Trabajo 1

Diseño e implementación de un SPI maestro

Objetivos:

- Diseño de un protocolo de comunicación básico
- Comunicación entre autómatas en un diseño VHDL
- Diseño de registros de desplazamiento con carga paralelo y conexión serie en cascada
- Aplicación a la comunicación con dos dispositivos comerciales
- Manejo del osciloscopio del programa Proteus
- Trabajo en equipo

Profesor encargado del trabajo 1:

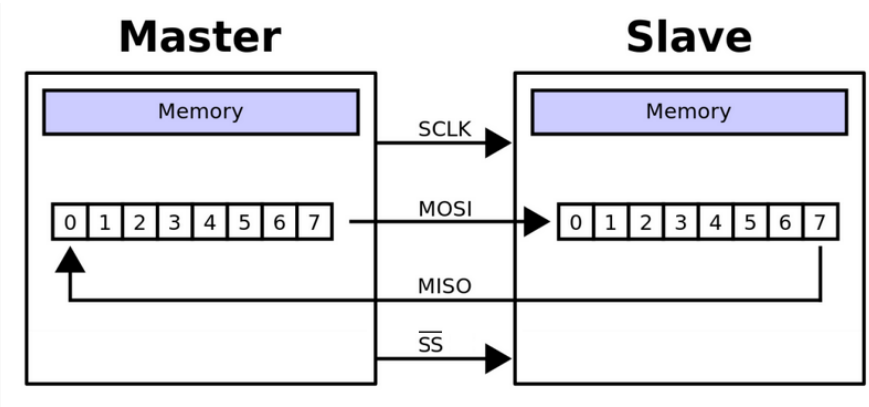
- Millán, Rafael rmillan@us.es

1. Características básicas de la comunicación serie síncrona SPI

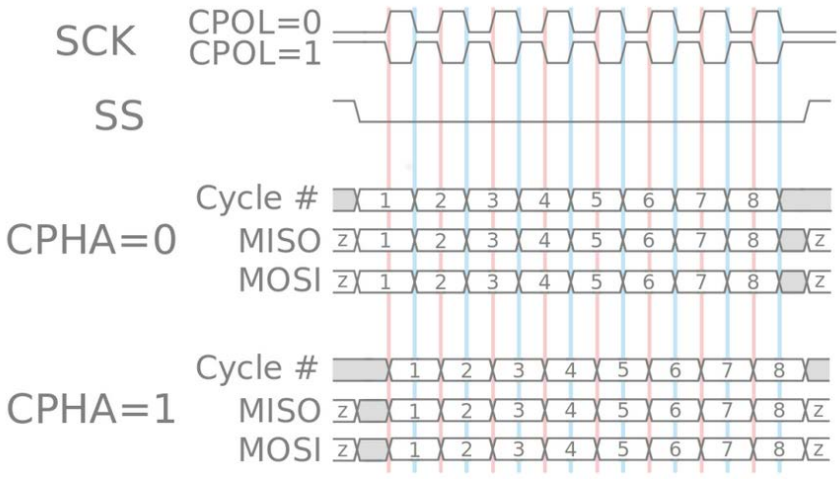
La palabra SPI es un acrónimo de **S**erial **P**eripheral **I**nterface. Permite comunicar un dispositivo maestro con uno o varios esclavos a través de una comunicación serie síncrona. La palabra síncrona implica el envío de una señal de reloj que sincroniza cada bit que se envía en serie. La velocidad de la comunicación depende de la señal del reloj que envía el maestro.

Este tipo de comunicación serie permite ratios de velocidad más altas que la comunicación serie I2C y la asíncrona. La comunicación SPI permite velocidades máximas del orden de varios Megabaudios, el I2C hasta 1Mbaud y la comunicación serie asíncrona hasta 115200 baudios.

En la siguiente figura se muestra el esquema de conexiones en el caso de que haya un solo esclavo. La transmisión comienza por el bit más significativo del registro de desplazamiento. La señal SS (Slave Select) estará a nivel bajo durante la transmisión. El flanco de subida de esta señal –al finalizar la transmisión- es utilizada por muchos esclavos para pasar la información del registro de desplazamiento a un registro interno para su procesamiento (memoria interna).



El maestro y esclavo deben estar configurados con la misma polaridad y mismo flanco activo de reloj. La polaridad la fija el bit CKP (0: el reloj está a '0' en reposo, 1: el reloj está a '1' en reposo). El flanco activo lo marca el bit CKE (0: el bit se transmite en el flanco de reloj que va del estado de reposo a activo –primer flanco del reloj-, 1: el bit se transmite en el flando de reloj que va del estado activo al de reposo –segundo flanco del reloj-).

