UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Máster Universitario en Ingeniería de Telecomunicación

Ejercicio 1

Tutorial de Herramientas Cuestiones prácticas co-diseño

CHS

Autor:
Andrés Ruz Nieto

VALENCIA, 2021

TELECOM UPV VLC

1. Tutorial de Refresco de uso de las herramientas

1. **¿Qué modificaciones hay que realizar en el código para poder realizar la simulación “Gate-Level”? ¿Por qué?**

   Deberemos comentar ”*.fin_cuenta(modulo)*” del testbench

```
contador #(.fin_cuenta(modulo)) i1 (
                                    .iCLOCK(CLK),
                                    .iRESET_n(RESET_A),
                                    .iENABLE(ENABLE),
                                    .iUP_DOWN(UP_DOWN),
                                    .oCOUNT(COUNT),
                                    .oTC(TC));
```

   Esto es debido a que para ejecutar una simulación ”Gate-Level” previamente el programa realiza el ”Place & Route” sobre la placa, una vez hecho este proceso no podemos asignar o cambiar este valor ya que el nivel de las puertas lógicas es fijo.

2. **¿Qué diferencias hay entre la simulación RTL y la “Gate-Level”?**

   - **RTL**: La simulación RTL solo se verifica la funcionalidad del código, en caso de que la sintaxis sea correcta la simulación será existosa. En este tipo de simulación no se tienen en cuenta los retardos temporales.
   - **Gate-Level**: La simulación Gate-Level tiene en cuenta los retardos temporales y para la realización de esta simulación previamente se hace el ”Place & Route” sobre la placa, por lo que este tipo de simulación es más restrictiva.

3. **¿Cómo se podría modificar la velocidad con que varia la cuenta?**

   Modificando la frecuencia del reloj

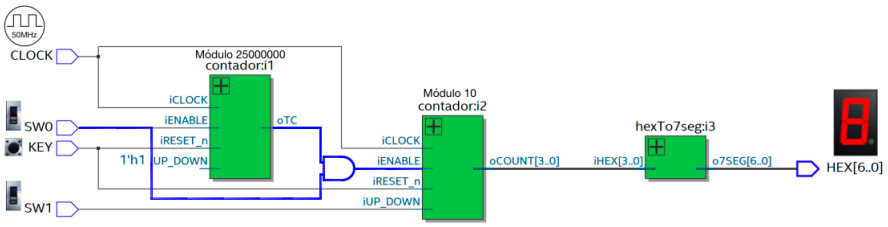
4. **¿Qué modificaciones del diseño se tendrían que hacer para tener un contador de hexadecimal, en lugar de decimal?**

   Se deberá cambiar el valor que se le pasa a ”*fin_cuenta*” de 10 a 16

```
contador #(.fin_cuenta(10)) i2 (
                                    .iCLOCK(CLOCK),
                                    .iRESET_n(KEY),
                                    .iENABLE(TC),
                                    .iUP_DOWN(SW1),
                                    .oCOUNT(COUNT),
                                    .oTC());
```

5. **¿Qué ocurre si se detiene el contador:i1 en el instante que su salida oTC está activa? ¿Cómo solucionar este pequeño fallo?**

   Si al detener el *contador:i1*, *oTC* se queda activada, la señal de habilitación (*iENABLE*) también estará activa por lo que el *contador:i2* no dejará de contar en ningún momento, además contará con una frecuencia de 50MHz (la del reloj), para solucionar este problema se puede añadir una puerta AND cuyas entradas sean *oTC* y *iENABLE* del contador:i1 y su salida será el *iENABLE* del contador:i2.



2. Tutorial de Herramientas

1. **¿De que depende la velocidad de intermitencia del led? ¿Cómo se puede modificar?**

   La velocidad de intermitencia del led depende del valor máximo de "*delay*"

```
IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, count & 0x01);
delay = 0;
while(delay<2000000)
{
      delay++;
}
count++;
```

   Reduciendo este valor el led parpadeará más rápido.

2. **¿De que manera se puede conseguir que la intermitencia del led se ajuste a un parámetro temporal de N segundos exactos e invariables?**

   A través de software no se puede asegurar un tiempo exacto, para ello se podría emplear un timer en hardware.

3. **¿Cómo modificar el programa para reflejar en los leds una cuenta binaria de 4 bits? ¿y si se quieren 8 bits de cuenta?**

   En el segundo parámetro de la función "*IOWR_ALTERA_AVALON_PIO_DATA*" se está pasando el valor del contador pero "enmascarado". Solo se queda con el bit menos significativo, ya que se está realizando un AND lógico con el contador. Cambiando el 0x01 por 0x0F se obtendrán 4 bits, y si se cambia por 0xFF se obtendrán 8 bits.

```
IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, count & 0x01);
delay = 0;
while(delay<2000000)
{
      delay++;
}
count++;
```

4. **¿Qué ocurre en la placa después de la realización del punto 58? ¿Por qué?**

   Se puede ver sobre la FPGA que el contador está funcionando pero el led no se enciende, esto es debido a que para poder reconfigurar la FPGA desde Quartus, hay que detener el programa en ECLIPSE ya que este tiene el control de JTAG.

5. **Describe que diseño se tiene dentro de la FPGA después del punto 64.**

   La FPGA contiene un NIOS II que controla el encendido y apagado de los LEDS, también contiene, de forma independiente, el contador que se ha realizado al principio de la sesión de prácticas y se encarga de mostrar los números de la cuenta en el *display* de siete segmentos. Como se puede ver ambos diseños están conectados al mismo reloj de 50MHz.