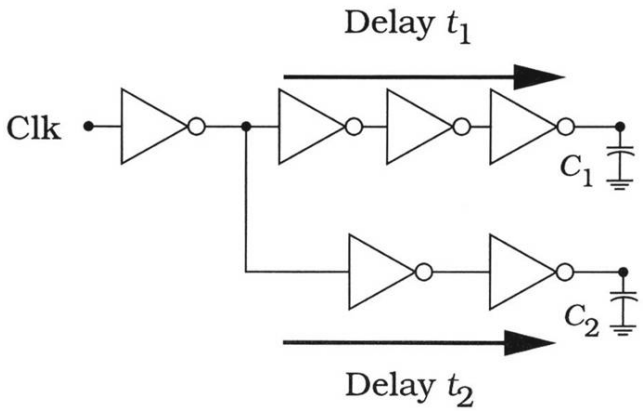


Figure 15.38 Skew minimization circuit

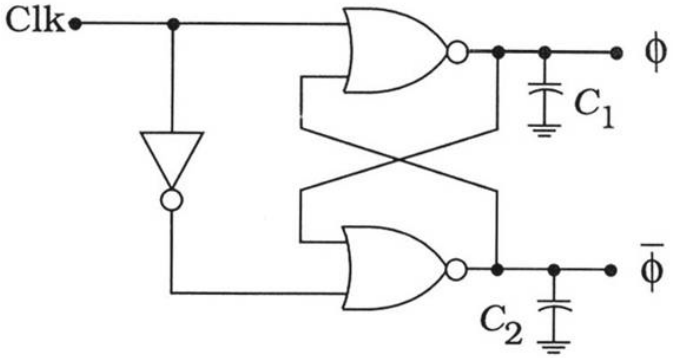


Figure 15.39 Generating complementary clocks using a latch

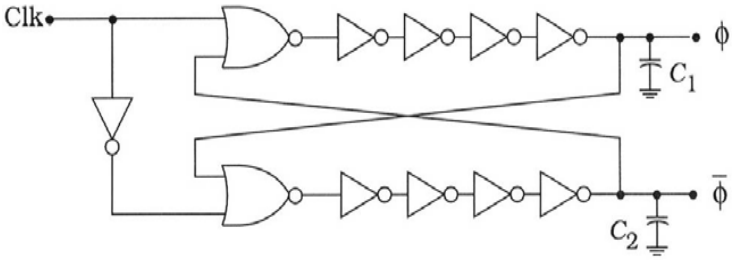
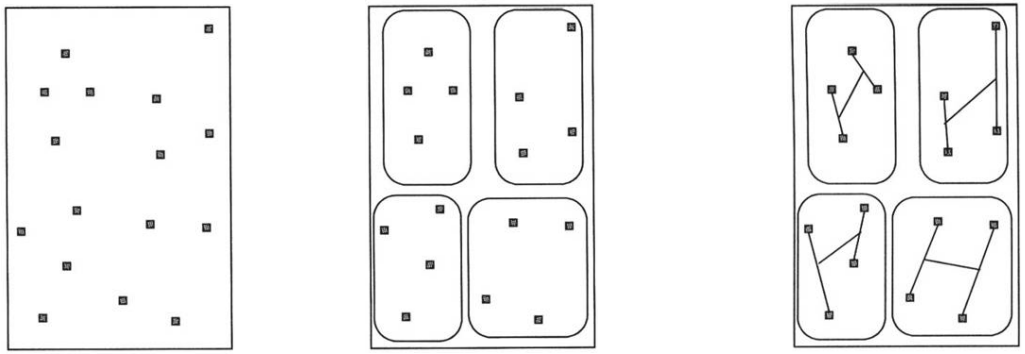


