Assignment-07 (Block statements)

Use for_Generate

Ripple Carry Adder (Structural Design)

- Write a **1-bit full adder** module.
- Using a for-generate loop, instantiate N full adders to build an 8**-bit Ripple Carry Adder**.
- Inputs: A[7:0], B[7:0], cin
- Outputs: SUM[7:0], cout
- Test with **4-bit** and **8-bit** versions.

Bitwise AND Gate Array

- Write a module that performs bitwise **AND operation** between two N-bit inputs.
- Use for-generate to assign each output bit.
- Inputs: A[7:0], B[7:0]
- Output: Y[7:0]

N-bit 2:1 Multiplexer

- Design a **1-bit 2:1 MUX**.
- Use generate to construct an 8**-bit wide multiplexer**.
- Inputs: A[7:0], B[7:0], sel
- Output: Y[7:0]

8-bit Barrel shifter design

- Refer following code from website (https://vlsigyan.com/barrel-shifter-verilog-code/)
- Modify code using for_generate statement

Use If_Generate statement

Configurable Logic Gate

- Write a module that takes two inputs a and b.
- Using if-generate, implement either:
  - AND gate (if USE_AND=1), or
  - OR gate (if USE_AND=0).
- Inputs: a, b
- Output: y

Parameterized Adder/Subtractor

- Build an 8-bit unit that performs:
  - Addition (y = a + b) if OP=0.
  - Subtraction (y = a - b) if OP=1.
- Use if-generate to select which operator is instantiated.

Configurable Shift Unit

- If DIR=0 → implement **logical left shift**.
- If DIR=1 → implement **logical right shift**

Adder/Subtractor

- If a parameter is set, generate either adder logic or subtractor logic

Use Case Generate

Configurable Logic Unit

- Write a module that performs one of the following (selected by parameter OP):
  - OP=0: AND gate
  - OP=1: OR gate
  - OP=2: XOR gate
- Inputs: a, b
- Output: y

Arithmetic Unit

- Write a parameterized arithmetic unit with parameter FUNC.
- Based on FUNC:
  - FUNC=0: Addition (a+b)
  - FUNC=1: Subtraction (a-b)
  - FUNC=2: Multiplication (a*b)
  - FUNC=3: Division (a/b)
- Inputs: a, b
- Output: y

Configurable Counter

- Design a counter where parameter MODE selects the behavior:
    - MODE=0: Up counter
    - MODE=1: Down counter
- Inputs: clk, rst, up_down
- Output: q[N-1:0]