**EJERCICIO 4**

Para el siguiente circuito determinar el setup slack y el hold slack. Asumir que el período de reloj es 10ns y que para todos los DFFs se cumple:

tsumax = 1,2ns

tsumin = 0,8ns

thmax = 0,7ns

thmin = 0,6ns

tclk,Qmax = 0,3ns

tclk,Qmin = 0,2ns

Considerar el multiplexor sin retardo.

1. Método BC-WC
2. B- Método OCV con CPPR

**Primer Flip-Flop D (FF1):**

- Entrada de reloj: Está conectado a la salida de la nube 1 (retardo Max: 2.3ns, Min: 1.2ns). La entrada de la nube 1 está conectada a una señal de reloj CLK

- Salida Q: La salida de FF1 se conecta como habilitación/selección del multiplexor.

- Entrada D: no está conectada

**Segundo Flip-Flop D (FF2):**

- Entrada de reloj: Está conectado a la salida de la nube 2 (Max: 1.8ns, Min: 1.5ns). La nube 2 está conectada a la salida de la nube 0 (Max: 0.3ns, Min: 0.1ns). La entrada de la nube 0 está conectada a la señal de reloj CLK.

-Salida Q: La salida de FF2 se conecta a la entrada de la nube 3 (Max: 3.7ns, Min: 2.5ns). La nube 3 está conectado al multiplexor (entrada 0)

- Entrada D: no está conectada

**Tercer Flip-Flop D (FF3):**

-Entrada de reloj: Está conectado a la salida de la nube 4 (Max: 1.9ns, Min: 1.1ns). La entrada a la nube 4 está conectada a la salida de la nube 0

-Salida Q: La salida de FF3 se conecta como entrada a la nube 5 (Max: 3.5ns, Min: 2.8ns). La salida de la nube 5 se conecta a la entrada del multiplexor (entrada 1)

-Entrada D: no está conectada

**Cuarto Flip-Flop D (FF4):**

-Entrada de reloj: recibe la salida de la nube 6 (Max: 1.6ns, Min: 0.9ns). La entrada de la nube 6 recibe la salida de la nube 0

-Salida Q: Es la salida final del circuito.

**Multiplexor:**

-Salida: La salida del multiplexor se conecta a la entrada D del FF4.

Para determinar el \*\*setup slack\*\* y el \*\*hold slack\*\* en el circuito proporcionado, primero es necesario entender los conceptos y los tiempos involucrados:

- \*\*Setup Time (tSU)\*\*: Tiempo mínimo que la señal de datos debe ser estable antes del flanco de reloj para que el flip-flop pueda registrar correctamente el dato.

- \*\*Hold Time (tH)\*\*: Tiempo mínimo que la señal de datos debe ser estable después del flanco de reloj para que el flip-flop pueda registrar correctamente el dato.

- \*\*Clock-to-Q Time (tCLK,Q)\*\*: Tiempo que tarda la señal de reloj en propagarse desde la entrada del flip-flop hasta su salida Q.

- \*\*Periodo de Reloj (T)\*\*: Tiempo entre dos flancos de reloj consecutivos.

### Datos del Problema:

- T = 10 ns

- tSUmax = 1.2 ns

- tSUmin = 0.8 ns

- tHmax = 0.7 ns

- tHmin = 0.6 ns

- tCLK,Qmax = 0.3 ns

- tCLK,Qmin = 0.2 ns

### Rutas de Reloj:

- Nube 1: Rango de retardo [1.2 ns, 2.3 ns]

- Nube 2: Rango de retardo [1.5 ns, 1.8 ns]

- Nube 0: Rango de retardo [0.1 ns, 0.3 ns]

- Nube 4: Rango de retardo [1.1 ns, 1.9 ns]

- Nube 6: Rango de retardo [0.9 ns, 1.6 ns]

### Rutas de Datos:

- Nube 3: Rango de retardo [2.5 ns, 3.7 ns]

- Nube 5: Rango de retardo [2.8 ns, 3.5 ns]

### Método BC-WC (Best Case - Worst Case)

#### Setup Slack:

Para el setup slack, consideramos el peor caso del camino de datos y el mejor caso del camino de reloj.

