EEM412 Course

Introduction to IC Design – II

Instructor

Dr. İsmail Enis Ungan

Subsystem Design

- Digital functions may be divided into the categories of;
  - data-path operators
  - memory elements
  - control structures
  - I/O cells

- Data-path Operators
  - Data is usually wide
  - Function can be implemented by n identical circuits (bit-slice).
  - Operators may be sequenced in time or in space, therefore they are placed next to each other.
  - Generally data flow is in one direction, while control signals are orthogonal to data flow.

Data-Path Operators

- Data-path Operators
  - Addition/Subtraction
  - Parity Generators
  - Comparators
  - Zero/One Detectors
  - Binary Counters
  - Boolean Operations - ALUs
  - Multipliers
  - Shifters

Addition/Subtraction

- Full Adder

$$SUM = ABC + AB'C' + A'B'C + A'BC'$$

$$SUM = C(AB + A'B') + C'(AB' + A'B)$$

$$SUM = A \oplus B \oplus C$$

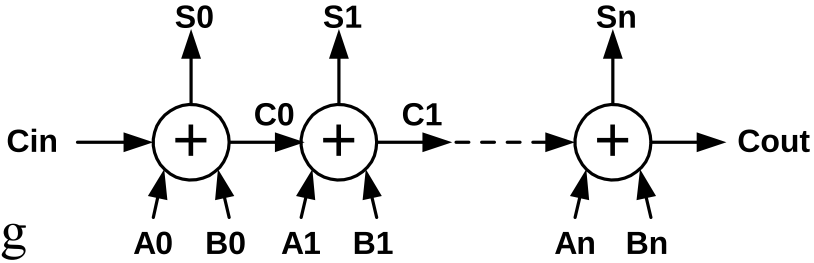
$$CARRY = AB + AC + BC$$

$$CARRY = AB + C(A + B)$$

$$\Rightarrow SUM = ABC + (A+B+C)\,CARRY'$$

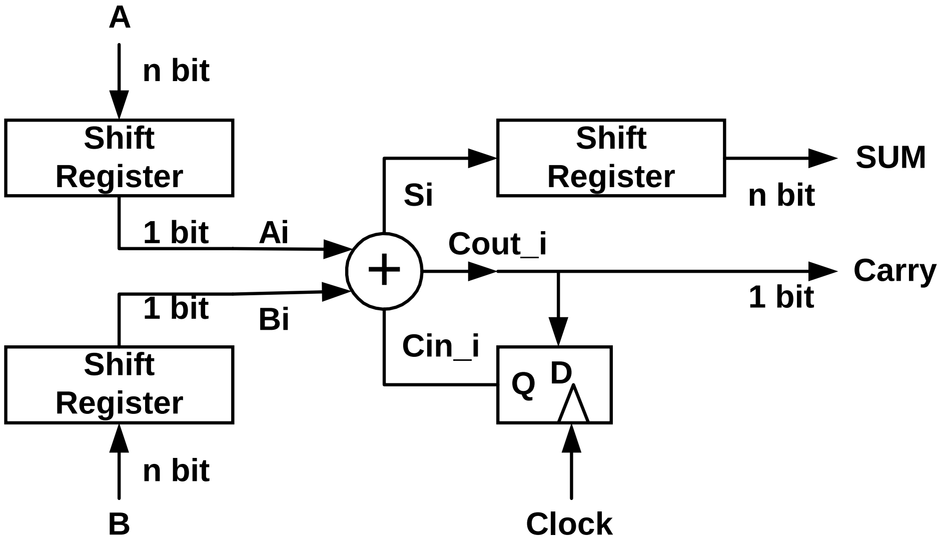
- n-bit Ripple Carry Adder
  - Built from one-bit full adders.
  - Adder becomes subtractor by inverting one of the two inputs and providing ‘1’ as the carry input.

Ripple Carry Adder



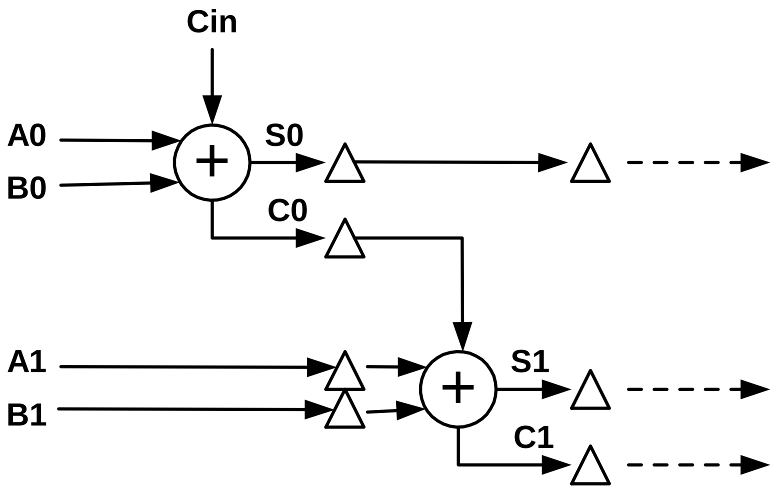
Addition/Subtraction

- Serial Adder
  - Requires a number of very short clock cycles to add two n-bit numbers.
  - Can be implemented on nybbles (4-bit words).
  - If both SUM and CARRY are registered at each cycle then adder is called carry-save adder. Hence, Serial Adder is a carry-save adder.



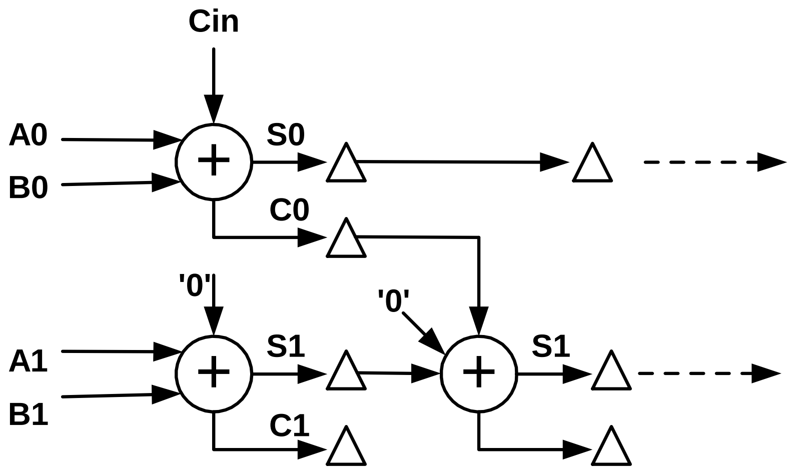
Addition/Subtraction

- Pipelining
  - Example: Both sum and carry are saved at the full-adder outputs. Delay elements are simply D-FFs.