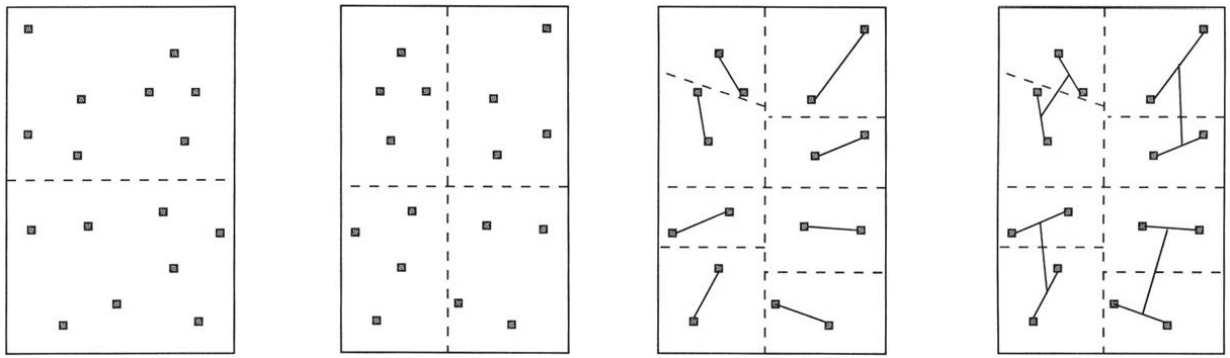
Figure 15.40 Circuit for producing non-overlapping clocks

Clock Routing and Driver Trees



(a) Clocking points (b) First grouping (c) Interior routing

Figure 15.41 Simplified view of the clock routing problem



(a) First (b) Second (c) Third (d) Routing

Figure 15.42 Partitioning steps for defining clocking groups

Driver Tree (1 / 2)

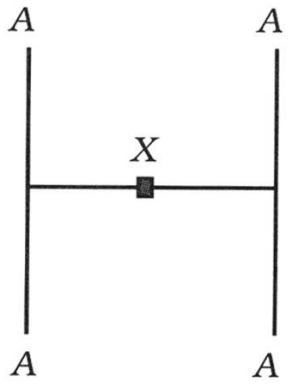
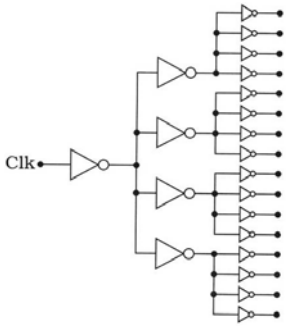
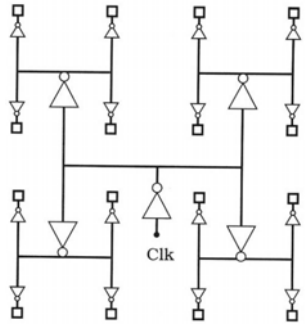


Figure 15.43 Geometrical analysis of the letter “H”



(a) Driver tree



(b) Application to H-tree

Figure 15.45 Driver tree arrangement

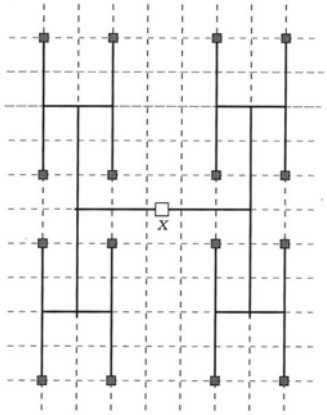


Figure 15.44 Macro-level H-type distribution tree

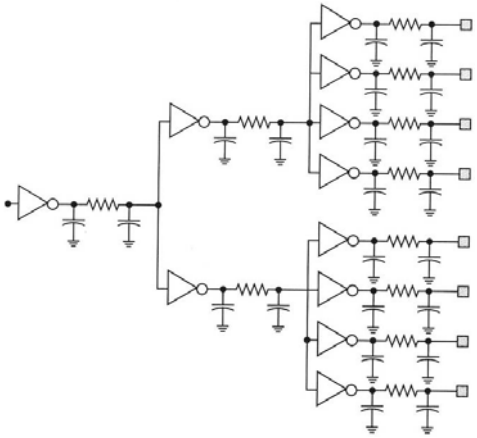


Figure 15.46 Driver tree design with interconnect parasitics

Driver Tree (2 / 2)