1. \*\*FF1 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo máximo = 0.3 ns (Nube 0) + 1.6 ns (Nube 6) = 1.9 ns

- \*\*Ruta de Datos (FF1 a FF4):\*\*

- FF1 a Multiplexor: tCLK,Qmax (FF1) = 0.3 ns

- Multiplexor a FF4: 0 ns (asumimos sin retardo)

- Total: 0.3 ns

- \*\*Setup Slack:\*\*

\[

\text{Setup Slack} = T - \text{Ruta de Datos (Worst Case)} - tSUmax - \text{Ruta de Reloj (Best Case)}

\]

\[

\text{Setup Slack} = 10 \text{ ns} - 0.3 \text{ ns} - 1.2 \text{ ns} - 1.9 \text{ ns} = 6.6 \text{ ns}

\]

2. \*\*FF2 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo máximo = 0.3 ns (Nube 0) + 1.6 ns (Nube 6) = 1.9 ns

- \*\*Ruta de Datos (FF2 a FF4):\*\*

- FF2 a Nube 3: 3.7 ns

- Nube 3 a Multiplexor: 0 ns (asumimos sin retardo)

- Total: 3.7 ns

- \*\*Setup Slack:\*\*

\[

\text{Setup Slack} = T - \text{Ruta de Datos (Worst Case)} - tSUmax - \text{Ruta de Reloj (Best Case)}

\]

\[

\text{Setup Slack} = 10 \text{ ns} - 3.7 \text{ ns} - 1.2 \text{ ns} - 1.9 \text{ ns} = 3.2 \text{ ns}

\]

3. \*\*FF3 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo máximo = 0.3 ns (Nube 0) + 1.6 ns (Nube 6) = 1.9 ns

- \*\*Ruta de Datos (FF3 a FF4):\*\*

- FF3 a Nube 5: 3.5 ns

- Nube 5 a Multiplexor: 0 ns (asumimos sin retardo)

- Total: 3.5 ns

- \*\*Setup Slack:\*\*

\[

\text{Setup Slack} = T - \text{Ruta de Datos (Worst Case)} - tSUmax - \text{Ruta de Reloj (Best Case)}

\]

\[

\text{Setup Slack} = 10 \text{ ns} - 3.5 \text{ ns} - 1.2 \text{ ns} - 1.9 \text{ ns} = 3.4 \text{ ns}

\]

#### Hold Slack:

Para el hold slack, consideramos el mejor caso del camino de datos y el peor caso del camino de reloj.

1. \*\*FF1 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo mínimo = 0.1 ns (Nube 0) + 0.9 ns (Nube 6) = 1.0 ns

- \*\*Ruta de Datos (FF1 a FF4):\*\*

- FF1 a Multiplexor: tCLK,Qmin (FF1) = 0.2 ns

- Multiplexor a FF4: 0 ns (asumimos sin retardo)

- Total: 0.2 ns

- \*\*Hold Slack:\*\*

\[

\text{Hold Slack} = \text{Ruta de Datos (Best Case)} - \text{Ruta de Reloj (Worst Case)} - tHmin

\]

\[

\text{Hold Slack} = 0.2 \text{ ns} - 1.0 \text{ ns} - 0.6 \text{ ns} = -1.4 \text{ ns}

\]

2. \*\*FF2 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo mínimo = 0.1 ns (Nube 0) + 0.9 ns (Nube 6) = 1.0 ns

- \*\*Ruta de Datos (FF2 a FF4):\*\*

- FF2 a Nube 3: 2.5 ns

- Nube 3 a Multiplexor: 0 ns (asumimos sin retardo)

- Total: 2.5 ns

- \*\*Hold Slack:\*\*

\[

\text{Hold Slack} = \text{Ruta de Datos (Best Case)} - \text{Ruta de Reloj (Worst Case)} - tHmin

\]

\[

\text{Hold Slack} = 2.5 \text{ ns} - 1.0 \text{ ns} - 0.6 \text{ ns} = 0.9 \text{ ns}

\]

3. \*\*FF3 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo mínimo = 0.1 ns (Nube 0) + 0.9 ns (Nube 6) = 1.0 ns

- \*\*Ruta de Datos (FF3 a FF4):\*\*

- FF3 a Nube 5: 2.8 ns

- Nube 5 a Multiplexor: 0 ns (asumimos sin retardo)