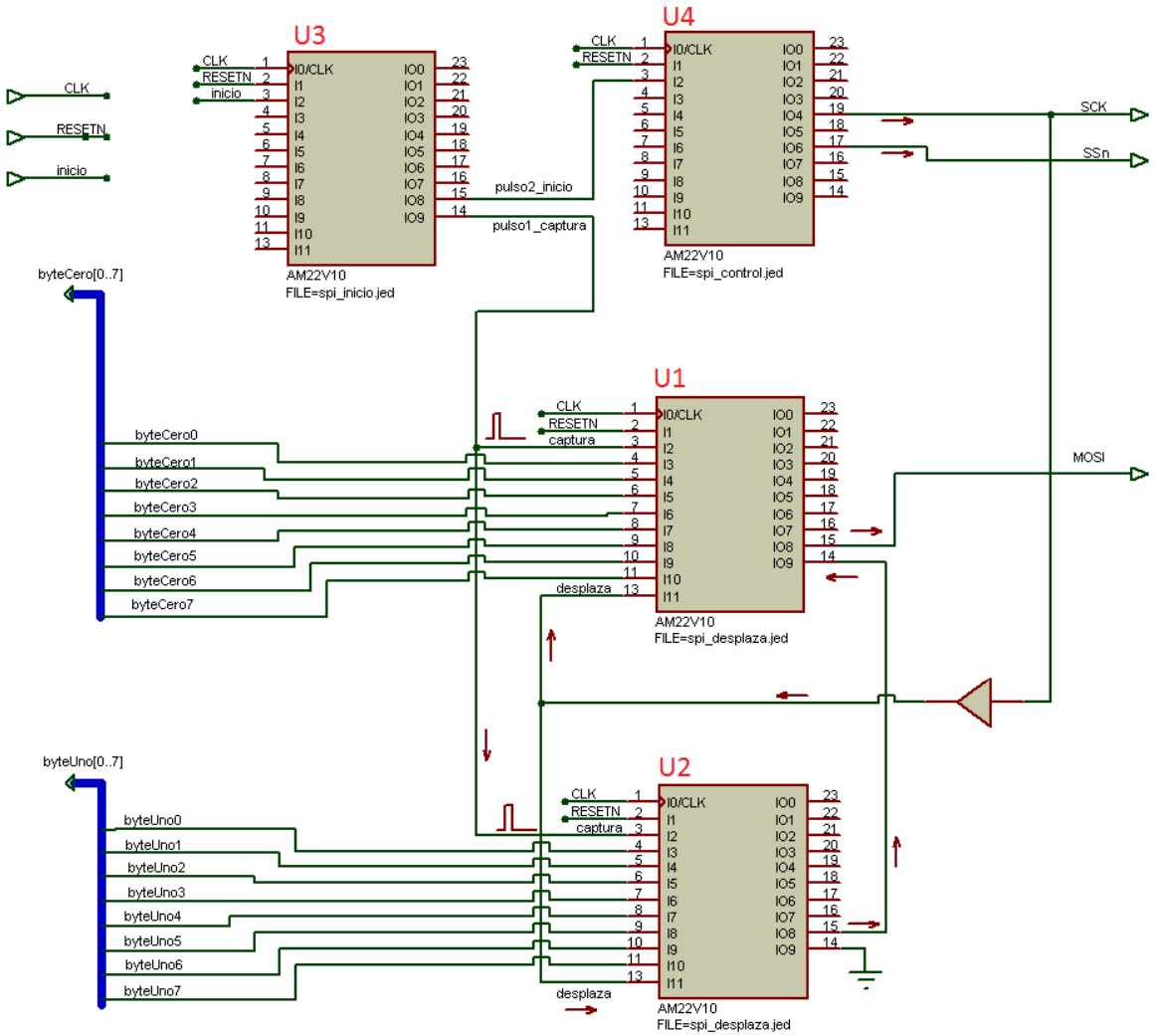


La mayoría de los esclavos disponibles en la tienda reconocen duplas 0,0 y 1,1 para la polaridad y fase, respectivamente.

2. Descripción del trabajo

Se pretende diseñar un hardware específico en VHDL que implemente la comunicación SPI de un maestro que transmitirá tramas de 2 bytes y que no recibirá datos del esclavo. Se simplifica la comunicación full-duplex a semi-duplex. En este trabajo se utilizará el diseño para comunicarse con periféricos comerciales que no transmiten (solo reciben 2 bytes -16bits- del maestro para su gobierno). La fase y polaridad serán 0,0.

