PATRIoT lab

Lecture 15 – Sequential circuits

Dr. Aftab M. Hussain,

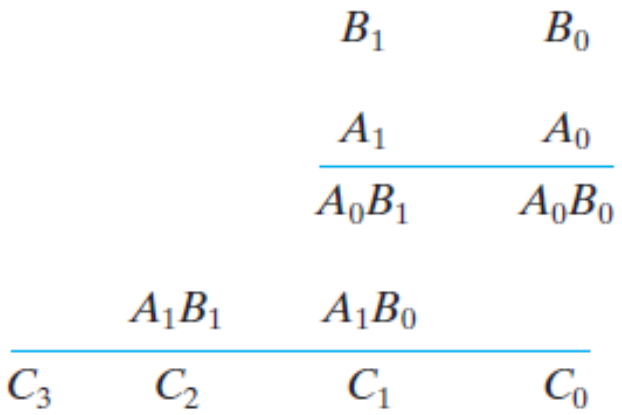
Assistant Professor, PATRIoT Lab, CVEST

Chapter 5

The Binary Multiplier

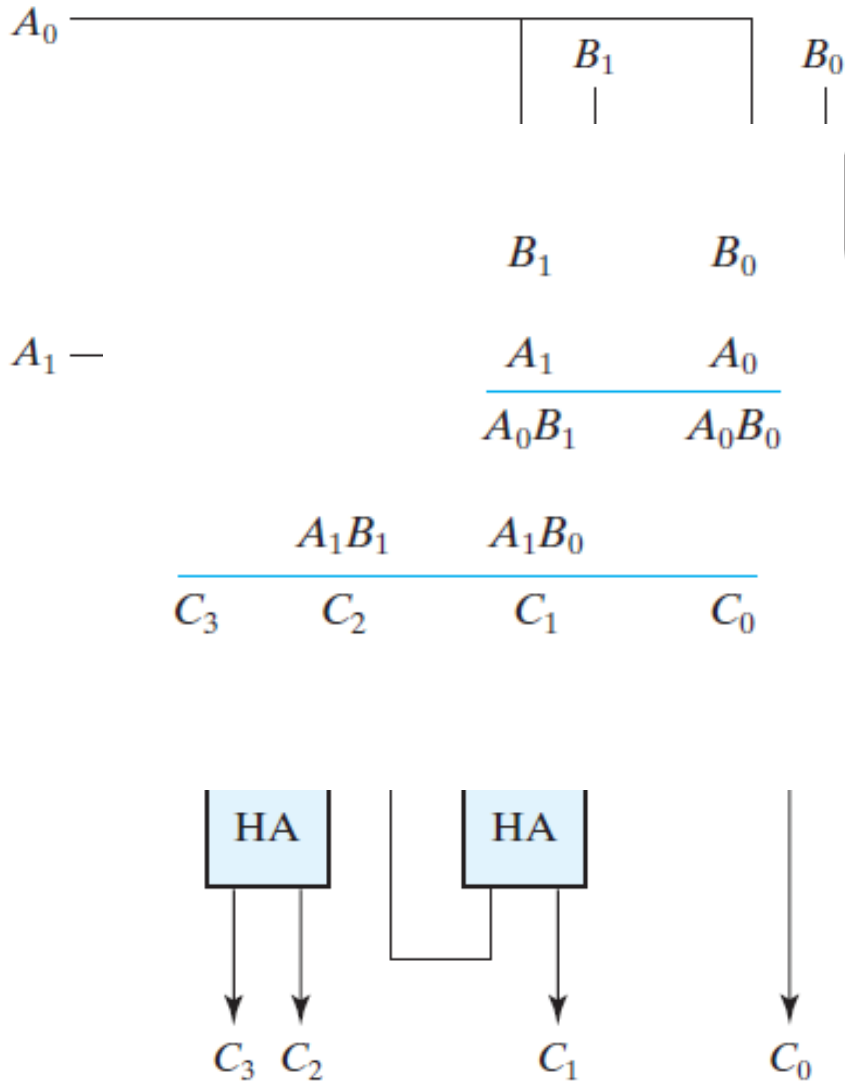
Binary multiplier

- Multiplication of binary numbers is performed in the same way as multiplication of decimal numbers
- The multiplicand is multiplied by each bit of the multiplier, starting from the least significant bit
- Each such multiplication forms a partial product
- Successive partial products are shifted one position to the left
- The final product is obtained from the sum of the partial products
- Consider a 2-bit x 2-bit multiplier. Number of inputs/outputs?