- Total: 2.8 ns

- \*\*Hold Slack:\*\*

\[

\text{Hold Slack} = \text{Ruta de Datos (Best Case)} - \text{Ruta de Reloj (Worst Case)} - tHmin

\]

\[

\text{Hold Slack} = 2.8 \text{ ns} - 1.0 \text{ ns} - 0.6 \text{ ns} = 1.2 \text{ ns}

\]

### Método OCV con CPPR (On-Chip Variation with Common Path Pessimism Removal)

En el método OCV, se consideran las variaciones en el proceso, voltaje y temperatura (PVT) y se aplica un factor de corrección para reducir el pesimismo en las rutas comunes.

1. \*\*Setup Slack:\*\*

- \*\*FF1 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo máximo = 1.9 ns

- \*\*Ruta de Datos (FF1 a FF4):\*\* 0.3 ns

- \*\*Setup Slack:\*\*

\[

\text{Setup Slack} = T - \text{Ruta de Datos (Worst Case)} - tSUmax - \text{Ruta de Reloj (Best Case)} + \text{CPPR}

\]

\[

\text{Setup Slack} = 10 \text{ ns} - 0.3 \text{ ns} - 1.2 \text{ ns} - 1.9 \text{ ns} + 0.3 \text{ ns} = 6.9 \text{ ns}

\]

- \*\*FF2 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo máximo = 1.9 ns

- \*\*Ruta de Datos (FF2 a FF4):\*\* 3.7 ns

- \*\*Setup Slack:\*\*

\[

\text{Setup Slack} = 10 \text{ ns} - 3.7 \text{ ns} - 1.2 \text{ ns} - 1.9 \text{ ns} + 0.3 \text{ ns} = 3.5 \text{ ns}

\]

- \*\*FF3 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo máximo = 1.9 ns

- \*\*Ruta de Datos (FF3 a FF4):\*\* 3.5 ns

- \*\*Setup Slack:\*\*

\[

\text{Setup Slack} = 10 \text{ ns} - 3.5 \text{ ns} - 1.2 \text{ ns} - 1.9 \text{ ns} + 0.3 \text{ ns} = 3.7 \text{ ns}

\]

2. \*\*Hold Slack:\*\*

- \*\*FF1 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo mínimo = 1.0 ns

- \*\*Ruta de Datos (FF1 a FF4):\*\* 0.2 ns

- \*\*Hold Slack:\*\*

\[

\text{Hold Slack} = \text{Ruta de Datos (Best Case)} - \text{Ruta de Reloj (Worst Case)} - tHmin + \text{CPPR}

\]

\[

\text{Hold Slack} = 0.2 \text{ ns} - 1.0 \text{ ns} - 0.6 \text{ ns} + 0.3 \text{ ns} = -1.1 \text{ ns}

\]

- \*\*FF2 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo mínimo = 1.0 ns

- \*\*Ruta de Datos (FF2 a FF4):\*\* 2.5 ns

- \*\*Hold Slack:\*\*

\[

\text{Hold Slack} = 2.5 \text{ ns} - 1.0 \text{ ns} - 0.6 \text{ ns} + 0.3 \text{ ns} = 1.2 \text{ ns}

\]

- \*\*FF3 a FF4:\*\*

- \*\*Ruta de Reloj (FF4):\*\* Retardo mínimo = 1.0 ns

- \*\*Ruta de Datos (FF3 a FF4):\*\* 2.8 ns

- \*\*Hold Slack:\*\*

\[

\text{Hold Slack} = 2.8 \text{ ns} - 1.0 \text{ ns} - 0.6 \text{ ns} + 0.3 \text{ ns} = 1.5 \text{ ns}

\]

### Resumen:

- \*\*Setup Slack (BC-WC):\*\*

- FF1 a FF4: 6.6 ns

- FF2 a FF4: 3.2 ns

- FF3 a FF4: 3.4 ns

- \*\*Hold Slack (BC-WC):\*\*

- FF1 a FF4: -1.4 ns (violation)

- FF2 a FF4: 0.9 ns

- FF3 a FF4: 1.2 ns

- \*\*Setup Slack (OCV with CPPR):\*\*

- FF1 a FF4: 6.9 ns

- FF2 a FF4: 3.5 ns

- FF3 a FF4: 3.7 ns

- \*\*Hold Slack (OCV with CPPR):\*\*

- FF1 a FF4: -1.1 ns (violation)

- FF2 a FF4: 1.2 ns

- FF3 a FF4: 1.5 ns

### Conclusiones:

- El circuito tiene un problema de hold time en la ruta FF1 a FF4, tanto en BC-WC como en OCV con CPPR.