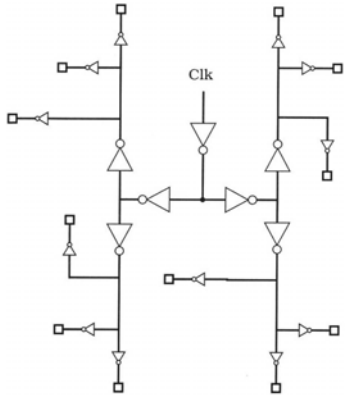
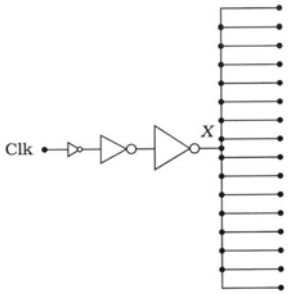
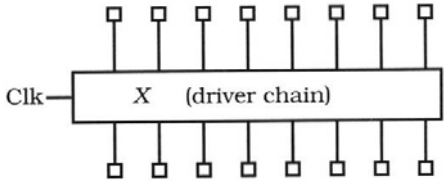


Figure 15.47 Distribution scheme with equivalent driver segments



(a) Driver tree



(b) Chip distribution

Figure 15.49 Single driver tree with multiple outputs

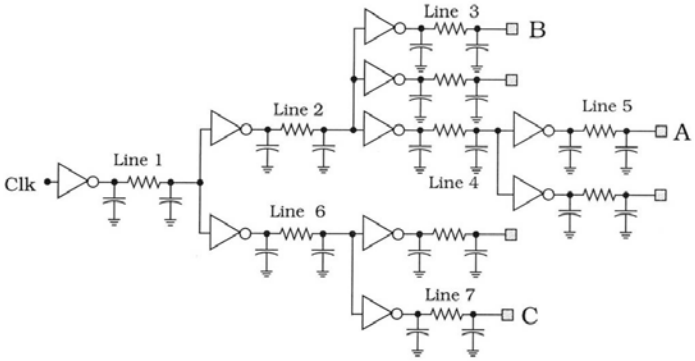


Figure 15.48 Electrical circuit for a non-symmetrical distribution

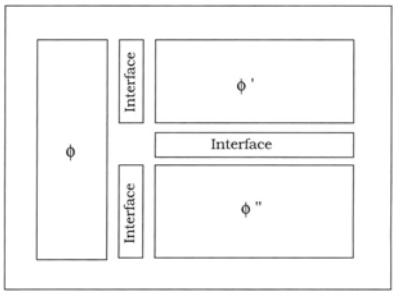


Figure 15.50 Asynchronous system clocking

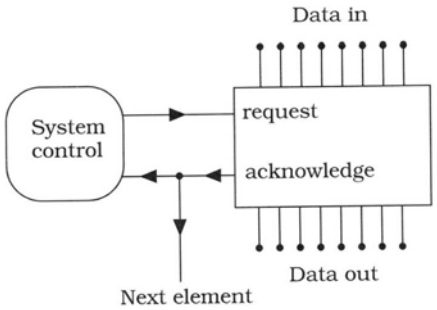


Figure 15.51 Operation of a self-timed element

Outline

- Clocked Flip-flops
- CMOS Clocking Styles
- Pipelined Systems
- Clock Generation and Distribution
- System Design Considerations

Bit Slice Design (1 / 2)

- For example, an ALU circuit design (Fig. 15.52)

$$C = C(A, B; control) \quad (15.30)$$

- Bit slice design is based on the fact that logic deals with data at the bit-level