El sistema a diseñar constará de 4 dispositivos PAL22V10. Éstas son los componentes del esquema (cajas) etiquetados U1, U2, U3 y U4. Las cajas del circuito etiquetadas U1 y U2 contienen sendos registros de desplazamiento de 8 bits. Ambos realizan la captura en paralelo de un byte y están conectados en serie. De esta forma el byte capturado por U2 se transfiere al registro de desplazamiento de la caja U1. La caja U3 contiene un autómata que genera el pulso inicial de captura y, a continuación, el pulso de inicio de la transmisión. La caja U4 genera la señal de reloj y la señal *SSn* de selección de esclavo. Esta señal estará a nivel bajo durante la transmisión de los 16 bits.

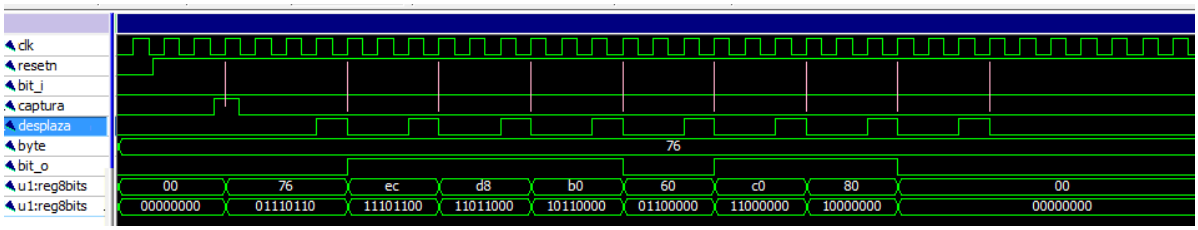


2.1. Registros de desplazamiento (spi_desplaza.vhd)

En los apuntes de VHDL se tiene un diseño de un registro de desplazamiento que desplaza los bits en el sentido contrario al que lo hace el protocolo serie SPI. Además, habrá que añadirles las entradas llamada “captura” y “desplaza”. No hay que olvidar que todo el diseño debe ser síncrono por lo que debe estar dentro del bloque “if (clk’event

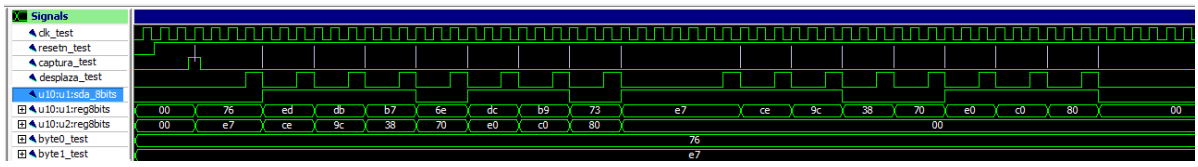
and clk='1') then" que es el que define el flanco de subida de reloj que sincroniza todos los biestables del dispositivo.

El reset inicial pone el registro interno de 8 bits (reg8bits) a cero. Este registro se muestra en la siguiente figura tanto en formato hexadecimal como en binario. Por defecto, el programa no muestra este registro ya que es una señal interna. Si la señal **captura**='1' y llega un flanco de subida de reloj, se capturarán los 8 bits de entrada (**byte**). Si la señal **desplaza**='1' y llega un flanco de subida de reloj entonces se desplazan los bits del registro de forma que el bit más significativo sale por el pin 15 (**bit_o**) y entra un nuevo bit por el pin 14 (**bit_i**). La entrada **desplaza** vendría de un autómata (dispositivo U4) que la genera con una anchura de un ciclo de reloj. Cada pulso de esta señal cubre un flanco de subida del reloj (clk). Es decir, el flanco de bajada de la señal **desplaza** está a la derecha del flanco de subida del reloj CLK aunque en la figura no se aprecie (parecen coincidentes).