Addition/Subtraction

- Pipelining
  - Example: That can be extended to add 4 numbers.

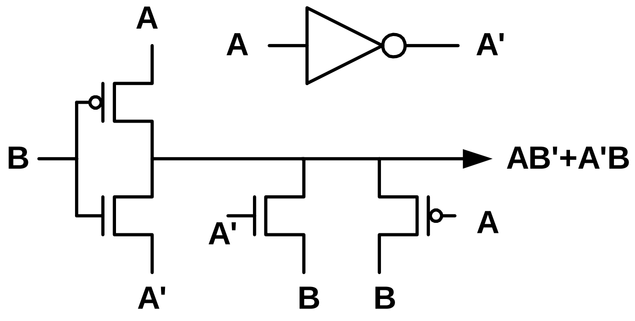


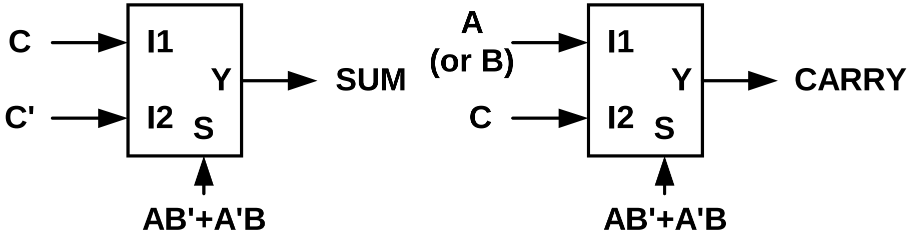
Addition/Subtraction

- Transmission-Gate Adder
  - Less MOSFETs.

$$SUM = A \oplus B \oplus C = \begin{cases} C & when \quad A \oplus B = 0 \\ \overline{C} & when \quad A \oplus B = 1 \end{cases}$$

$$CARRY = \begin{cases} C & when \quad A \oplus B = 1 \\ A & when \quad A \oplus B = 0 \end{cases}$$





Addition/Subtraction

- Carry-Look-Ahead Adder

$$C_0 = A_0 \cdot B_0 + (A_0 + B_0) \cdot C_{in}$$

$$C_1 = A_1 \cdot B_1 + (A_1 + B_1) \cdot C_0$$

$$\downarrow$$

$$C_i = A_i \cdot B_i + (A_i + B_i) \cdot C_{i-1}$$

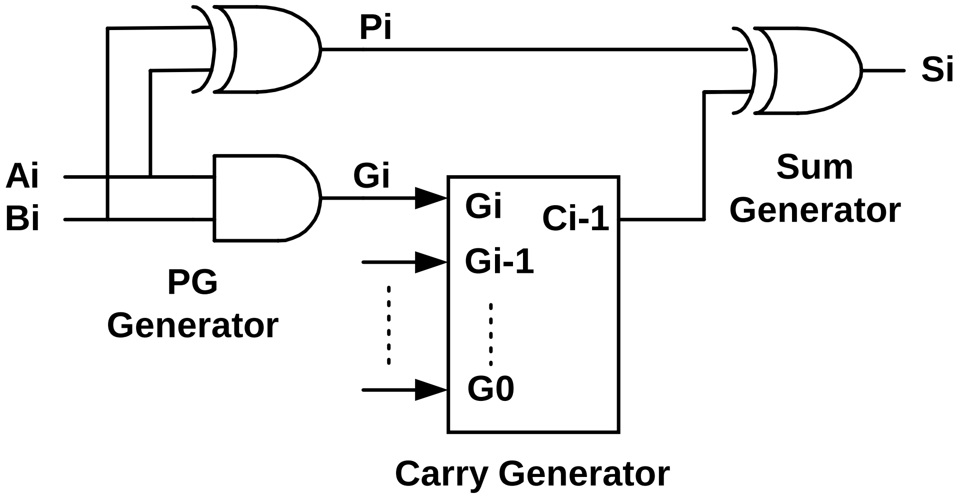
$$C_i = G_i + P_i \cdot C_{i-1}$$

$$\Downarrow$$

$$C_i = G_i + P_i \cdot G_{i-1} + P_{i-1} P_i \cdot G_{i-2} + \ldots + P_i \ldots P_0 \cdot C_{in}$$

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

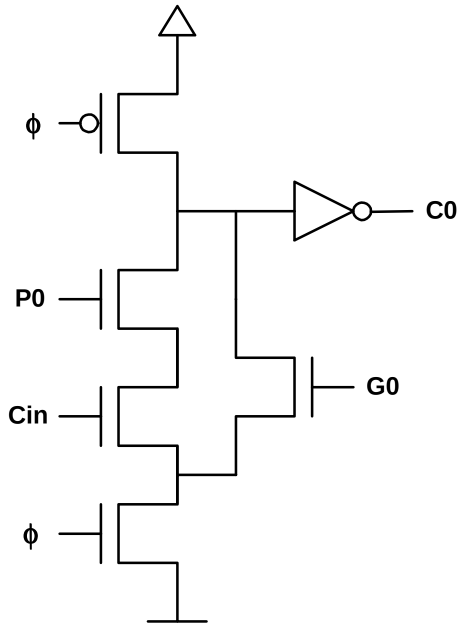
$$S_i = P_i \oplus C_{i-1} \quad if \quad P_i = A_i \oplus B_i$$