- El setup slack es suficiente en todas las rutas.

**EJERCICIO 3**

Si para el circuito del ejercicio 1 (N=8), se deja fija la entrada enable a ’1’ constante, cual es el factor de actividad de la entrada D de cada DFF? Si cada compuerta del circuito (INV,NOR) posee los siguientes consumos de potencia: INV: 1 unidad

NOR: 2 unidades

DFF: 8 unidades.

Determine la potencia de consumo dinámico en unidades equivalentes de potencia.

(Ejercicio 1): Implementar en Verilog un contador genérico de N bits con señal de enable. Cuando dicha entrada es ’0’, el contador detiene la cuenta. Sintetizar “manualmente” dicho contador suponiendo que solo se posee DFFs, inversores, compuertas NOR y ICGs para el caso de N=8.

Para resolver el ejercicio 3, primero necesitamos entender el funcionamiento del contador genérico de N bits con señal de enable, y luego calcular el factor de actividad y el consumo de potencia dinámica.

### Circuito del Contador

El contador genérico de 8 bits con señal de enable (E) se puede implementar utilizando DFFs, inversores y compuertas NOR. La señal de enable (E) controla si el contador avanza o no. Cuando E = 1, el contador avanza; cuando E = 0, el contador se detiene.

### Sintetización Manual del Contador

Para N = 8, el contador se compone de 8 DFFs, cada uno con una entrada D que depende de las salidas de los DFFs anteriores y la señal de enable. La lógica para cada DFF se puede implementar utilizando compuertas NOR e inversores.

#### Síntesis de la Lógica

1. \*\*DFF 0 (Q0):\*\*

- D0 = E

- No requiere lógica adicional.

2. \*\*DFF 1 (Q1):\*\*

- D1 = E NOR Q0

- Requiere 1 compuerta NOR y 1 inversor (para E).

3. \*\*DFF 2 (Q2):\*\*

- D2 = E NOR (Q0 NOR Q1)

- Requiere 2 compuertas NOR y 1 inversor (para E).

4. \*\*DFF 3 (Q3):\*\*

- D3 = E NOR (Q0 NOR Q1 NOR Q2)

- Requiere 3 compuertas NOR y 1 inversor (para E).

5. \*\*DFF 4 (Q4):\*\*

- D4 = E NOR (Q0 NOR Q1 NOR Q2 NOR Q3)

- Requiere 4 compuertas NOR y 1 inversor (para E).

6. \*\*DFF 5 (Q5):\*\*

- D5 = E NOR (Q0 NOR Q1 NOR Q2 NOR Q3 NOR Q4)

- Requiere 5 compuertas NOR y 1 inversor (para E).

7. \*\*DFF 6 (Q6):\*\*

- D6 = E NOR (Q0 NOR Q1 NOR Q2 NOR Q3 NOR Q4 NOR Q5)

- Requiere 6 compuertas NOR y 1 inversor (para E).

8. \*\*DFF 7 (Q7):\*\*

- D7 = E NOR (Q0 NOR Q1 NOR Q2 NOR Q3 NOR Q4 NOR Q5 NOR Q6)

- Requiere 7 compuertas NOR y 1 inversor (para E).

### Factor de Actividad

El factor de actividad (FA) de una señal es la proporción de tiempo que la señal está cambiando. Para un contador binario de N bits, la entrada D de cada DFF cambia con una frecuencia que es la mitad de la frecuencia de la señal de reloj (CLK) para el primer DFF, y cada DFF subsiguiente cambia la mitad de la frecuencia del DFF anterior.

- \*\*D0:\*\* Cambia en cada ciclo de reloj.

- \*\*D1:\*\* Cambia en la mitad de los ciclos de reloj.