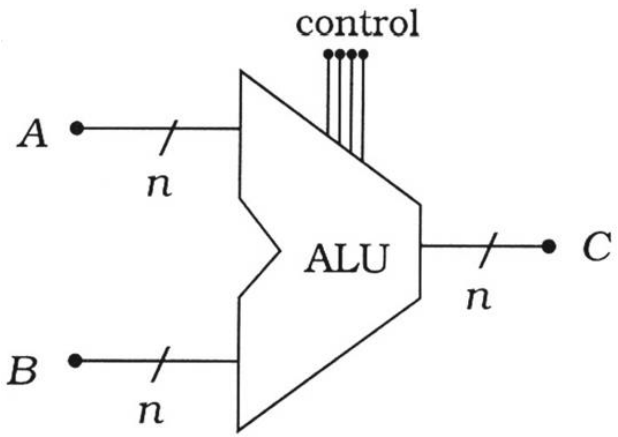


Figure 15.52 An n-bit ALU

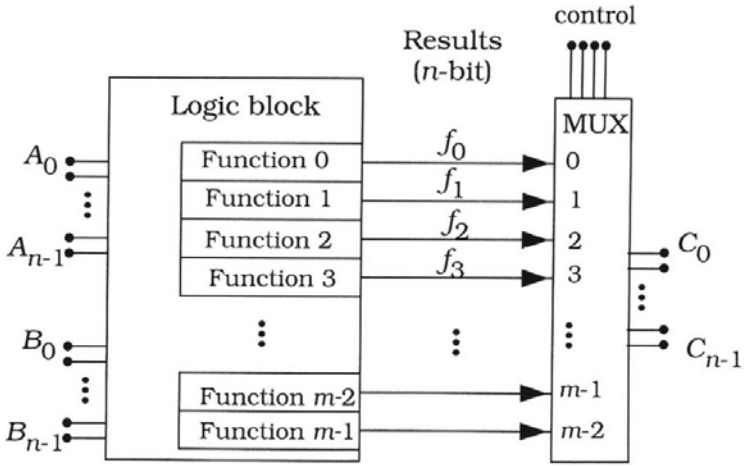


Figure 15.53 Block description

Bit Slice Design (2 / 2)

- With the bit slice philosophy, an *n*-bit ALU is created by paralleling *n* identical slices as shown in Figure 15.55
  - » The repetition will also appear at the logic, circuit, and silicon levels
  - » Advantage: library and instance building for large system design
- For example, in Verilog

reg A, B, C; (1-bit data-type)

reg[31,0] *A, B, C;* (32-bit data-type)