Binary multiplier

- The multiplicand bits are B_1 and B_0, the multiplier bits are A_1 and A_0, and the product is $C_3C_2C_1C_0$
- The first partial product is formed by multiplying B_1B_0 by A_0
- The multiplication of two bits such as A_0 and B_0 produces a 1 if both bits are 1; otherwise, it produces a 0
- This is identical to an AND operation
- Therefore, the partial products can be implemented with simple AND gates
- The two partial products are added with two half-adder (HA) circuits

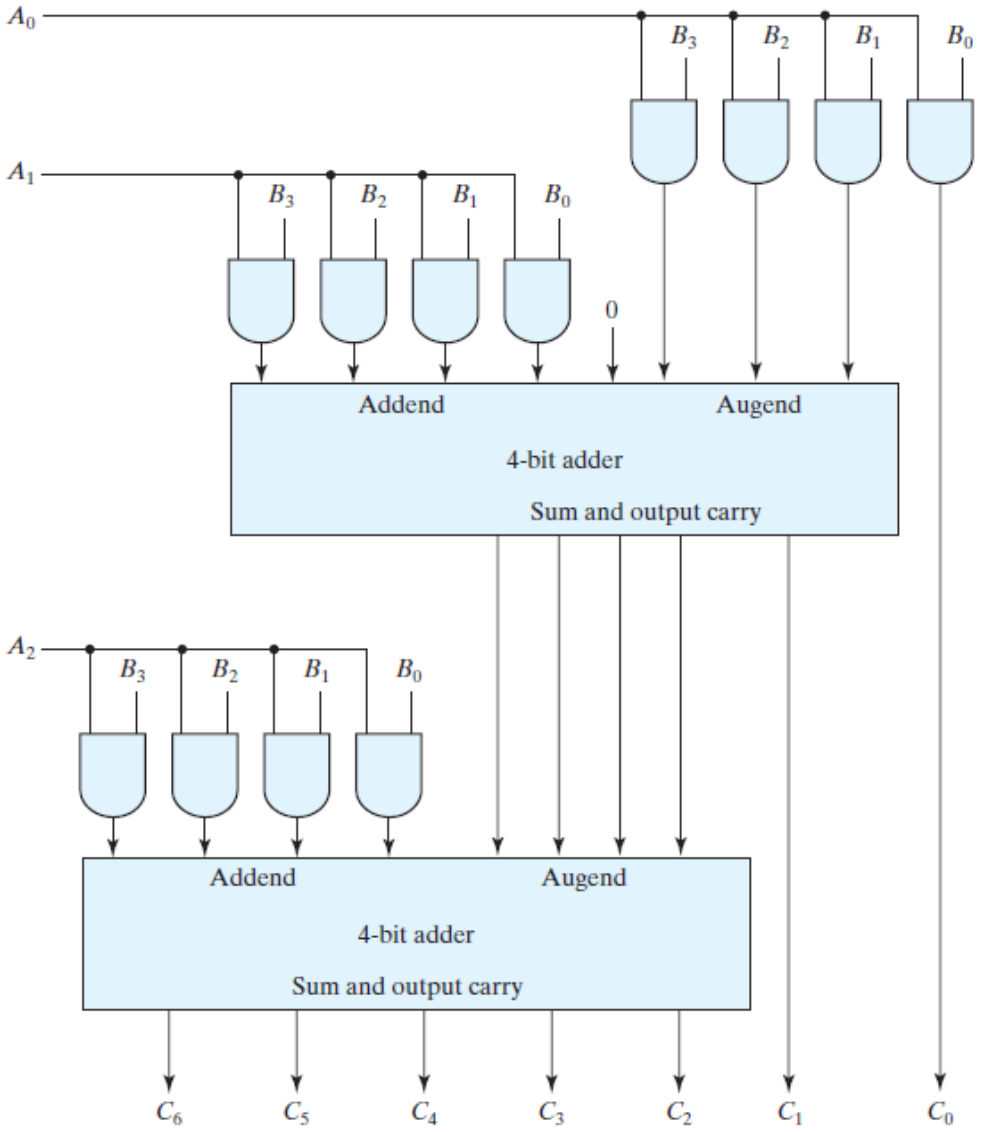


Binary multiplier

- A combinational circuit binary multiplier with more bits can be constructed in a similar fashion
- A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier
- The binary output in each level of AND gates is added with the partial product of the previous level to form a new partial product
- The last level produces the product
- For J multiplier bits and K multiplicand bits, we need $J * K$ AND gates and $(J – 1)$ K -bit adders to produce a product of $(J + K)$ bits
- Consider a multiplier circuit that multiplies a binary number represented by four bits by a number represented by three bits
- Let the multiplicand be represented by $B_3B_2B_1B_0$ and the multiplier by $A_2A_1A_0$