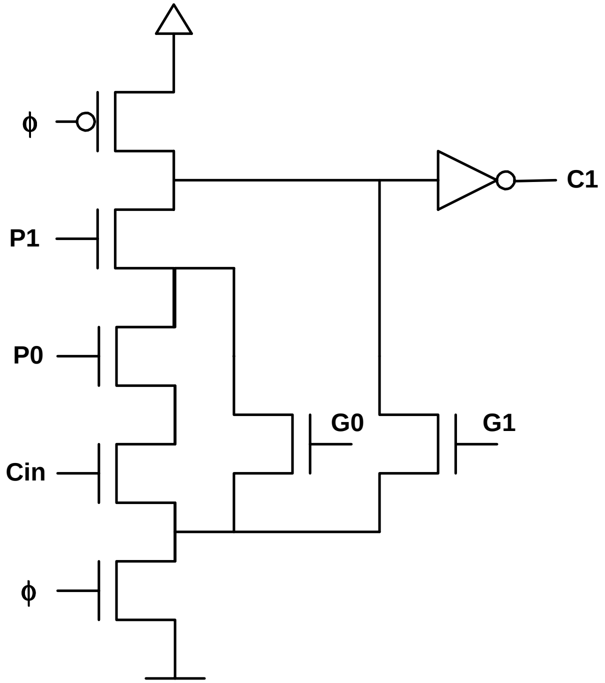
Addition/Subtraction

- Carry Generator by Dynamic Logic

$$C_i = G_i + P_i \cdot G_{i-1} + P_{i-1}P_i \cdot G_{i-2} + \ldots + P_i \ldots P_0 \cdot C_{in}$$



