CLOCK GATING – A POWER OPTIMIZING TECHNIQUE FOR VLSI CIRCUITS

***VERILOG CODES*:-**

1) Without Gated Clock:
   - *Source Code* –

```
module ALU_System (
    input [3:0] A,         // 4-bit input A (now guaranteed to update)
    input [3:0] B,         // 4-bit input B
    input [2:0] opcode,    // Operation selector
    input clk,             // Global clock
    input reset_n,         // Active-low reset
    output reg [3:0] d_out, // Output register
    output carry_out       // Carry-out flag
);

    // Clock Divider (Reduced for simulation)
    parameter DIV_FACTOR = 5;  // Small value for quick simulation
    reg [31:0] counter;
    reg clk_divided;

    always @(posedge clk or negedge reset_n) begin
        if (!reset_n) begin
            counter <= 0;
            clk_divided <= 0;
        end else begin
            if (counter == DIV_FACTOR - 1) begin
                clk_divided <= ~clk_divided;
                counter <= 0;
            end else begin
                counter <= counter + 1;
            end
        end
    end

    // ALU Core Logic (Fixed to ensure A is processed)
    reg [4:0] alu_result;  // 5-bit to include carry

    always @(*) begin
        case (opcode)
```

```
            3'b000: alu_result = A + B;       // ADD
            3'b001: alu_result = A - B;       // SUB
            3'b010: alu_result = {1'b0, A & B}; // AND
            3'b011: alu_result = {1'b0, A | B}; // OR
            3'b100: alu_result = {1'b0, A ^ B}; // XOR
            3'b101: alu_result = {1'b0, ~A};    // NOT
            3'b110: alu_result = A << 1;       // SHL (MSB->carry)
            3'b111: alu_result = {1'b0, A[3], A[2:0]}; // SHR (LSB->carry)
            default: alu_result = 5'b0;        // Default
        endcase
    end

    assign carry_out = alu_result[4];  // Carry/borrow flag

    // Output register (synchronized to divided clock)
    always @(posedge clk_divided or negedge reset_n) begin
        if (!reset_n) begin
            d_out <= 0;
        end else begin
            d_out <= alu_result[3:0];  // Update only on clk_divided
        end
    end

    // Debugging: Monitor A and ALU operations
    initial begin
        $monitor("Time=%t: A=%b B=%b op=%b | alu_result=%b d_out=%b",
                 $time, A, B, opcode, alu_result, d_out);
    end

endmodule
```

- TestBench Code –

```
module ALU_System_tb();

    // Test signals
    reg [3:0] A, B;
    reg [2:0] opcode;
    reg clk, reset_n;
```

```
wire [3:0] d_out;
wire carry_out;

// Instantiate ALU with fast clock divider
ALU_System dut (
    .A(A),
    .B(B),
    .opcode(opcode),
    .clk(clk),
    .reset_n(reset_n),
    .d_out(d_out),
    .carry_out(carry_out)
);

// Clock generation (100 MHz)
initial begin
    clk = 0;
    forever #5 clk = ~clk;  // 10ns period
end

// Reset and test sequence
initial begin
    // Initialize and reset
    A = 0; B = 0; opcode = 0;
    reset_n = 0;  // Assert reset
    #20 reset_n = 1;  // Release reset

    // Test ADD (opcode=000)
    opcode = 3'b000;
    A = 4'b0011; B = 4'b0101;  #50;  // 3 + 5 = 8 (1000)
    A = 4'b1111; B = 4'b0001;  #50;  // 15 + 1 = 0 (carry=1)

    // Test SUB (opcode=001)
    opcode = 3'b001;
    A = 4'b1000; B = 4'b0011;  #50;  // 8 - 3 = 5 (0101)

    // Test SHL (opcode=110)
    opcode = 3'b110;
    A = 4'b0001;  #50;  // 1 << 1 = 2 (0010)
```

```
    // Terminate
    #100 $finish;
  end

  // Waveform dumping (for debugging)
  initial begin
    $dumpfile("alu_waveform.vcd");
    $dumpvars(0, ALU_System_tb);
  end

endmodule
```