Binary multiplier

- A combinational circuit binary multiplier with more bits can be constructed in a similar fashion
- A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier
- The binary output in each level of AND gates is added with the partial product of the previous level to form a new partial product
- The last level produces the product
- For J multiplier bits and K multiplicand bits, we need $J * K$ AND gates and $(J - 1)$ K -bit adders to produce a product of $(J + K)$ bits
- Consider a multiplier circuit that multiplies a binary number represented by four bits by a number represented by three bits
- Let the multiplicand be represented by $B_3B_2B_1B_0$ and the multiplier by $A_2A_1A_0$



The Binary Comparator

Binary comparator

- The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number
- A *magnitude comparator* is a combinational circuit that compares two numbers *A* and *B* and determines their relative magnitudes
- The outcome of the comparison is specified by three binary variables that indicate whether $A > B$, $A = B$, or $A < B$
- On the one hand, the circuit for comparing two n -bit numbers has 2^{2n} entries in the truth table and becomes too cumbersome, even with $n = 3$
- On the other hand, as one may suspect, a comparator circuit possesses a certain amount of regularity

Binary comparator

- Consider two numbers, *A* and *B* , with four digits each $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$
- The two numbers are equal if all pairs of significant digits are equal: $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$, **and** $A_0 = B_0$
- To check bit-wise equality, we can use the XNOR gate

$$x_i = A_iB_i + A_i'B_i' \text{ for } i = 0, 1, 2, 3$$

- For equality to exist, all x_i variables must be equal to 1, a condition that dictates an AND operation of all variables:

$$(A = B) = x_3x_2x_1x_0$$

Binary comparator

- To determine whether *A* is greater or less than *B*, we inspect the relative magnitudes of pairs of significant digits, starting from the most significant position
- If the two digits of a pair are equal, we compare the next lower significant pair of digits
- The comparison continues until a pair of unequal digits is reached
- If the corresponding digit of *A* is 1 and that of *B* is 0, we conclude that *A* > *B*. If the corresponding digit of *A* is 0 and that of *B* is 1, we have *A* < *B*
- The comparison can be expressed logically by the two Boolean functions:

$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$
$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

