Escuela de Ingeniería Electrónica
Maestría en electrónica
Curso: Prototipado en FPGAs

Tercer Proyecto

Valor 15%

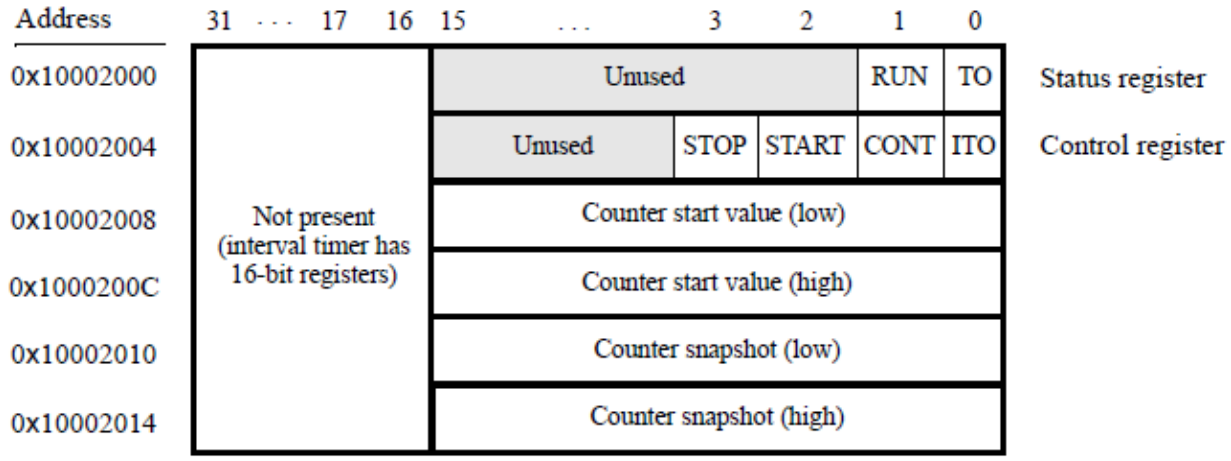
Fecha de entrega: 22 de Julio, 2017

Introducción

Los circuitos tipo Timer/Counter son muy utilizados en sistemas embebidos. En este ejercicio se utilizará el módulo "interval timer" que se encuentra en el repositorio de Altera. Se ilustrará como este módulo se puede usar para estimar rendimiento de una aplicación, este en términos del número total de ciclos de reloj usados para ejecutar el programa.

El timer se puede configurar en un valor específico y su contador interno se decrementará en cada ciclo de reloj. Cuando el contador llega a cero, un evento "timeout" se dispara. Con este evento se dispara una solicitud de interrupción y el contador puede volver al valor inicial.

La figura muestra un ejemplo de un timer con una base-address igual a 0x10002000



El contador contiene un registro de control en el offset 0x04. El valor inicial se configura en los dos registros que están en offset 0x08 y 0x0C, estos valores son 16 bits de ancho. En cualquier momento se puede tomar un *snapshot* del contador al escribir en el offset 0x10, una vez hecho esto se pueden leer dichos registros.

Registro	Dirección	Descripción
TO	R/W	Time out bit, Se setea en 1 cuando el contador llega a cero. Debe ser borrado al escribir sobre él un 0. De esta manera se limpia la bandera de interrupción
RUN	R	Muestra 1 cuando el contador está corriendo y 0 si no.
ITO	W	Habilita las interrupciones cuando es igual a 1
CONT	W	Si es 1 el contador no se detiene cuando llega a cero. Si es 0 el contador se detiene cuando llega a cero.
START	W	Cuando se escribe 1 dispara el contador
STOP	W	Cuando se escribe 1 detiene el contador

Adicionalmente se estudiará la puesta en marcha de un protocolo estándar de comunicación serial.

Descripción

Parte 1

1. Repase el tutorial introductorio a QSYS
2. Revise la documentación de los módulos “Interval Timer Core” y “SPI Core”(capítulos 9 y 28)

   https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_embedded_ip.pdf

3. Monte un proyecto nuevo en quartus, use como targe la tarjeta “DE0-nano-Soc”
4. Monte un sistema de la siguiente manera:
   a. On Chip memory - RAM 64Kbytes (65536 bytes)
   b. NIOS II processor
      i. Configure el módulo como NIOS II/e
      ii. Recuerde configurar los vectores de excepción y reset en la memoria On-Chip
   c. Un timer “full featured” de 32 bits
      i. Deje desmarcado: “No start/stop control bits” y “fixed period”
      ii. Marque “readable snapshot”
      iii. No es necesario que active las señales de salida.
   d. Agregue el Jtag Uart
5. Cambie los nombres de los periféricos para que reflejen su uso
6. Realice las conexiones necesarias para el sistema
7. Se desea que la memoria interna inicie en la dirección cero. Hacer doble clic en la dirección base la memoria y escribir 0x00000000 y luego en bloquee esta dirección haciendo clic en el símbolo de candado al lado. Luego deje asignaciones automáticas al usar system->assign base addresses
8. Observe y anote las direcciones asignadas como referencia para su diseño.
9. Genere el Test bech desde la interfaz QSYS.
   a. Seleccione la versión “SIMPLE” y language verilog

10. **[SOFTWARE C]**En aplicaciones de tiempo real es importante conocer el tiempo de ejecución de una aplicación, en este caso utilice el timer para determinar el consumo de CPU de la aplicación de ejemplo provista con el proyecto. Utilice el UART para comunicar sus resultados. **Para esta parte no utilice las bibliotecas de altera.**

Parte 2

Discuta sobre los usos de un timer como WatchDog, ¿Qué usos tiene? ¿Qué beneficios y/o complicaciones se pueden tener al usar un WatchDog?