C0 generation

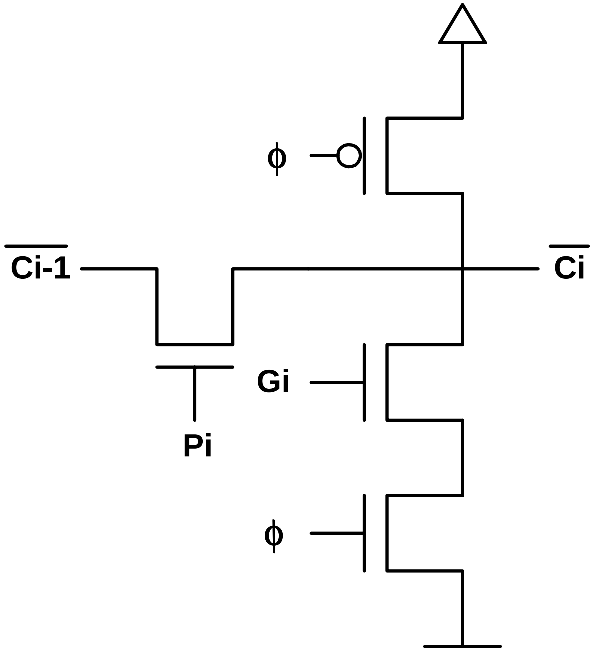


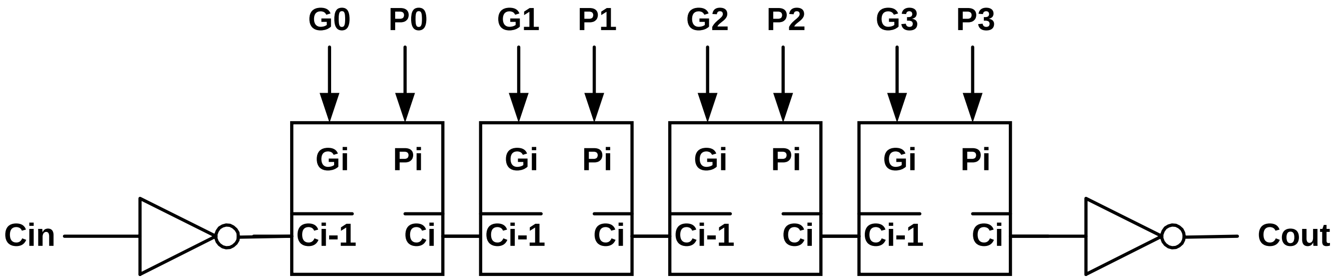
C1 generation

Addition/Subtraction

- Manchester Carry Chain

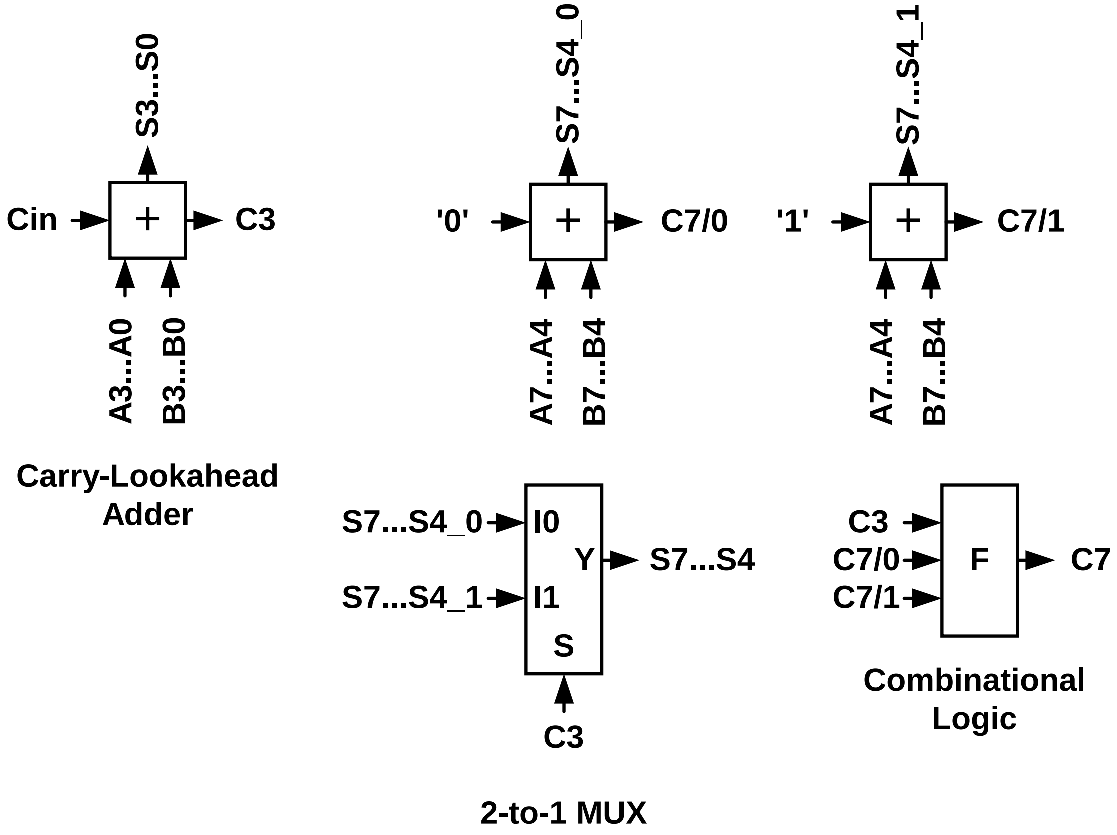
$$C_i = G_i + P_i \cdot C_{i-1}$$





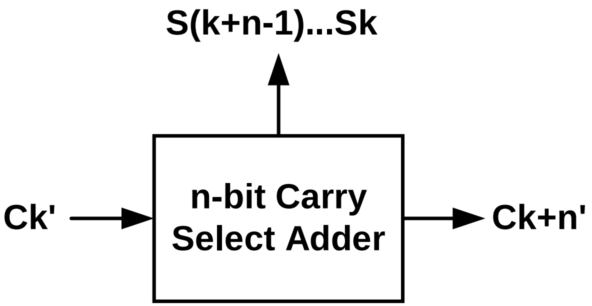
Addition/Subtraction

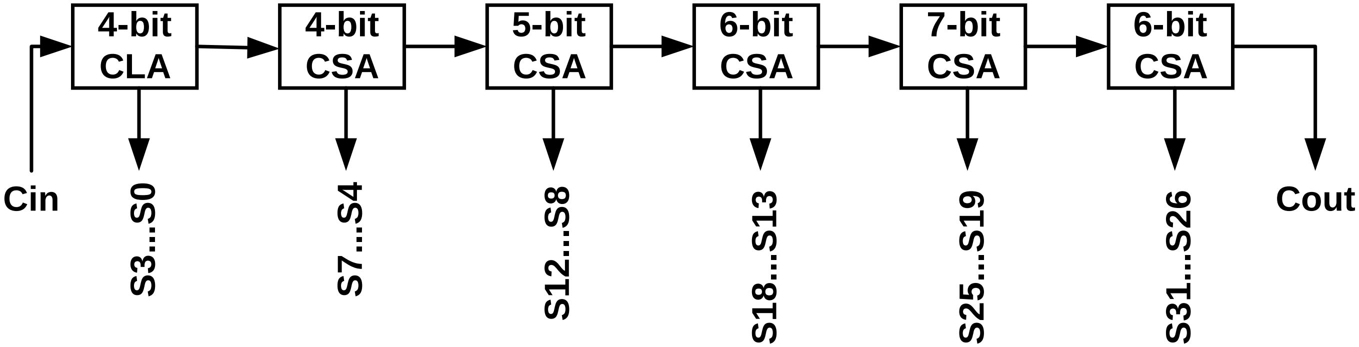
- Carry Select Adder



Addition/Subtraction

- Example: 32-bit Carry Select Adder
  - Delay = Delay_CLA + 5 x gate_delay



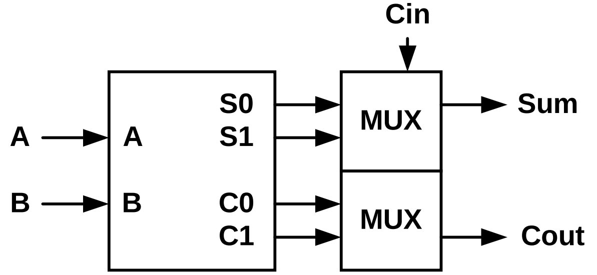


Addition/Subtraction

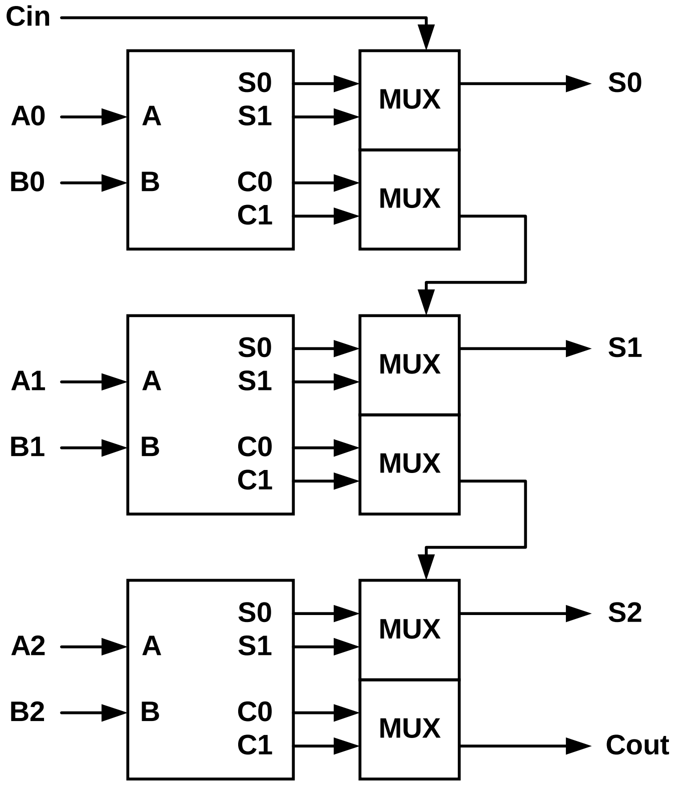
- Conditional Sum Adder

$$SUM == \begin{Bmatrix} A \oplus B & when & C_{in} = 0 \\ \overline{A \oplus B} & when & C_{in} = 1 \end{Bmatrix}$$

$$CARRY = \begin{Bmatrix} A \cdot B & when & C_{in} = 0 \\ A + B & when & C_{in} = 1 \end{Bmatrix}$$



- Very Wide Adders
    - Adders with word width greater than 32 bits can be constructed by applying hierarchy and modularity by combining adders with smaller widths.



Data-Path Operators

- Parity Generators
  - Detects whether the number of ones in an input word is even or odd.

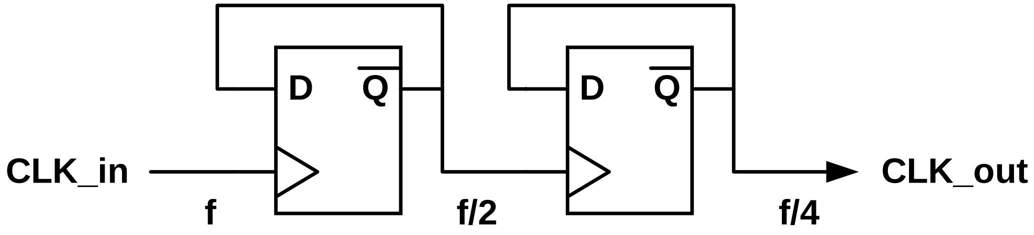
$$PARITY = A_0 \oplus A_1 \oplus A_2 \oplus \cdots \oplus A_n$$

- Comparators
  - Compares the magnitute of two binary numbers.