Binary comparator

$x_i = A_i B_i + A_i' B_i'$ for i = 0, 1, 2, 3

$(A = B) = x_3 x_2 x_1 x_0$

$(A > B)$

$= A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1'$

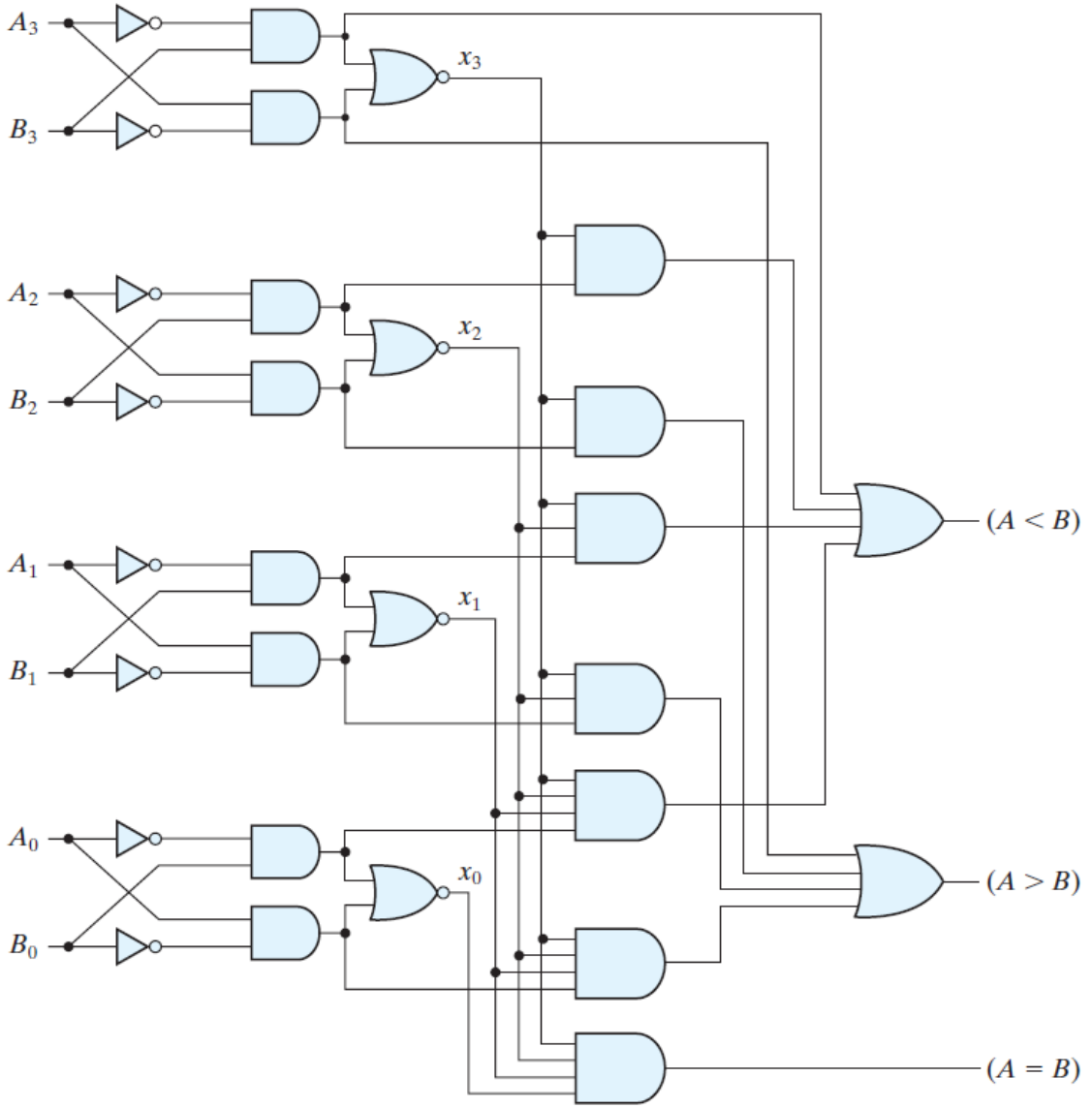
$+ x_3 x_2 x_1 A_0 B_0'$

$(A < B)$

$= A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1$

$+ x_3 x_2 x_1 A_0' B_0$

Interesting: Can we prove that only one of (A=B), (A>B) and (A<B) will be "1" at any given time?



