Implementation of FULL & HALF SUBTRACTOR using VHDL

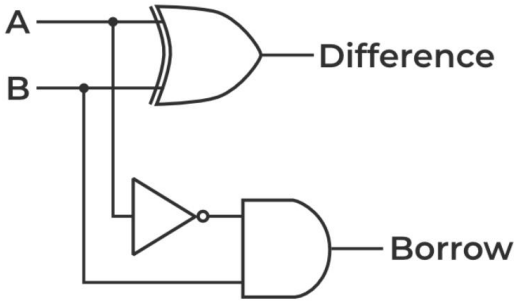
▪ OBJECTIVE:

To implement **Half & Full SUBTRACTOR** using VHDL and verify its waveform in Xilinx.

▪ INTRODUCTION:

1. Half Subtractor:

A **Half Subtractor** is a combinational logic circuit that performs the subtraction of two single-bit binary numbers. It has two inputs and two outputs.



Inputs:

1. A (minuend)
2. B (subtrahend)

Outputs:

1. **Difference (D):** The result of the subtraction A−B.
2. **Borrow (B_out):** Indicates if a borrow is needed when B>A.

Truth Table:

A	B	D (Difference)	B_{out} (Borrow)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Logic Expressions:

1. Difference (D):

$$D = A \oplus B$$

(Exclusive-OR operation)

2. Borrow (B_{out}):

$$B_{out} = \overline{A} \cdot B$$

(Logical AND of $\overline{A}$ and B)

Key Applications of Half Subtractor:

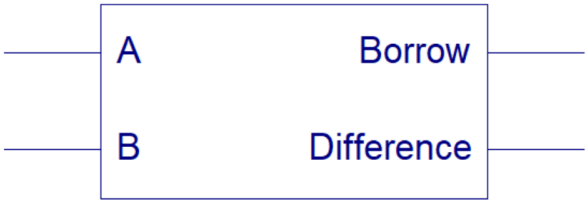
1. **Binary Subtraction**: Performs basic binary subtraction.
2. **ALU (Arithmetic Logic Unit)**: Enables subtraction in processors.
3. **Error Detection**: Helps in error checking in communication systems.
4. **Comparators**: Used in circuits for binary number comparison.
5. **Control Systems**: Calculates error signals in feedback loops.

It's a fundamental component in digital electronics and computational systems.

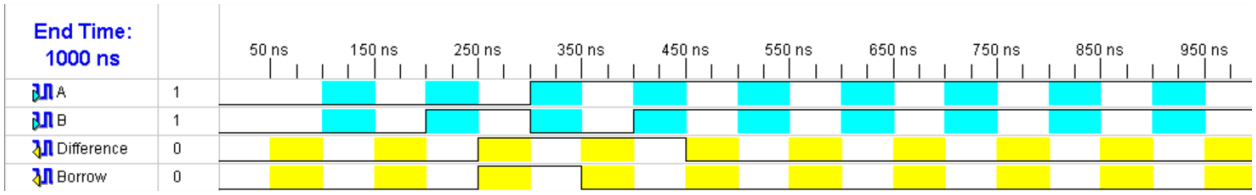
- ## Source Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Half_Subtractor is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           Difference : out  STD_LOGIC;
           Borrow : out  STD_LOGIC);
end Half_Subtractor;