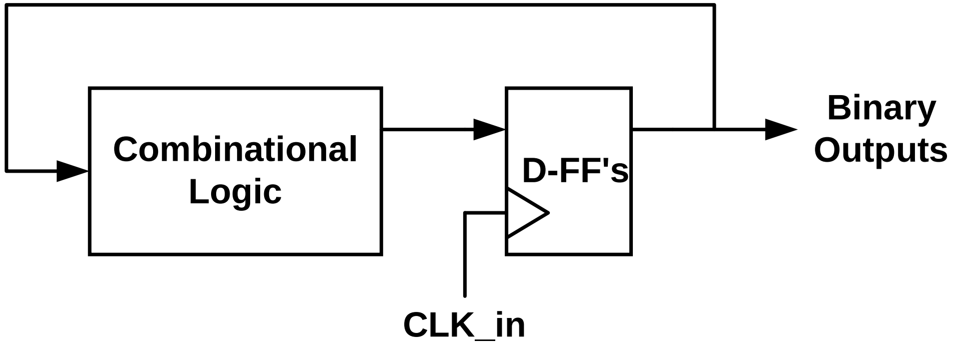
- Zero/One Detectors
  - Detects all ones/zeros on a word.

Data-Path Operators

- Binary Counters
  - Asynchronous counters
  - Synchronous counters



Asynchronous



Synchronous

Data-Path Operators

- Boolean Operations - ALUs
  - Arithmetic (addition/subtraction) and Boolean operations are performed.
  - Any Boolean function can be implemented by using TGs and MUXes.
- Multiplication
  - X is m-bit unsigned binary number
  - Y is n-bit unsigned binary number

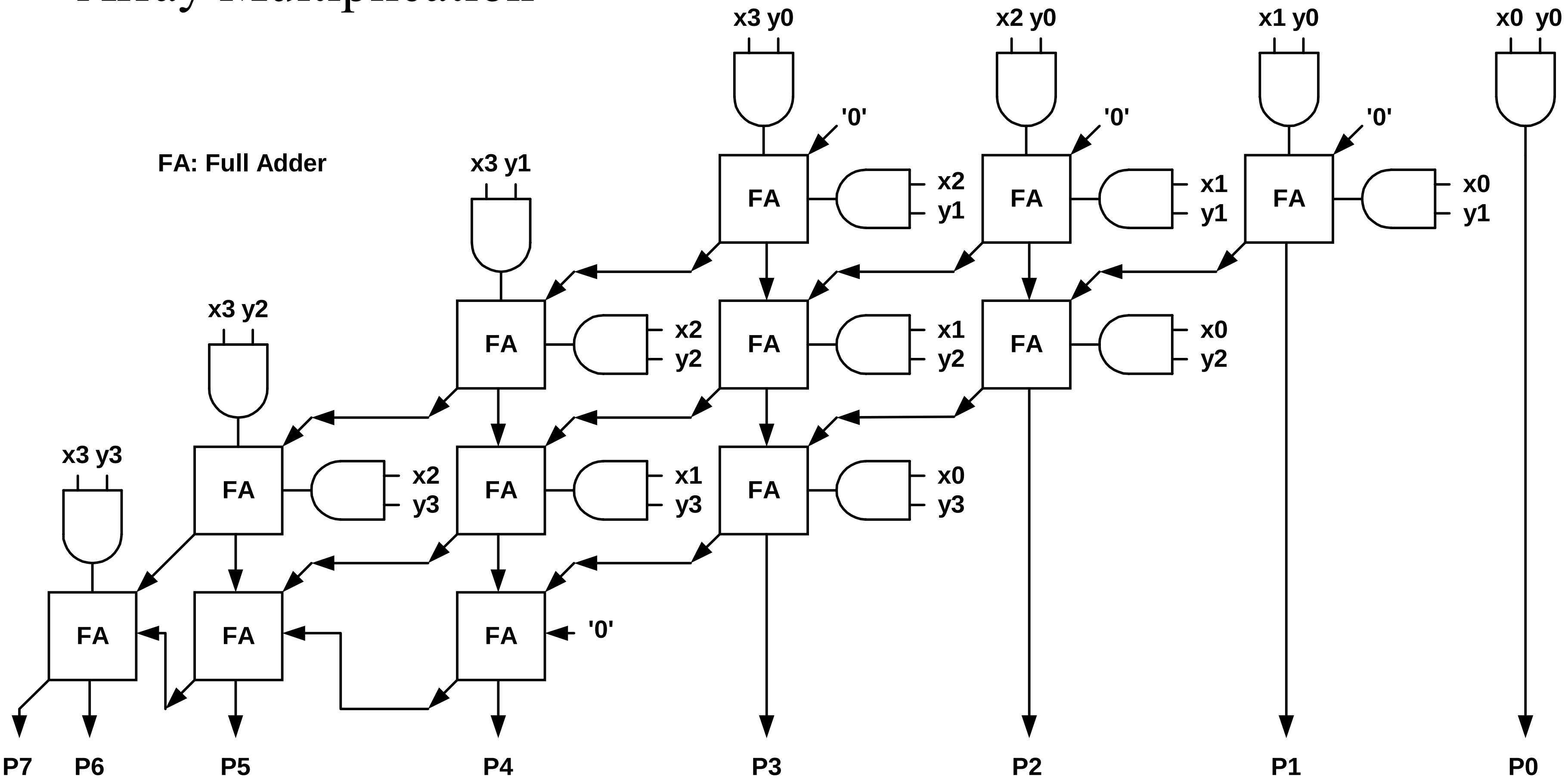
$$X = \sum_{i=0}^{m-1} x_i 2^i$$

$$Y = \sum_{j=0}^{n-1} y_j 2^j$$

$$P = X \times Y = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_i y_j 2^{i+j}$$