2.1.1. Conexión en cascada

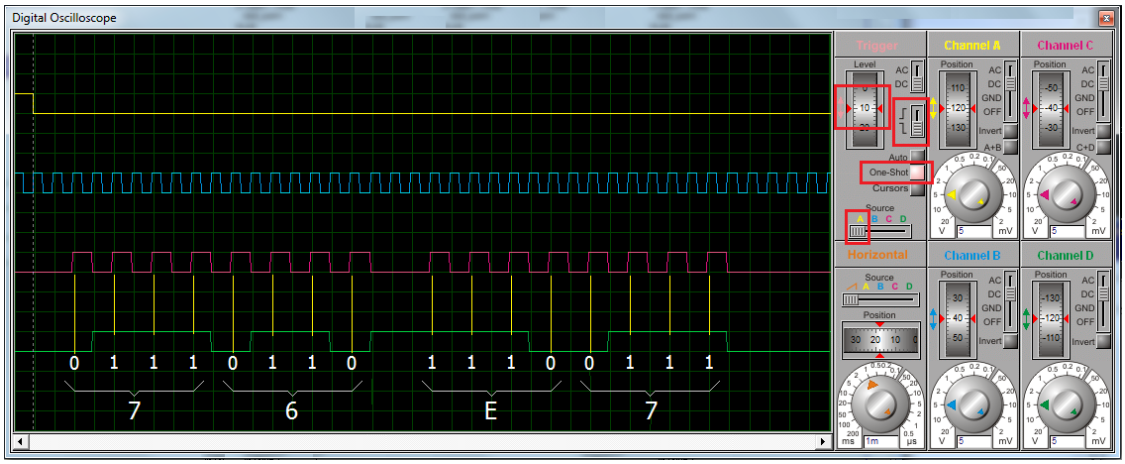
El sistema tiene dos registros de desplazamiento conectados en cascada. Comparten las señales de entrada *clk, resetn, captura* y *desplaza*. El dato de salida –serie- del dispositivo U2 será la entrada serie del dispositivo U1. Si el byte de entrada del dispositivo U1 es 0x76 y el del dispositivo U2 es 0xe7, la simulación del desplazamiento de estos 16 bits será el siguiente:



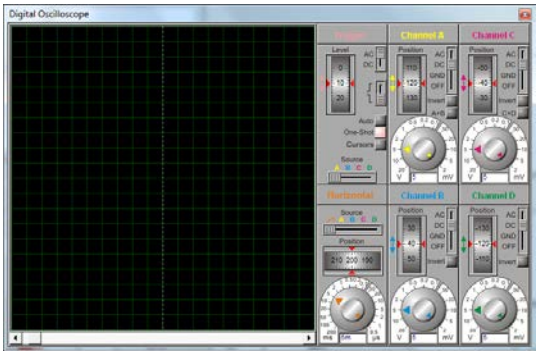
Habrá 16 pulsos en la entrada *desplaza* que permitirán sacar los 16 bits capturados por los dos registros de desplazamiento.

Se ha preparado un fichero de proteus para simular la implementación del diseño. Éste contiene un script (fichero de estímulos) con una señal de comienzo llamada *inicio.* Cuando se da un pulso en esta señal, el script genera un pulso en la señal *captura* (color amarillo), genera el reloj del sistema (clk, color azul) y genera los 16 pulsos de la señal *desplaza* cuya anchura era de un ciclo de reloj (color rojo).

En color verse se tiene la salida en serie de los 16 bits que se han desplazado (0x76+0xE7).



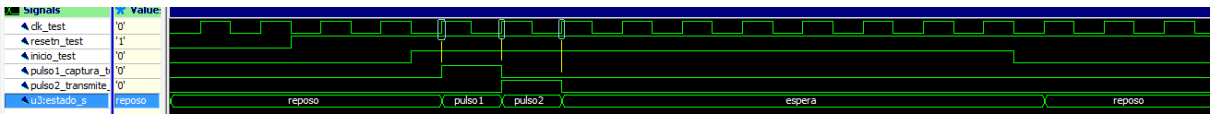
El osciloscopio se ha configurado (ver figura anterior) para que capture de un solo disparo (One-Shot) con el flanco de bajada de la primera señal (el valor de "Source" en la figura será la curva A de color amarillo). Las 4 señales están configuradas para que un cuadro vertical sea 5V. El nivel del disparo (level) se mueve para que esté entre el valor bajo y alto de la señal amarilla. El nivel de disparo se muestra como una señal horizontal blanca discontinua cuando se mueve la rueda "Level". Una vez configurado, se hace click-izquierdo en el botón "One-Shot". El osciloscopio queda a la espera para el disparo sin mostrar nada mientras no se cumpla la condición. Ese estado de espera se muestra en la siguiente figura.