- \*\*D2:\*\* Cambia en la cuarta parte de los ciclos de reloj.

- \*\*D3:\*\* Cambia en la octava parte de los ciclos de reloj.

- \*\*D4:\*\* Cambia en la décima sexta parte de los ciclos de reloj.

- \*\*D5:\*\* Cambia en la treinta y segunda parte de los ciclos de reloj.

- \*\*D6:\*\* Cambia en la sesenta y cuarta parte de los ciclos de reloj.

- \*\*D7:\*\* Cambia en la ciento veintiochoava parte de los ciclos de reloj.

El factor de actividad de la entrada D de cada DFF es:

- \*\*FA(D0) = 1\*\*

- \*\*FA(D1) = 0.5\*\*

- \*\*FA(D2) = 0.25\*\*

- \*\*FA(D3) = 0.125\*\*

- \*\*FA(D4) = 0.0625\*\*

- \*\*FA(D5) = 0.03125\*\*

- \*\*FA(D6) = 0.015625\*\*

- \*\*FA(D7) = 0.0078125\*\*

### Consumo de Potencia Dinámica

El consumo de potencia dinámica de un circuito se calcula multiplicando el consumo de cada componente por su factor de actividad.

- \*\*DFF:\*\* 8 unidades

- \*\*INV:\*\* 1 unidad

- \*\*NOR:\*\* 2 unidades

#### Consumo de Potencia de Cada Componente

1. \*\*DFFs:\*\*

- Consumo total = 8 \* (1 + 0.5 + 0.25 + 0.125 + 0.0625 + 0.03125 + 0.015625 + 0.0078125) = 8 \* 1.984375 = 15.875 unidades

2. \*\*Inversores:\*\*

- Hay 8 inversores (uno para cada DFF).

- Consumo total = 8 \* 1 \* 1 = 8 unidades

3. \*\*Compuertas NOR:\*\*

- D1: 1 NOR

- D2: 2 NOR

- D3: 3 NOR

- D4: 4 NOR

- D5: 5 NOR

- D6: 6 NOR

- D7: 7 NOR

- Consumo total = 2 \* (1 + 2 + 3 + 4 + 5 + 6 + 7) = 2 \* 28 = 56 unidades

#### Consumo de Potencia Total

- \*\*Total:\*\* 15.875 (DFFs) + 8 (Inversores) + 56 (NORs) = 80.875 unidades

### Resumen

- \*\*Factor de Actividad de la Entrada D de Cada DFF:\*\*

- D0: 1

- D1: 0.5

- D2: 0.25

- D3: 0.125

- D4: 0.0625

- D5: 0.03125

- D6: 0.015625

- D7: 0.0078125

- \*\*Consumo de Potencia Dinámica Total:\*\* 80.875 unidades

Este es el consumo de potencia dinámico del circuito del contador de 8 bits con señal de enable fija a 1.

**EJERCICIO 1**

Implementar en Verilog un contador genérico de N bits con señal de enable. Cuando dicha entrada es ’0’, el contador detiene la cuenta.

Sintetizar “manualmente” dicho contador suponiendo que solo se posee DFFs, inversores, compuertas NOR y ICGs para el caso de N=8.

Para implementar un contador genérico de N bits con señal de enable en Verilog, primero definiremos el módulo Verilog y luego sintetizaremos manualmente el diseño utilizando DFFs, inversores, compuertas NOR y ICGs (Integrated Clock Gating Cells) para el caso de \( N = 8 \).

### Paso 1: Definición del Módulo Verilog

El módulo Verilog para un contador de 8 bits con señal de enable puede ser definido de la siguiente manera:

```verilog

module counter\_8bit\_enable (

input wire clk,

input wire reset,

input wire enable,

output reg [7:0] count

);

always @(posedge clk or posedge reset) begin

if (reset)

count <= 8'b0;

else if (enable)

count <= count + 1;

end

endmodule

```

### Paso 2: Sintetización Manual

Para sintetizar manualmente el contador, necesitamos implementar los siguientes componentes:

1. \*\*DFFs (Flip-Flops D)\*\*: Para almacenar el valor del contador.

2. \*\*Inversores\*\*: Para invertir señales si es necesario.

3. \*\*Compuertas NOR\*\*: Para implementar la lógica de incremento.