Data-Path Operators

- Array Multiplication



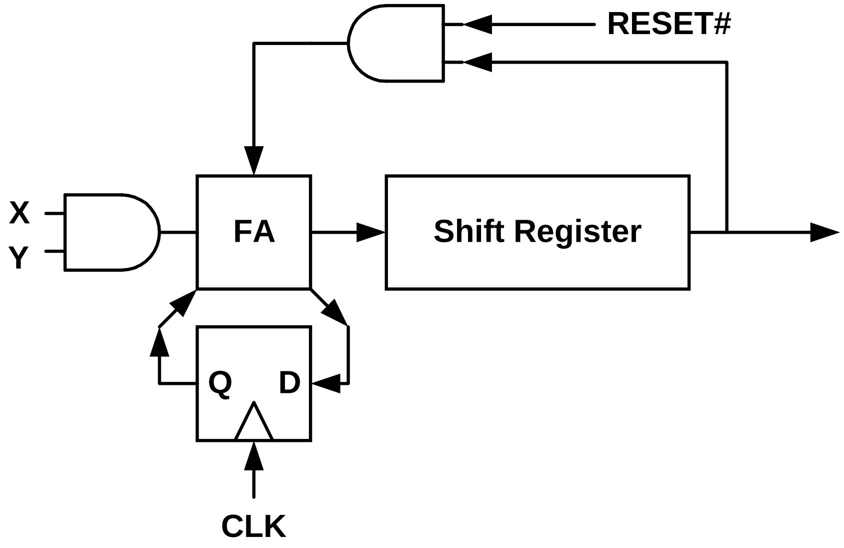
Multiplication

- Radix-n Multiplication
  - Aim is to reduce adders and delay.
  - Booth-recorded multiplier is a Radix-4 multiplication scheme.

- Wallace Tree Multiplication
  - Adder is a one’s counter circuit.
  - 1-bit adder provides 3:2 compression in the number of bits.
  - The addition of partial products in a column of an array multiplier is actually a counting of number of ones with carry.
  - An adder that provides 5:3 compression is often used.

Multiplication

- Serial Multiplication
  - X and Y are input serially at different rates.
  - Partial products are accumulated and stored in the shift register
  - Reset input is used to reset the partial sum at the beginning of multiplication cycle.
  - Product of m+n bits are obtained in mxn cycles.



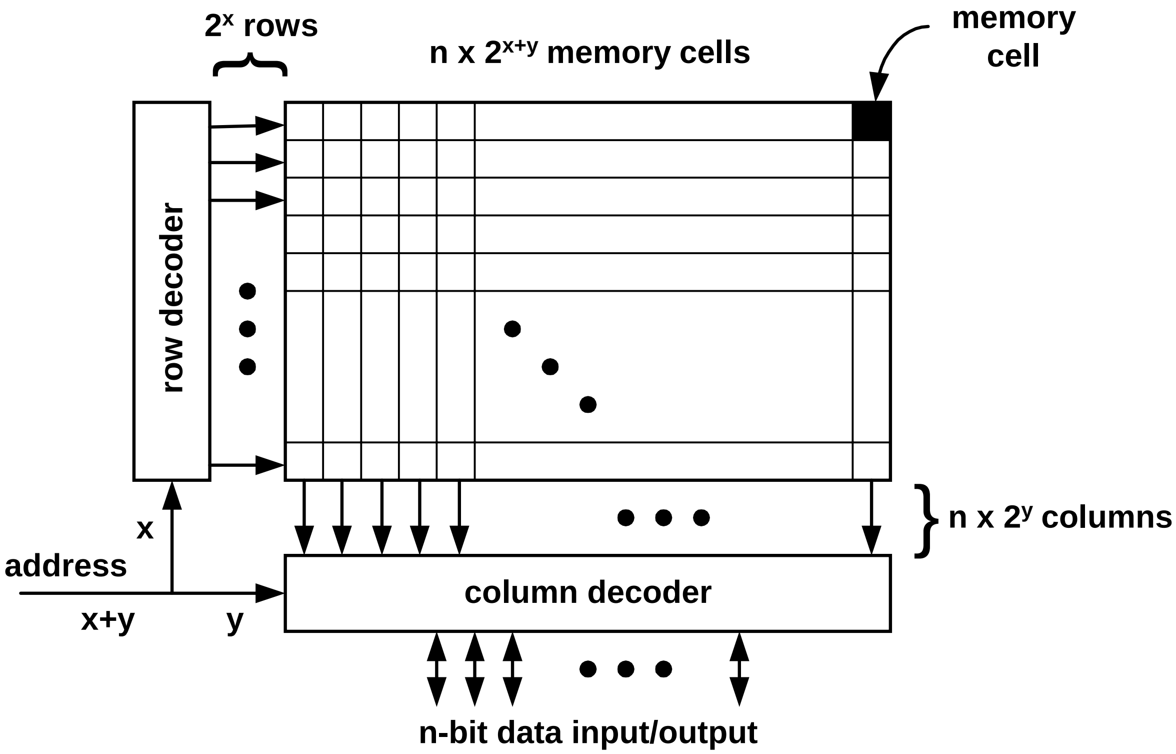
Data-Path Operators

- Serial/Parallel Multiplication
  - Product of m+n bits are obtained in m+n cycles.

- Shifters
  - Arithmetic shifting
  - Logical shifting
  - Rotation functions
  - Shuffles
  - Bit reversals/interchanges