architecture Behavioral of Half_Subtractor is

begin
    -- Logic for Difference
    Difference <= A XOR B;
    -- Logic for Borrow
    Borrow <= NOT A AND B;
end Behavioral;
```

- **RTL (Register Transfer Level) Schematic**:

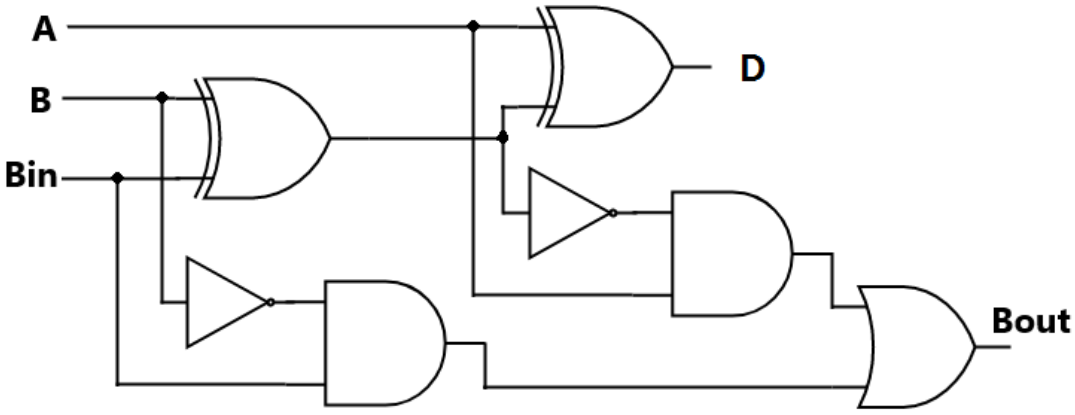


- **Test Bench Waveform:**



2. Full Subtractor:

A **Full Subtractor** is a combinational logic circuit that performs binary subtraction of three bits: the minuend (A), subtrahend (B), and a borrow from the previous stage (B_{in}). It produces two outputs: the difference (D) and the borrow (B_{out}).



Structure of a Full Subtractor

A Full Subtractor has three inputs and two outputs:

Inputs:

1. **A** (Minuend): The bit from which another bit is subtracted.
2. **B** (Subtrahend): The bit to be subtracted.
3. **B_{in}**(Borrow In): Borrow from the previous stage.

Outputs:

1. **D** (Difference): The result of $A-B-B_{in}$.
2. **B_{out}**(Borrow Out): Indicates if borrow is needed for the next stage when the subtraction result is negative.

Internal Logic:

1. Difference (D):
   - Calculated as:
$$D = (A \oplus B) \oplus B_{in}$$
2. Borrow (B_{out}):
   - Borrow is generated if:
     - B is greater than A (first half subtractor).
     - B_{in} is greater than the result of the first subtraction.
   - Final expression:
$$B_{out} = (\overline{A} \cdot B) + (\overline{A \oplus B} \cdot B_{in})$$

Applications of Full Subtractor:

1. **Binary Arithmetic**: Performs subtraction of three binary bits.
2. **ALU**: Key component in Arithmetic Logic Units for subtraction.
3. **Digital Computers**: Used in microprocessors for binary subtraction.
4. **Cascaded Subtraction**: Subtracts multi-bit binary numbers.
5. **Error Detection**: Assists in error-checking algorithms.
6. **Digital Signal Processing**: Used in image/audio processing.
7. **Control Systems**: Computes error signals in feedback systems.
8. **Binary Comparators**: Helps compare binary numbers

Truth Table:

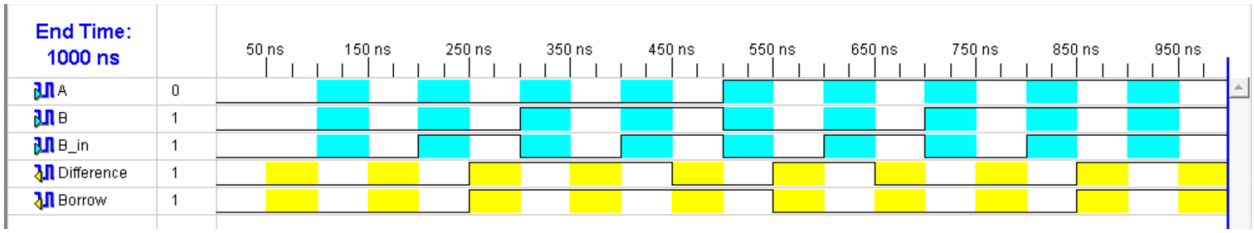
A	B	B_{in}	D (Difference)	B_{out} (Borrow)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

- **Source Code:**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Full_Subtractor is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           B_in : in  STD_LOGIC;
           Difference : out  STD_LOGIC;
           Borrow : out  STD_LOGIC);
end Full_Subtractor;s

architecture Behavioral of Full_Subtractor is

begin

    -- Logic for Difference
    Difference <= A XOR B XOR B_in;

    -- Logic for Borrow
    Borrow <= (NOT A AND B) OR (NOT A AND B_in) OR (B AND B_in);

end Behavioral;
```

- **RTL (Register Transfer Level) Schematic**:

- **Test Bench Waveform:**



▪ Discussion:

We designed and simulated half and full subtractors using VHDL in Xilinx. The half subtractor used XOR and AND gates, while the full subtractor combined two half subtractors with an OR gate for handling borrow input. Simulations confirmed the correctness of the circuits, aligning with expected truth tables.

▪ Conclusion:

The experiment successfully validated the design of both subtractors. The half subtractor handles two-bit subtraction, while the full subtractor supports multi-bit subtraction with borrow propagation. This lab reinforced our understanding of VHDL, combinational logic, and their applications in digital systems.