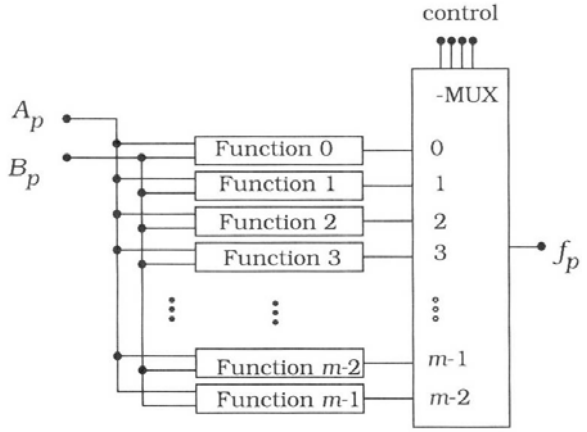


Figure 15.54 An ALU bit slice

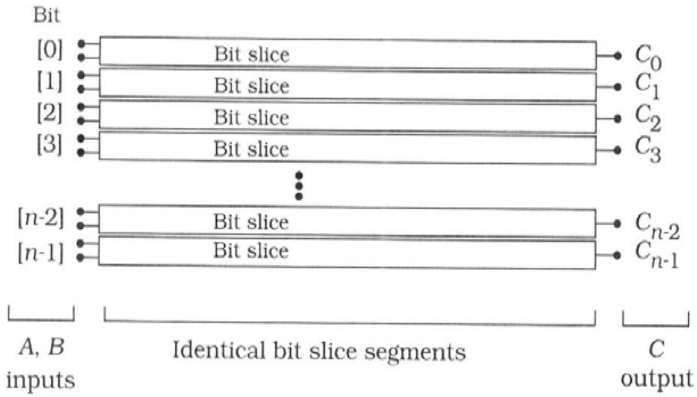
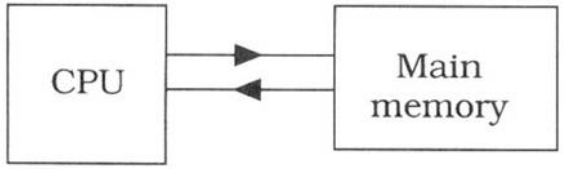


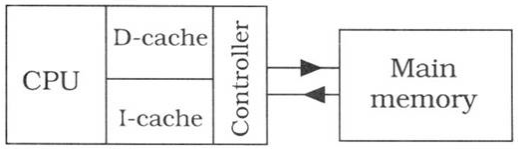
Figure 15.55 ALU design with bit slices

Cache Memory

- Cache memory is designed to be place in between the CPU and the main memory to speed up the system operation
  - » SRAM memory structure
  - » Instruction-cache (I-cache) : is used to hold instructions that are fetched from the M.M. where the program to run freely
  - » Data-cache (D-cache): is used to hold the results of computations
- Figure 15.57 shows a simple block diagram for a *dual-issue superscalar* design using I-cache and D-cache (*Computer Architecture*)



(a) Basic system



(b) Cache modification

Figure 15.56 Adding cache memory

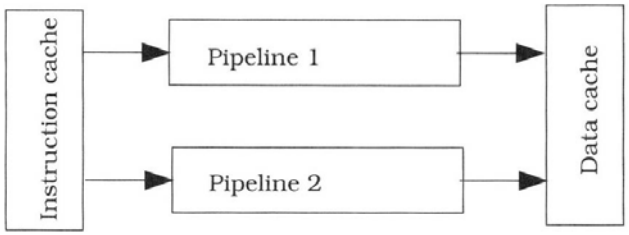


Figure 15.57 Block diagram of a dual-issue superscalar machine

Systolic Systems and Parallel Processing

- In a systolic system, the movement of the data is controlled by a clock, moving one phase on each cycle
- In parallel processing, a PE (processor element) can be as simple as an AND gate, or as a complex as a general-purpose microprocessor (Fig. 15.58)
  - » Individual process elements communicate via switching arrays, which are controlled by either local or global signals
  - » The strongest aspects of VLSI: *Wafer scale integration, WSI*

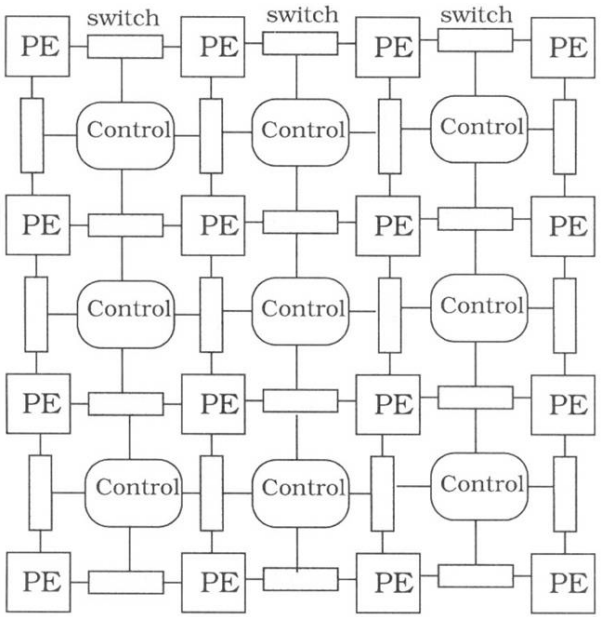


Figure 15.58 Regular patterning in a parallel processing network