- Constraints –

```
## Clock (100MHz on Nexys A7)
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add
[get_ports clk]

## Reset (BtnC on Nexys A7)
set_property -dict {PACKAGE_PIN D9 IOSTANDARD LVCMOS33} [get_ports reset_n]

## ALU Inputs A[3:0] (Switches 0-3)
set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {A[0]}]
set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {A[1]}]
set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {A[2]}]
set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {A[3]}]

## ALU Inputs B[3:0] (Switches 4-7)
set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {B[0]}]
set_property -dict {PACKAGE_PIN T18 IOSTANDARD LVCMOS33} [get_ports {B[1]}]
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {B[2]}]
set_property -dict {PACKAGE_PIN R13 IOSTANDARD LVCMOS33} [get_ports {B[3]}]

## Opcode Inputs [2:0] (Switches 8-10)
set_property -dict {PACKAGE_PIN T8 IOSTANDARD LVCMOS33} [get_ports
{opcode[0]}]
```

```
set_property -dict {PACKAGE_PIN U8 IOSTANDARD LVCMOS33} [get_ports
{opcode[1]}]
set_property -dict {PACKAGE_PIN R16 IOSTANDARD LVCMOS33} [get_ports
{opcode[2]}]

## Outputs d_out[3:0] (LEDs 0-3)
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports
{d_out[0]}]
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports
{d_out[1]}]
set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports
{d_out[2]}]
set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports
{d_out[3]}]

## Carry out (LED 4)
set_property -dict {PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports
carry_out]

set_operating_conditions -process maximum
```

2) With Gated Clock:-

- Source Code –

```
module ALU_System_Gated (
   input  wire [3:0] A,         // 4-bit input A
   input  wire [3:0] B,         // 4-bit input B
   input  wire [2:0] opcode,    // Operation selector
   input  wire       clk,       // Global clock
   input  wire       reset_n,   // Active-low reset
   output reg  [3:0] d_out,     // 4-bit ALU output
   output wire       carry_out, // Carry flag
   output wire       clk_out,   // Divided clock for reference
   output wire       gated_clk, // Gated clock output
   output wire       ce_sync    // Change enable sync signal
);

   // Clock Divider (Divide-by-5)
   parameter DIV_FACTOR = 5;
```

```
reg [31:0] counter;
reg clk_divided;

always @(posedge clk or negedge reset_n) begin
   if (!reset_n) begin
      counter <= 0;
      clk_divided <= 0;
   end else begin
      if (counter == DIV_FACTOR - 1) begin
         clk_divided <= ~clk_divided;
         counter <= 0;
      end else begin
         counter <= counter + 1;
      end
   end
end

assign clk_out = clk_divided;

// ALU Logic (5-bit for carry)
reg [4:0] alu_result;

always @(*) begin
   case (opcode)
      3'b000: alu_result = A + B;
      3'b001: alu_result = A - B;
      3'b010: alu_result = {1'b0, A & B};
      3'b011: alu_result = {1'b0, A | B};
      3'b100: alu_result = {1'b0, A ^ B};
      3'b101: alu_result = {1'b0, ~A};
      3'b110: alu_result = {A[3], A, 1'b0} >> 1; // SHL
      3'b111: alu_result = {A[0], A >> 1};       // SHR
      default: alu_result = 5'b00000;
   endcase
end

assign carry_out = alu_result[4];

// Change Detection
```

```
reg [3:0] prev_result;
reg change_detected;

always @(posedge clk_divided or negedge reset_n) begin
   if (!reset_n) begin
      prev_result <= 4'b0000;
      change_detected <= 1'b0;
   end else begin
      change_detected <= (alu_result[3:0] != prev_result);
      prev_result <= alu_result[3:0];
   end
end

// Clock Enable Sync
reg ce_reg;
always @(posedge clk_divided or negedge reset_n) begin
   if (!reset_n)
      ce_reg <= 0;
   else
      ce_reg <= change_detected;
end
assign ce_sync = ce_reg;

// Clock Gating using BUFGCE
(* DONT_TOUCH = "TRUE" *)
BUFGCE bufgce_inst (
   .I(clk_divided),
   .CE(ce_reg),
   .O(gated_clk)
);

// Output Register
always @(posedge gated_clk or negedge reset_n) begin
   if (!reset_n)
      d_out <= 4'b0000;
   else
      d_out <= alu_result[3:0];
end
```

```
    // Simulation Debug Monitor
    initial begin
        $monitor("Time=%t | A=%b B=%b opcode=%b | alu_result=%b carry=%b
d_out=%b ce=%b gated_clk=%b",
                 $time, A, B, opcode, alu_result, carry_out, d_out, ce_sync, gated_clk);
    end

    endmodule
```