PATRIoT lab

Sequential circuits

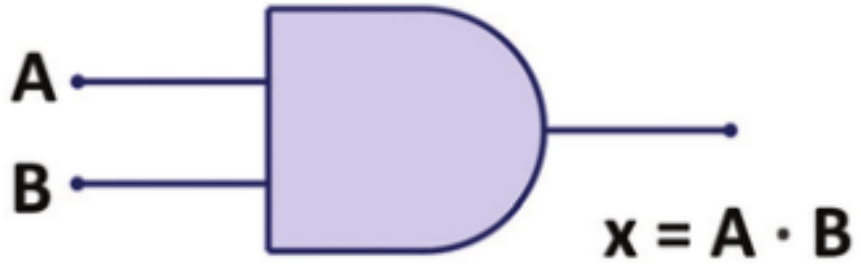
Dr. Aftab M. Hussain,

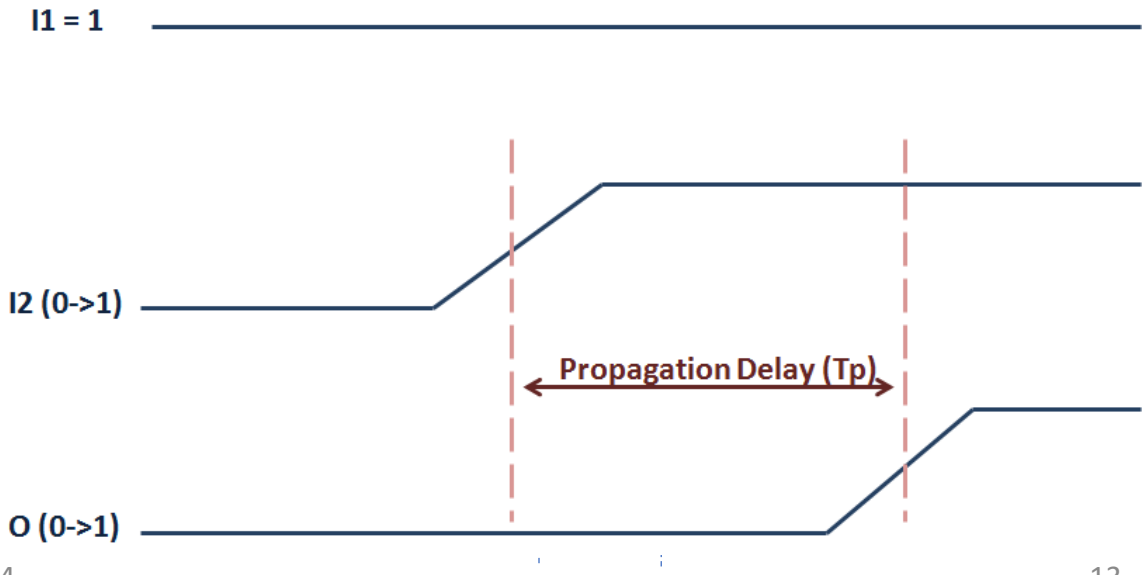
Assistant Professor, PATRIoT Lab, CVEST

Chapter 5

Gate delays

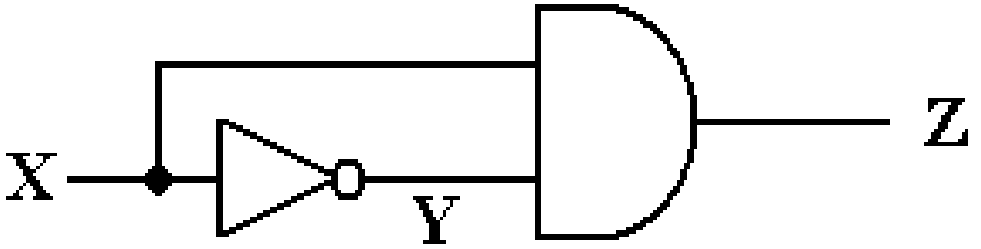
- When we apply an input to a gate, the output does not change immediately – there is a delay between the input application and appearance of the correct output
- This delay is generally of the order of nanoseconds, but can add up as signal goes through multiple gates
- *While making a digital IC design, gate delay (and by extension timing) is the single biggest consideration of all time!*
- Other things to worry about are silicon real estate, power consumption, reliability, testability, etc.

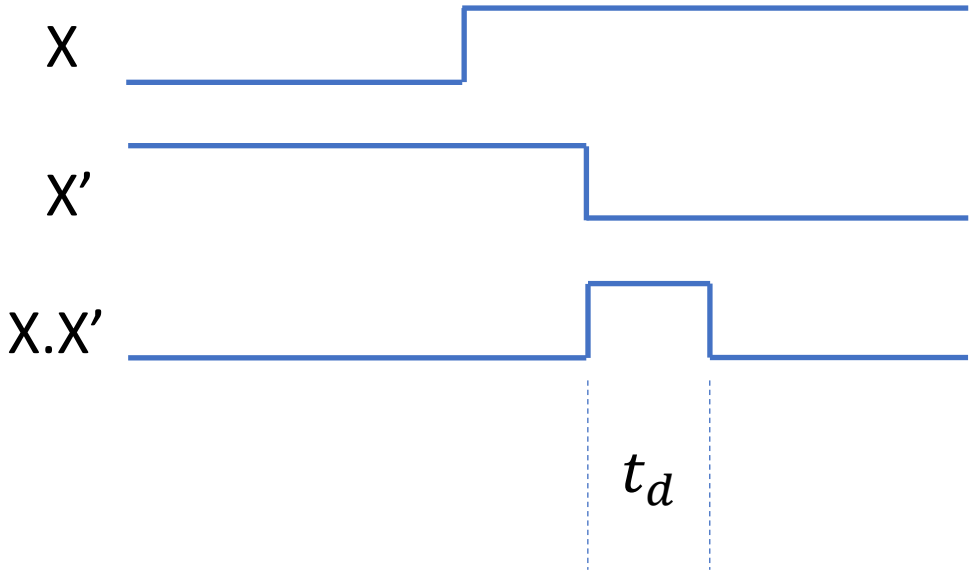




Glitches

- Improperly handled timing can lead to glitches and cause unexpected results
- This is particularly true for multi-level logic implementation wherein different literals are bypassing some levels
- Consider an AND gate connected to X and X’
- Such a connection will ALWAYS produce a glitch for X (0->1)
- You can be sad about this glitch…
- Or you can call this circuit a “pulse generator”! ☺





Ring oscillators

- Another clever use of gate delays in is in the construction of ring oscillators
- It is a series of odd number of NOT gates with the output connected back to the input gate of the first gate
- With this system, we can generate a continuous square wave at Q
- What is the frequency of the wave?
- In this case, it is $f = \frac{1}{T} = \frac{1}{3 \times 2t_d}$