Tutorial 5

1. a. Explain full adder. Design its truth table.
   b. Express sum and carry in terms of mean terms and max terms.
   c. Design its ckt. Diagram using basic logic gates.
   d. Implement full adder using half adder.

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to the another.

### Full Adder Truth Table:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Logical Expression for SUM:

= A' B' C-IN + A' B C-IN' + A B' C-IN' + A B C-IN
= C-IN (A' B' + A B') + C-IN' (A' B + A B')
= C-IN XOR (A XOR B)
= (1, 2, 4, 7)

### Logical Expression for C-OUT:

= A' B C-IN + A B' C-IN + A B C-IN'
= A B + B C-IN + A C-IN
= (3, 5, 6, 7)

### Another form in which C-OUT can be implemented:

= A B + A C-IN + B C-IN' (A + A')
= A B C-IN + A B + A C-IN + A B C-IN
= A B (1+C-IN) + A C-IN + A' B C-IN
= A B + A C-IN + A' B C-IN
= A B + A C-IN (B + B') + A' B C-IN
= A B C-IN + A B + A B' C-IN + A' B C-IN
= A B (C-IN + 1) + A B' C-IN + A' B C-IN

---

[Diagram of Full Adder with inputs A, B, and C-IN, and outputs Sum and C-OUT]
\[ A \times B + A \times B' + C \times \text{IN} + A' \times B \times \text{IN} \]

Therefore, \( C_{OUT} = AB + C_{IN} (A' \times B + A \times B') \)

Implementation of Full Adder using Half Adders

2 Half Adders and a OR gate is required to implement a Full Adder.

Full Subtractor in Digital Logic

2.

a. Explain full substractor. Design its truth table.

b. Express sum and carry in terms of mean terms and max terms.

c. Design its ckt. Diagram using basic logic gates.

d. Implement full adder using half adder.
A full subtractor is a **combinational circuit** that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. This circuit **has three inputs and two outputs**. The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrow, respectively.

**Truth Table** –

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

From above table we can draw the K-Map as shown for “difference” and “borrow”.

**Logical expression for difference** –

\[
D = A'B'Bin + A'BBin' + AB'Bin' + ABBin
\]

\[
= Bin(A'B' + AB) + Bin'(AB' + A'B)
\]

\[
= Bin(A \text{ XNOR } B) + Bin'(A \text{ XOR } B)
\]

\[
= Bin(A \text{ XNOR } B)' + Bin'(A \text{ XOR } B)
\]

\[
= Bin \text{ XOR } (A \text{ XOR } B)
\]

\[
= (A \text{ XOR } B) \text{ XOR } Bin
\]
Logical expression for borrow –
Bout = A’B’Bin + A’BBin + ABBin

\[= A’B’Bin + A’BBin’ + A’BBin + A’BBin + ABBin\]

\[= A’Bin(B + B’) + A’B(Bin + Bin’) + BBin(A + A’)\]

\[= A’Bin + A’B + BBin\]

OR

Bout = A’B’Bin + A’BBin + A’BBin + ABBin

\[= Bin(AB + A’B’) + A’B(Bin + Bin’)\]

\[= Bin(A \text{ XNOR } B) + A’B\]

\[= Bin(A \text{ XOR } B)’ + A’B\]

**Logic Circuit for Full Subtractor –**

![Logic Circuit for Full Subtractor](image)

**Implementation** of Full Subtractor using Half Subtractors –
2 Half Subtractors and an OR gate is required to implement a Full Subtractor.

![Logic Circuit for Half Subtractor](image)

3.

a) **Explain working principle of parallel adder and parallel substractor with proper ckt. diagram using full adders.**

b) **Explain its advantages and disadvantages**

**Parallel Adder** –
A single full adder performs the addition of two one bit numbers and an input carry. But a **Parallel Adder** is a digital circuit capable of finding the arithmetic sum of two binary numbers that is greater than one bit in length by operating on corresponding pairs of bits in parallel. It consists of full adders connected in a chain where the output carry from each full adder is connected to the carry input of the next higher order full adder in the chain. A **n bit parallel adder requires n full adders to perform the operation.** So for the two-bit number, two adders are needed while for four bit number, four adders are needed and so on. Parallel adders normally incorporate carry lookahead logic to ensure that carry propagation between subsequent stages of addition does not limit addition speed.
Working of Parallel Adder –
1. As shown in the figure, firstly the full adder FA1 adds A1 and B1 along with the carry C1 to generate the sum S1 (the first bit of the output sum) and the carry C2 which is connected to the next adder in chain.
2. Next, the full adder FA2 uses this carry bit C2 to add with the input bits A2 and B2 to generate the sum S2 (the second bit of the output sum) and the carry C3 which is again further connected to the next adder in chain and so on.
3. The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and Bn to generate the last bit of the output along last carry bit Cout.