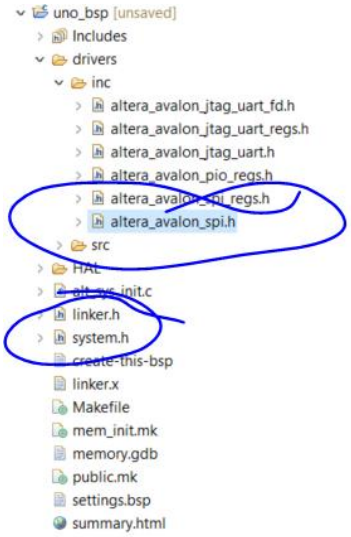
Parte 3

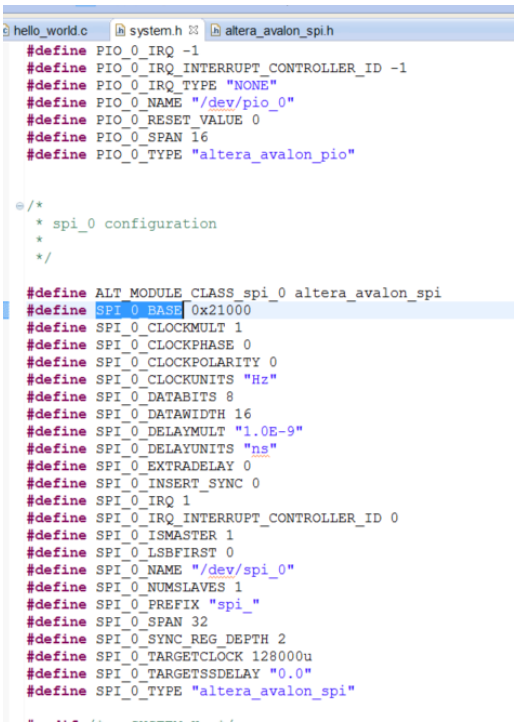
1. Monte un proyecto nuevo en quartus, use como targe la tarjeta "DE0-nano-Soc"
2. Monte un sistema de la siguiente manera:
    a. On Chip memory - RAM 64Kbytes (65536 bytes)
    b. NIOS II processor
        i. Configure el módulo como NIOS II/e
        ii. Recuerde configurar los vectores de excepción y reset en la memoria On-Chip
    c. Un timer "full featured" de 32 bits
        i. Deje desmarcado: "No start/stop control bits" y "fixed period"
        ii. Marque "readable snapshot"
        iii. No es necesario que active las señales de salida.
    d. Agregue el Jtag Uart
    e. Agregue un módulo SPI:
        i. Utilice el Ip core de: Interface protocols-> Serial -> SPI (3wire serial)
        ii. Tipo: Master con un esclavo
        iii. SPI clock: 128000Hz
        iv. Width 8 bits
        v. MSB first
        vi. Polarity: 0 y phase 0
        vii. Exporte el bus SPI
3. Cambie los nombres de los periféricos para que reflejen su uso
4. Realice las conexiones necesarias para el sistema
5. Se desea que la memoria interna inicie en la dirección cero. Hacer doble clic en la dirección base la memoria y escribir 0x00000000 y luego en bloquee esta dirección haciendo clic en el símbolo de candado al lado. Luego deje asignaciones automáticas al usar system->assign base addresses
6. Observe y anote las direcciones asignadas como referencia para su diseño.
7. Genere el Test bech desde la interfaz QSYS.
    a. Seleccione la versión "SIMPLE" y language verilog
8. Genere un módulo que simule el comportamiento de un esclavo conectado a su interfaz SPI.
    a. Solo a nivel de comportamiento.

b. Debe ser capaz de interactuar con las líneas SPICLK, SlaveSelect, MISO y MOSI que genera su sistema empotrado
c. Debe ser capaz generar un set de datos que se envían por el MISO cuando el sistema establece una comunicación
d. Utilice el comando $display para desplegar en el simulador los datos que el "esclavo" recibe desde el MOSI

9. Modifique el test bench generado por QSYS para agregar su módulo "esclavo"

10. **[SOFTWARE C]** En muchos sistemas el uso de protocolos estándar es necesario para comunicarse con diferentes partes del proyecto. SPI es uno de los estándares de comunicación serial más utilizados. Para esta parte, genere una aplicación en C que se comunique con el puerto serial y sea capaz de escribir y leer datos desde la interfaz. **Para esta parte utilice los drivers provisto por altera en el BSP**.





Función: `int alt_avalon_spi_command(alt_u32 base, alt_u32 slave, alt_u32 write_length, const alt_u8 * write_data, alt_u32 read_length, alt_u8 * read_data, alt_u32 flags);`

- Base: es la dirección base del módulo SPI (la encuentra en system.h)
- Slave: el indicador del esclavo a hablar, en nuestro caso solo tenemos uno entonces dejar este valor en 0
- X_length: la longitud de datos a leer/escribir en bytes
- X_data: los punteros del buffer de lectura/escritura.
- Flags: una de dos:
  - `ALT_AVALON_SPI_COMMAND_MERGE`

- `ALT_AVALON_SPI_COMMAND_TOGGLE_SS_N`

En el caso de este Proyecto utilice la primera bandera!

Evaluación

Escriba un artículo de no más de 4 páginas (tipo IEEE, doble columna), donde se describa el proceso de diseño:

1. Proceso de diseño
2. Resultados obtenidos
3. Conclusiones
4. Junto con el informe debe agregar sus archivos fuentes.