4. \*\*ICGs (Integrated Clock Gating Cells)\*\*: Para controlar el enable.

#### Implementación de los DFFs

Cada bit del contador se almacena en un DFF. Para un contador de 8 bits, necesitamos 8 DFFs.

```verilog

module dff (

input wire clk,

input wire reset,

input wire d,

output reg q

);

always @(posedge clk or posedge reset) begin

if (reset)

q <= 1'b0;

else

q <= d;

end

endmodule

```

#### Implementación de la Lógica de Incremento

La lógica de incremento para un contador de 8 bits se puede implementar utilizando compuertas NOR y inversores. La lógica de incremento es básicamente un sumador de 1, que se puede implementar con compuertas NOR y inversores.

#### Implementación de los ICGs

Los ICGs se utilizan para controlar el enable. Un ICG puede ser implementado con una compuerta AND que combina la señal de enable con el clock.

```verilog

module icg (

input wire clk,

input wire enable,

output wire gated\_clk

);

assign gated\_clk = clk & enable;

endmodule

```

### Integración del Diseño

Ahora, integramos todos los componentes para formar el contador de 8 bits con enable.

```verilog

module counter\_8bit\_enable\_synth (

input wire clk,

input wire reset,

input wire enable,

output wire [7:0] count

);

wire gated\_clk;

icg icg\_inst (

.clk(clk),

.enable(enable),

.gated\_clk(gated\_clk)

);

wire [7:0] next\_count;

wire [7:0] count\_reg;

// Generate the next count value

assign next\_count[0] = count\_reg[7] & count\_reg[6] & count\_reg[5] & count\_reg[4] & count\_reg[3] & count\_reg[2] & count\_reg[1] & count\_reg[0] ? 1'b1 : ~count\_reg[0];

assign next\_count[1] = count\_reg[7] & count\_reg[6] & count\_reg[5] & count\_reg[4] & count\_reg[3] & count\_reg[2] & count\_reg[1] ? 1'b1 : ~count\_reg[1];

assign next\_count[2] = count\_reg[7] & count\_reg[6] & count\_reg[5] & count\_reg[4] & count\_reg[3] & count\_reg[2] ? 1'b1 : ~count\_reg[2];

assign next\_count[3] = count\_reg[7] & count\_reg[6] & count\_reg[5] & count\_reg[4] & count\_reg[3] ? 1'b1 : ~count\_reg[3];

assign next\_count[4] = count\_reg[7] & count\_reg[6] & count\_reg[5] & count\_reg[4] ? 1'b1 : ~count\_reg[4];

assign next\_count[5] = count\_reg[7] & count\_reg[6] & count\_reg[5] ? 1'b1 : ~count\_reg[5];

assign next\_count[6] = count\_reg[7] & count\_reg[6] ? 1'b1 : ~count\_reg[6];

assign next\_count[7] = count\_reg[7] ? 1'b1 : ~count\_reg[7];

// Instantiate DFFs for each bit of the counter

dff dff\_inst0 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[0]),

.q(count\_reg[0])

);

dff dff\_inst1 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[1]),

.q(count\_reg[1])

);

dff dff\_inst2 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[2]),

.q(count\_reg[2])

);

dff dff\_inst3 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[3]),

.q(count\_reg[3])

);

dff dff\_inst4 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[4]),

.q(count\_reg[4])

);

dff dff\_inst5 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[5]),

.q(count\_reg[5])

);

dff dff\_inst6 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[6]),

.q(count\_reg[6])

);

dff dff\_inst7 (

.clk(gated\_clk),

.reset(reset),

.d(next\_count[7]),

.q(count\_reg[7])

);

// Assign the register values to the output

assign count = count\_reg;

endmodule

```

### Explicación

1. \*\*ICG\*\*: El ICG combina la señal de enable con el clock para generar un clock gated que se usa para controlar el enable.

2. \*\*DFFs\*\*: Cada bit del contador se almacena en un DFF que se actualiza en el flanco de subida del clock gated.

3. \*\*Lógica de Incremento\*\*: La lógica de incremento se implementa con compuertas NOR y inversores para calcular el siguiente valor del contador.

Este diseño sintetizado manualmente utiliza solo DFFs, inversores, compuertas NOR y ICGs, como se requirió.