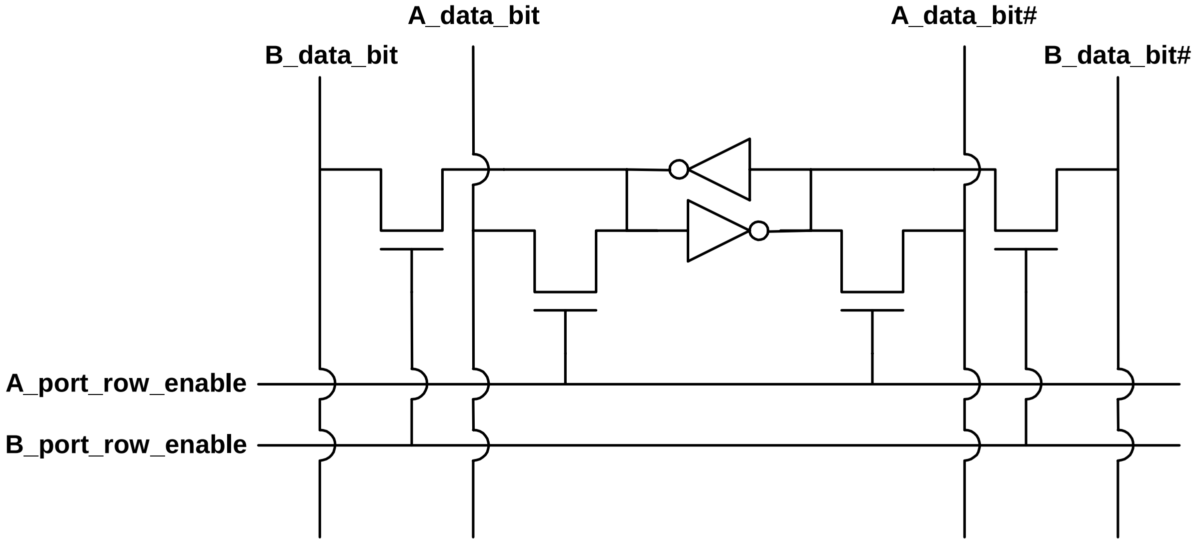
Memory Elements

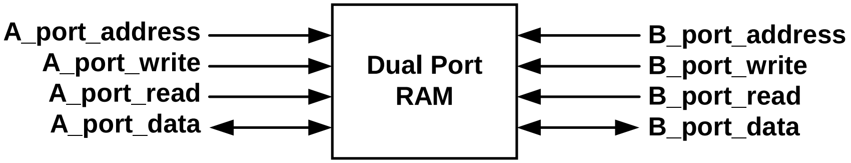
- Random Access Memory
  - Access time is independent of the physical location of the data.
  - Memory architecture:



Memory Elements

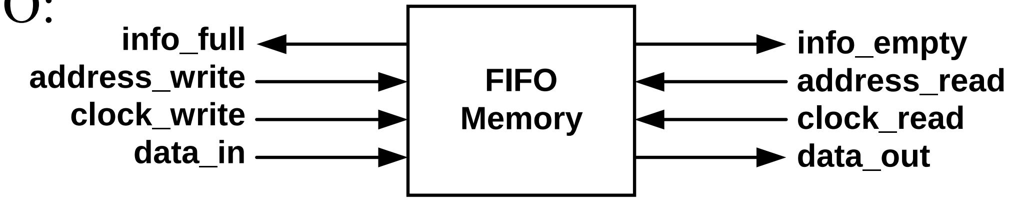
- Register Files
  - Fast RAMs with multiple read and write ports
  - Example: Dual port RAM memory cell circuitry



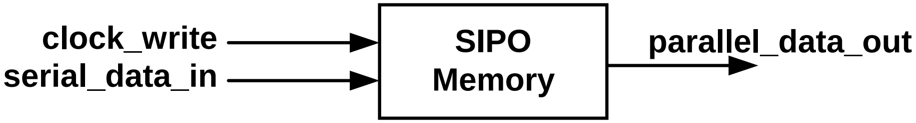


Memory Elements

- First-In-First-Out (FIFO) Memory
  - Useful in buffering data between two asynchronous data streams.
  - FIFO memory I/O:

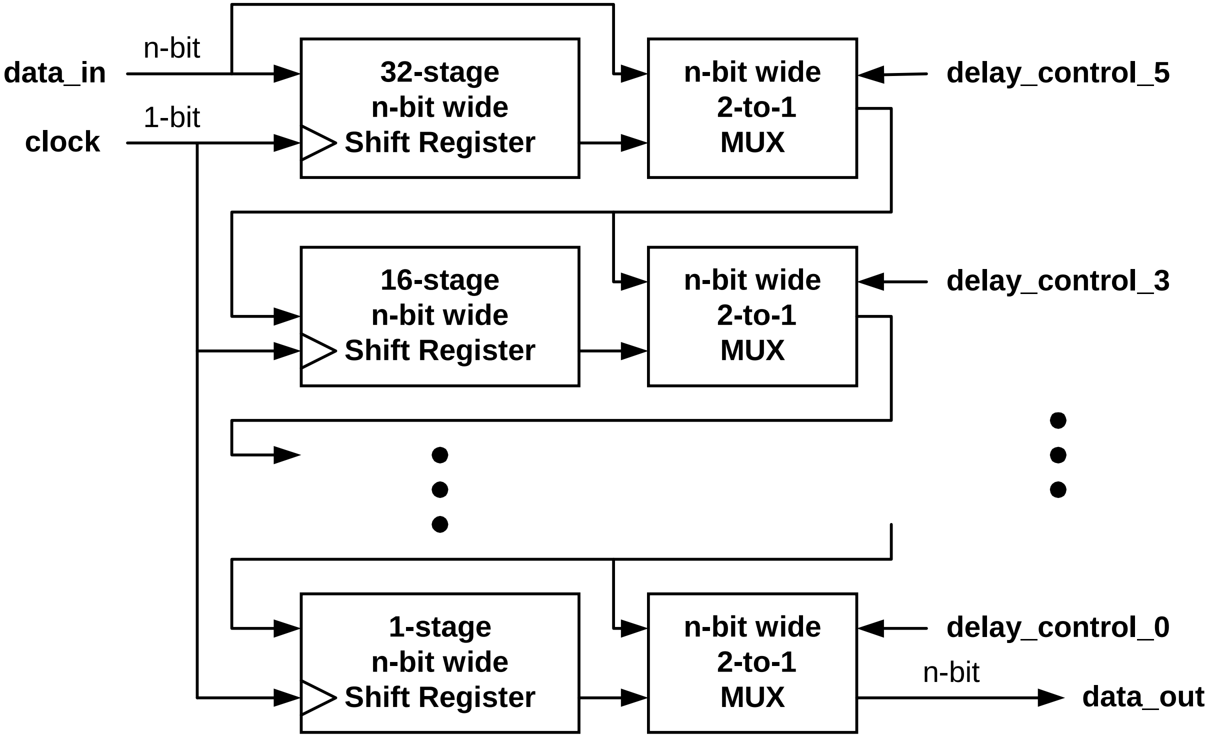


- Last-In-First-Out (LIFO) Memory
  - Used in sub-routine calls in uP (Stack).
- Serial-In-Parallel-Out (SIPO) Memory
  - Used to convert serial data to parallel form.



