Examen de Diseño de Circuitos Integrados Digitales – Maestría en Ciencias de la Ingeniería

Ing. Mariano Morel

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 se define el módulo Verilog y luego se sintetiza el diseño utilizando DFFs, inversores, compuertas NOR y ICGs para N = 8.

MODULO CONTADOR GENERICO

```
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
```

A continuación y a modo de complemento, se describen los módulos de un contador de 8bits, implementados con DFFs e ICG. Luego se analiza la sintesis.

MODULO INTEGRADO

```
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;
```

// Contador: Estas asignaciones implementan la lógica de incremento para un contador de 8 bits. Cada bit del contador se actualiza basado en el valor actual de todos los bits superiores. Si todos los bits superiores son 1, el bit actual se setea a 1. Si no, se invierte el valor actual del bit. Esto asegura que el contador incremente correctamente en 1 en cada ciclo de reloj.

```
    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];

    // Instancia de DFFs para cada bit del contador
    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])
);
```

```
// Salida
    assign count = count_reg;
endmodule
```

MODULO DFF

```
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
```

MODULO ICG

```
module icg (
    input wire clk,
    input wire enable,
    output wire gated_clk
);
    assign gated_clk = clk & enable;
endmodule
```

```
//El ICG combina la señal de enable con el clock para generar un clock gated que se usa para controlar el enable. Cada bit del contador se almacena en un DFF que se actualiza en el flanco de subida del clock gated.
```

Para la **síntesis manual**, cada ICG tiene dos entradas: CLK y EN. La salida del ICG se conecta al pin de clock de cada DFF. Para el resto del contador, la lógica es (no lo dibujo pero se puede seguir fácilmente):

Q0 -> Inversor0 -> Q0'
Q0 y Q1' -> NOR0 -> D1
Q1 -> Inversor1 -> Q1'
Q1 y Q2' -> NOR1 -> D2
Q2 -> Inversor2 -> Q2'
Q2 y Q3' -> NOR2 -> D3
Q3 -> Inversor3 -> Q3'
Q3 y Q4' -> NOR3 -> D4

Q4 -> Inversor4 -> Q4'

Q4 y Q5' -> NOR4 -> D5

Q5 -> Inversor5 -> Q5'

Q5 y Q6' -> NOR5 -> D6

Q6 -> Inversor6 -> Q6'

Q6 y Q7' -> NOR6 -> D7

Q7 -> Inversor7 -> Q7'

En definitiva se usan 8 DFFs, un ICG común a todos ellos, 7 compuertas NOR y 8 INVERSORES.

Ejemplo de Cuenta: Inicialmente supongamos que todas las salidas Q están en '0' (estado inicial del contador).

- Primer Flanco de Reloj: Q0 cambia a '1' porque D0 es Q0' (que es '1' cuando Q0 es '0'). D1 es Q0 NOR Q1' = '0' NOR '1' = '0', por lo que Q1 sigue siendo '0'. El estado del contador es 00000001.
- Segundo Flanco de Reloj: Q0 cambia a '0' porque D0 es Q0' (que es '0' cuando Q0 es '1'). D1 es Q0 NOR Q1' = '0' NOR '1' = '1', por lo que Q1 cambia a '1'. El estado del contador es 00000010.
- Tercer Flanco de Reloj: Q0 cambia a '1' porque D0 es Q0' (que es '1' cuando Q0 es '0'). D1 es Q0 NOR Q1' = '1' NOR '0' = '0', por lo que Q1 sigue siendo '1'. D2 es Q1 NOR Q2' = '1' NOR '1' = '0', por lo que Q2 sigue siendo '0'. El estado del contador es 00000011.
- Cuarto Flanco de Reloj: Q0 cambia a '0' porque D0 es Q0' (que es '0' cuando Q0 es '1'). D1 es Q0 NOR Q1' = '0' NOR '0' = '1', por lo que Q1 cambia a '0'. D2 es Q1 NOR Q2' = '0' NOR '1' = '1', por lo que Q2 cambia a '1'. El estado del contador es 00000100.

Este diseño permite que el contador cuente de 0 a 255 (para 8 bits) y se detenga cuando EN es '0'.

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.