**EJERCICIO 2**

Para el siguiente circuito determine las SDC constraints necesarias para realizar el STA del path A-B. Considerar que el reloj es de 100MHz y el contador “divide” al reloj por 8. El circuito es el siguiente: Una señal de CLK se conecta a la entrada de reloj de un flip flop y a la entrada de un contador. La entrada D del FF no se conecta a nada. La salida Q del FF se conecta a una nube de lógica combinacional, y la salida de esta nube se conecta a la entrada (1) de un multiplexor. La salida del contador se conecta se conecta a la entrada de enable/selección del MUX. La salida del MUX se conecta a otro FF en su entrada D. La señal CLK se conecta también a la entrada de reloj de este FF. La salida Q de este FF es la salida definitiva que a su vez se conecta con la entrada (0) del MUX. La salida Q del primer FF es también llamada A y la entrada D del último FF es llamada B

Para determinar las SDC (Synopsys Design Constraints) necesarias para realizar el STA (Static Timing Analysis) del path A-B en el circuito descrito, siguen estos pasos:

### Paso 1: Identificar los Elementos del Circuito

1. \*\*CLK\*\*: Señal de reloj de 100 MHz.

2. \*\*Contador\*\*: Divide el reloj por 8.

3. \*\*FF1\*\*: Primer flip-flop con entrada D no conectada.

4. \*\*Lógica Combinacional\*\*: Nube de lógica combinacional.

5. \*\*MUX\*\*: Multiplexor con dos entradas (0 y 1).

6. \*\*FF2\*\*: Segundo flip-flop.

7. \*\*A\*\*: Salida Q del primer FF.

8. \*\*B\*\*: Entrada D del segundo FF.

### Paso 2: Determinar las Constraints

#### 1. Definer el reloj

El reloj es de 100 MHz, lo que implica un período de 10 ns.

```tcl

create\_clock -name clk -period 10.000 [get\_ports clk]

```

#### 2. Definir la división del reloj por el contador

El contador divide el reloj por 8, lo que implica que el período del reloj dividido es 80 ns.

```tcl

create\_generated\_clock -name clk\_div -source [get\_ports clk] -divide\_by 8 [get\_pins contador/output]

```

#### 3. Definir las entradas y salidas del circuito

Las entradas y salidas del circuito son `A` y `B`.

```tcl

set\_input\_delay -clock clk 0 [get\_ports A]

set\_output\_delay -clock clk 0 [get\_ports B]

```

#### 4. Definir los paths críticos

El path crítico es de `A` a `B`.

```tcl

set\_false\_path -from [get\_ports A] -to [get\_pins FF2/D]

set\_false\_path -from [get\_pins contador/Output] -to [get\_pins MUX/SEL]

```

### Paso 3: Escribir el archivo S DC

Aquí está el archivo SDC completo:

```tcl

# Define the clock

create\_clock -name clk -period 10.000 [get\_ports clk]

# Define the generated clock from the counter

create\_generatedclock -name clk\_div -source [get\_ports clk] -divide\_by 8 [get\_pins contador/Output]

# Define the input and output delays

set\_input\_delay -clock clk 0 [get\_ports A]

set\_output\_delay -clock clk 0 [get\_ports B]

# Define the critical paths

set\_false\_path -from [get\_ports A] -to [get\_pins FF2/D]

set\_false\_path -from [get\_pins contador/Output] -to [get\_pins MUX/SEL]

```

### Explicación

1. \*\*create\_clock\*\*: Define el reloj principal de 100 MHz con un período de 10 ns.

2. \*\*create\_generated\_clock\*\*: Define el reloj generado por el contador, que divide el reloj principal por 8, resultando en un período de 80 ns.

3. \*\*set\_input\_delay\*\*: Define el retardo de entrada para la señal `A`.

4. \*\*set\_output\_delay\*\*: Define el retardo de salida para la señal `B`.

5. \*\*set\_false\_path\*\*: Define los paths que no deben ser considerados en el análisis de tiempo, como el path de `A` a `D` del segundo FF y el path de la salida del contador a la selección del MUX.

Este archivo SDC asegura que el STA se realiza correctamente considerando las características del circuito y las divisiones de reloj.