Memory Elements

- Serial-Access Memory
  - Used in signal processing for storage and delaying signals.
  - Example: Tapped delay line RTL description:



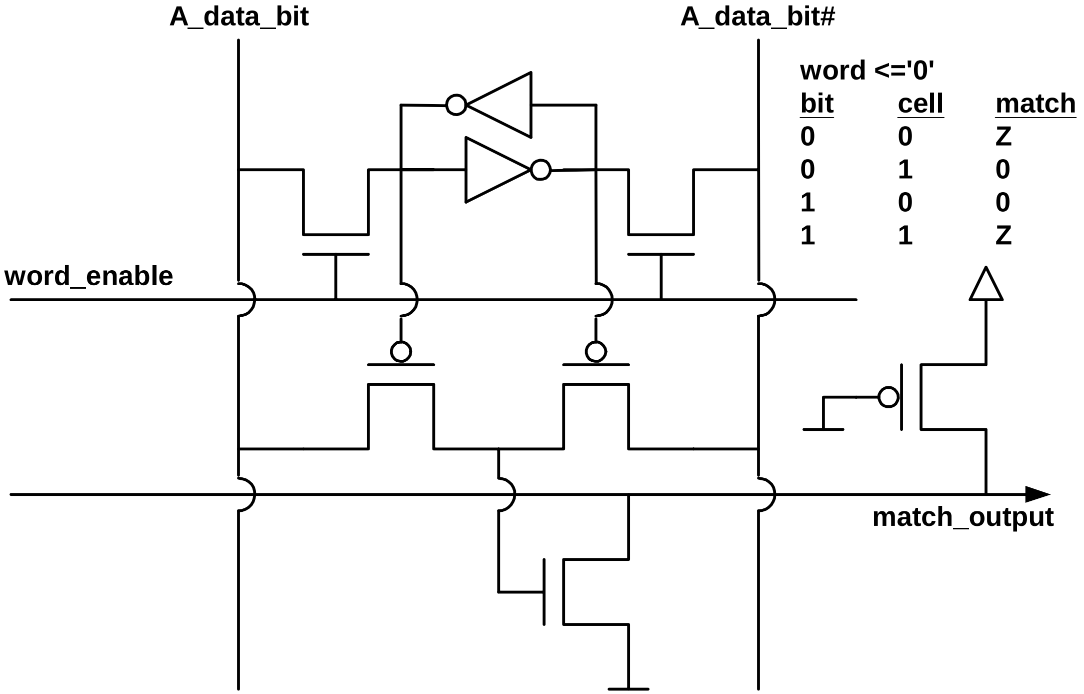
Memory Elements

- Read-Only Memory
  - Single MOSFET per bit of storage.
  - Static memory structure.
  - Bit storage even without power supply.
  - Usually implemented as a NOR array. NAND array is slow but less area.

- Content Addressable Memory
  - Compares a data word with the stored data words.
  - If one or more internal data match with the input data then internal data addresses are output as match outputs.

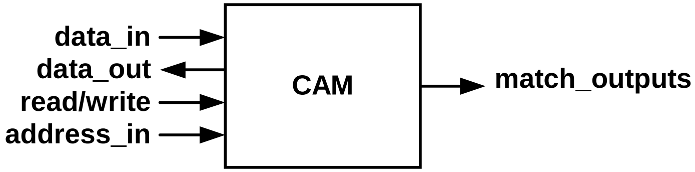
Memory Elements

- … Content Addressable Memory
  - CAM cell:



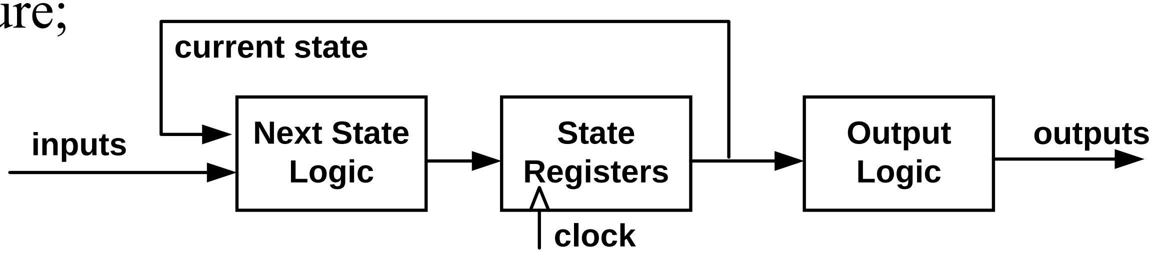
word <='0'

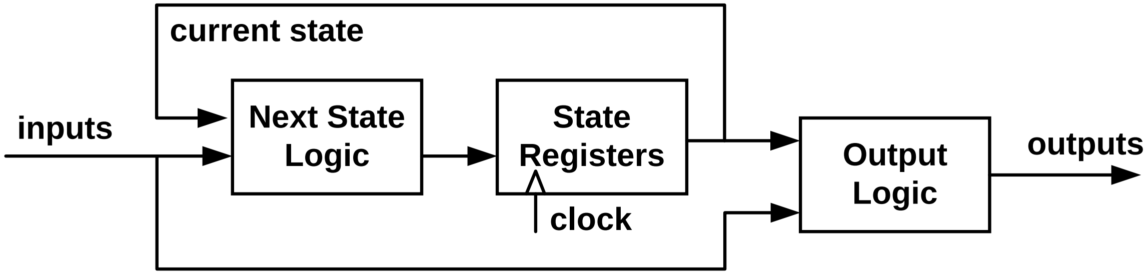
bit	cell	match
0	0	Z
0	1	0
1	0	0
1	1	Z



Control Structures

- Finite-State Machines
  - Organized structure for control sequencing/operation
  - Two basic types of state machines can be designed; Mealy, and Moore.
  - In Moore FSM structure, outputs are functions of state only.
  - In Mealy FSM structure, outputs are functions of both state and inputs.
  - Structure;





Control Structures

- FSM Design Procedure
  - Draw state-transition diagram
  - Check the diagram
  - Write state equations
  - Assign binary numbers to the states
  - Construct the resulting logic and registers.

- Control Logic Implementation
  - Two-level sum-of-products logic
    - Programmable Logic Array (PLA) is a regular structure to implement combinational and sequential logic functions.
    - Typical PLA uses two-level sum-of-products AND-OR structure.

Control Structures

- … Control Logic Implementation
  - Multi-Level logic
    - Common method for CMOS.
    - Regular gates are cascaded.
    - Many CAD tools minimize the logic for a set of Boolean functions.
    - FSMs can be directly synthesized form a high-level description.