- TestBench Code –

```
    `timescale 1ns / 1ps

    module tb_ALU_System_Gated();

        // Inputs
        reg [3:0] A;
        reg [3:0] B;
        reg [2:0] opcode;
        reg clk;
        reg reset_n;

        // Outputs
        wire [3:0] d_out;
        wire carry_out;
        wire clk_out;
        wire gated_clk;
        wire ce_sync;

        // Instantiate the Unit Under Test (UUT)
        ALU_System_Gated uut (
            .A(A),
            .B(B),
            .opcode(opcode),
            .clk(clk),
            .reset_n(reset_n),
            .d_out(d_out),
            .carry_out(carry_out),
            .clk_out(clk_out),
```

```
    .gated_clk(gated_clk),
    .ce_sync(ce_sync)
);

// Clock generation: 10ns period -> 100MHz
initial clk = 0;
always #5 clk = ~clk;

// Test stimulus
initial begin
    // Initialize Inputs
    A = 4'b0000;
    B = 4'b0000;
    opcode = 3'b000;
    reset_n = 0;

    // Apply reset
    #20;
    reset_n = 1;

    // Test ADD
    #20;
    A = 4'b0101; // 5
    B = 4'b0011; // 3
    opcode = 3'b000; // ADD

    // Test SUB
    #100;
    opcode = 3'b001;

    // Test AND
    #100;
    opcode = 3'b010;

    // Test OR
    #100;
    opcode = 3'b011;

    // Test XOR
```

```
        #100;
        opcode = 3'b100;

        // Test NOT
        #100;
        opcode = 3'b101;

        // Test SHL
        #100;
        opcode = 3'b110;

        // Test SHR
        #100;
        opcode = 3'b111;

        // Wait and finish
        #200;
        $finish;
    end

    // Waveform dumping (for debugging)
    initial begin
        $dumpfile("alu_waveform.vcd"); // Specify the name of the VCD file
        $dumpvars(0, tb_ALU_System_Gated); // Dump all signals in the top-level
testbench
    end

endmodule
```

- Constraints –

```
set_property PACKAGE_PIN E3 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000}
[get_ports clk]

set_property PACKAGE_PIN D9 [get_ports reset_n]
set_property IOSTANDARD LVCMOS33 [get_ports reset_n]
```

```
set_property PACKAGE_PIN J15 [get_ports {A[0]}]
set_property PACKAGE_PIN L16 [get_ports {A[1]}]
set_property PACKAGE_PIN M13 [get_ports {A[2]}]
set_property PACKAGE_PIN R15 [get_ports {A[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[*]}]

set_property PACKAGE_PIN R17 [get_ports {B[0]}]
set_property PACKAGE_PIN T18 [get_ports {B[1]}]
set_property PACKAGE_PIN U18 [get_ports {B[2]}]
set_property PACKAGE_PIN R13 [get_ports {B[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[*]}]

set_property PACKAGE_PIN T8 [get_ports {opcode[0]}]
set_property PACKAGE_PIN U8 [get_ports {opcode[1]}]
set_property PACKAGE_PIN R16 [get_ports {opcode[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {opcode[*]}]

set_property PACKAGE_PIN H17 [get_ports {d_out[0]}]
set_property PACKAGE_PIN K15 [get_ports {d_out[1]}]
set_property PACKAGE_PIN J13 [get_ports {d_out[2]}]
set_property PACKAGE_PIN N14 [get_ports {d_out[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {d_out[*]}]

set_property PACKAGE_PIN R18 [get_ports carry_out]
set_property IOSTANDARD LVCMOS33 [get_ports carry_out]

set_property PACKAGE_PIN U17 [get_ports clk_out]
set_property IOSTANDARD LVCMOS33 [get_ports clk_out]

set_property PACKAGE_PIN V17 [get_ports gated_clk]
set_property IOSTANDARD LVCMOS33 [get_ports gated_clk]

set_property PACKAGE_PIN V16 [get_ports ce_sync]
set_property IOSTANDARD LVCMOS33 [get_ports ce_sync]

set_operating_conditions -process maximum
```