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

1_Identificar los elementos del circuito (esquema)

2_Determinar las Constraints (archivo SDC):

//Definir el reloj: El reloj es de 100 MHz (T=10 ns).

create_clock -name clk -period 10.000 [get_ports clk]

//Definir la división del reloj por el contador: El contador divide el reloj por 8 (T=80 ns).

create_generated_clock -name clk_div -source [get_ports clk] -divide_by 8 [get_pins contador/output]

//Definir las entradas y salidas del circuito: A y B.

set_input_delay -clock clk 0 [get_ports A]

set_output_delay -clock clk 0 [get_ports B]

//Definir los paths críticos: El path crítico es de A a B.

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]

“create_clock” define el reloj principal de 100 MHz, “create_generated_clock” define el reloj generado por el contador, que divide el reloj principal por 8, “set_input_delay” define el retardo de entrada para la señal A, “set_output_delay”, define el retardo de salida para la señal B, “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.

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.

El factor de actividad 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.

FA(D0) = 1

FA(D1) = 0.5

…

FA(D7) = 0.0078125

El Consumo de Potencia Dinámica se calcula multiplicando el consumo de cada componente por su factor de actividad.

Se tiene -> DFF: 8 unidades, INV: 1 unidad y NOR: 2 unidades.

DFFs:

Consumo total DFFs = 8 x (1 + 0.5 + 0.25 + 0.125 + 0.0625 + 0.03125 + 0.015625 + 0.0078125)

= 8 x 1.984375 = 15.875 un.

Inversores:

Consumo total INVs= 1 x 8 = 8 un

Compuertas NOR:

Consumo total = 2 x 7 = 14 un.

Consumo de Potencia Total = Total: 15.875 (DFFs) + 8 (INVs) + 14 (NORs) = 37,875 un.

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.

A- Método BC-WC

B- B- Método OCV con CPPR

Rutas de Reloj:

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

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

Nube 2: Rango de retardo [1.5 ns, 1.8 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: se considera 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

$$\text{Setup Slack} = T - \text{Ruta de Datos (Worst Case)} - t_{SU,\max} - \text{Ruta de Reloj (Best Case)}$$

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

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

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

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

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

Hold Slack: se considera 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

$$\text{Hold Slack} = \text{Ruta de Datos (Best Case)} - \text{Ruta de Reloj (Worst Case)} - t_{H,\min}$$

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

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

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

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

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

Método OCV con CPPR (On-Chip Variation with Common Path Pessimism Removal): 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. Si bien se remueve el pesimismo, lo que se puede agregar un factor de corrección. Supongo un valor de CPPR de 0,3ns equivalente a tclk,Qmax.

1. Setup Slack:

- FF1 a FF4:

- Ruta de Reloj (FF4): Retardo máximo = 1.9 ns
- Ruta de Datos (FF1 a FF4): 0.3 ns

$$\text{Setup Slack} = T - \text{Ruta de Datos (Worst Case)} - t_{SU,\text{max}} - \text{Ruta de Reloj (Best Case)} + \text{CPPR}$$

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

- FF2 a FF4:

- Ruta de Reloj (FF4): Retardo máximo = 1.9 ns
- Ruta de Datos (FF2 a FF4): 3.7 ns

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

- FF3 a FF4:

- Ruta de Reloj (FF4): Retardo máximo = 1.9 ns
- Ruta de Datos (FF3 a FF4): 3.5 ns

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

2. Hold Slack:

- FF1 a FF4:

- Ruta de Reloj (FF4): Retardo mínimo = 1.0 ns
- Ruta de Datos (FF1 a FF4): 0.2 ns

$$\text{Hold Slack} = \text{Ruta de Datos (Best Case)} - \text{Ruta de Reloj (Worst Case)} - t_{H,\text{min}} + \text{CPPR}$$

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

- FF2 a FF4:

- Ruta de Reloj (FF4): Retardo mínimo = 1.0 ns
- Ruta de Datos (FF2 a FF4): 2.5 ns

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

- FF3 a FF4:

- Ruta de Reloj (FF4): Retardo mínimo = 1.0 ns
- Ruta de Datos (FF3 a FF4): 2.8 ns

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

Resultados:

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.