Parallel Subtractor –
A Parallel Subtractor is a digital circuit capable of finding the arithmetic difference of two binary numbers that is greater than one bit in length by operating on corresponding pairs of bits in parallel. The parallel subtractor can be designed in several ways including combination of half and full subtractors, all full subtractors or all full adders with subtrahend complement input.

Working of Parallel Subtractor –
1. As shown in the figure, the parallel binary subtractor is formed by combination of all full adders with subtrahend complement input.
2. This operation considers that the addition of minuend along with the 2’s complement of the subtrahend is equal to their subtraction.
3. Firstly the 1’s complement of B is obtained by the NOT gate and 1 can be added through the carry to find out the 2’s complement of B. This is further added to A to carry out the arithmetic subtraction.
4. The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and 2’s complement of Bn to generate the last bit of the output along last carry bit Cout.

Advantages of Parallel Adder/Subtractor –
1. The parallel adder/subtractor performs the addition operation faster as compared to serial adder/subtractor.
2. Time required for addition does not depend on the number of bits.
3. The output is in parallel form i.e all the bits are added/subtracted at the same time.
4. It is less costly.

Disadvantages of Parallel Adder/Subtractor –
1. Each adder has to wait for the carry which is to be generated from the previous adder in chain.
2. The propagation delay (delay associated with the travelling of carry bit) is found to increase with the increase in the number of bits to be added.

4. 
   a) Explain the motivation behind carry look ahead adder.
   b) Explain carry look ahead adder.
   c) What do you mean by carry propagate and carry generate? Explain.
   d) Derive the expression for sum and carry in terms of carry propagate and carry generate for carry look ahead adder.
   e) Design ckt. diagram of carry look ahead adder.

**Carry look-ahead adder**

The ripple-carry adder, its limiting factor is the time it takes to propagate the carry. The carry look-ahead adder solves this problem by calculating the carry signals in advance, based on the input signals. The result is a reduced carry propagation time.

To be able to understand how the carry look-ahead adder works, we have to manipulate the Boolean expression dealing with the full adder. The Propagate P and generate G in a full-adder, is given as:

\[ P_i = A_i \oplus B_i \quad \text{Carry propagate} \]

\[ G_i = A_iB_i \quad \text{Carry generate} \]

Notice that both propagate and generate signals depend only on the input bits and thus will be valid after one gate delay.

The new expressions for the output sum and the carryout are given by:

\[ S_i = P_i \oplus C_{i-1} \]

\[ C_{i+1} = G_i + P_iC_i \]

These equations show that a carry signal will be generated in two cases:

1) If both bits \( A_i \) and \( B_i \) are 1

2) If either \( A_i \) or \( B_i \) is 1 and the carry-in \( C_i \) is 1.

Let's apply these equations for a 4-bit adder:

\[ C_1 = G_0 + P_0C_0 \]

\[ C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0 \]

\[ C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0 \]

\[ C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0 \]
These expressions show that C2, C3 and C4 do not depend on its previous carry-in. Therefore C4 does not need to wait for C3 to propagate. As soon as C0 is computed, C4 can reach steady state. The same is also true for C2 and C3.

The general expression is

\[ C_{i+1} = G_i + P_i G_{i-1} + P_i P_{i+1} G_{i-2} + \ldots \ldots \ldots P_i \ldots P_i P_2 G_0 + P_i P_{i-1} \ldots P_i P_0 C_0. \]

This is a two level circuit. In CMOS however the delay of the function is nonlinearly dependent on its fan in. Therefore large fanin gates are not practical.

Carry look-ahead adder’s structure can be divided into three parts: the propagate/generate generator Fig. 1, the sum generator Fig. 2 and the carry generator Fig. 3.
Advantages and Disadvantages of Carry Look-Ahead Adder:

**Advantages –**
- The propagation delay is reduced.
- It provides the fastest addition logic.

**Disadvantages –**
- The Carry Look-ahead adder circuit gets complicated as the number of variables increase.
- The circuit is costlier as it involves more number of hardware.
References for tutorial 5