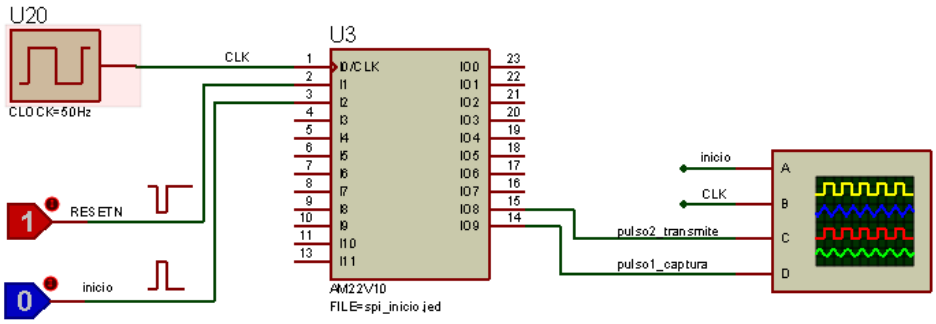
Se dará un reset y, a continuación, se hará click-izquierdo en la señal *inicio* del fichero proteus para que el script genere los estímulos para las entradas: clk, captura y desplaza. Es fundamental dar un reset al comienzo (solo una vez) para que todos los biestables tomen un valor inicial conocido (a cero lógico).

2.2. Autómata para los dos pulsos de comienzo (spi_inicio.vhd)

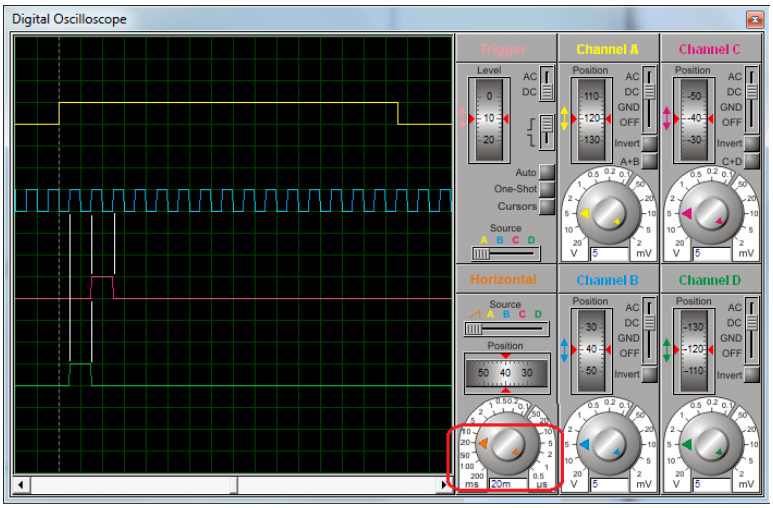
La caja etiquetada U3 contiene un autómata encargado de recibir la entrada *inicio*. Esta señal será un pulso de tamaño indefinido. El autómata tendrá 4 estados: reposo, pulso1, pulso2 y espera. Cuando se detecta el flanco de subida de la señal *inicio,* el autómata cambia del estado de *reposo* al estado *pulso1*. En este estado se activa a nivel alto la salida *pulso1_captura*. Este estado dura un ciclo de reloj. Esto implica que no tiene condición de permanencia o repetición del mismo. Se pasa incondicionalmente al estado *pulso2.* Este nuevo estado también dura un ciclo de reloj. Se activará a nivel alto la salida *pulso2_transmite*. Finalmente, el estado *espera* se mantiene mientras la señal de entrada *inicio* -que disparó el autómata- se mantenga aun activo a nivel alto. Cuando esta señal cambie a nivel bajo, el autómata evolucionará al estado de *reposo.*



El reloj se ha bajado a una frecuencia pequeña en el fichero de proteus para que pueda capturarse el pulso de *inicio* –completo- que tiene un tiempo indefinido. Este pulso es generado por un pulsador cuyo orden de magnitud es grande comparado con un sistema digital.



El resultado del osciloscopio sería el siguiente:



2.3. Autómatas para el control del desplazamiento (spi_control.vhd)

La caja etiquetada U4 contendrá dos autómatas que se hablarán entre ellos:

Trama: Tiene como entrada la señal externa *transmite*. Genera la salida *SSn* que pasa de nivel alto (valor de reposo) a nivel bajo mientras se envía la trama de 16 bits. También tiene la salida *ini_8bits* que será un pulso de un ciclo de reloj a nivel alto para que el otro autómata comience el envío de 8 bits. La señal *ini_8bits* es interna de la arquitectura. Los estados del autómata Trama son los siguientes:

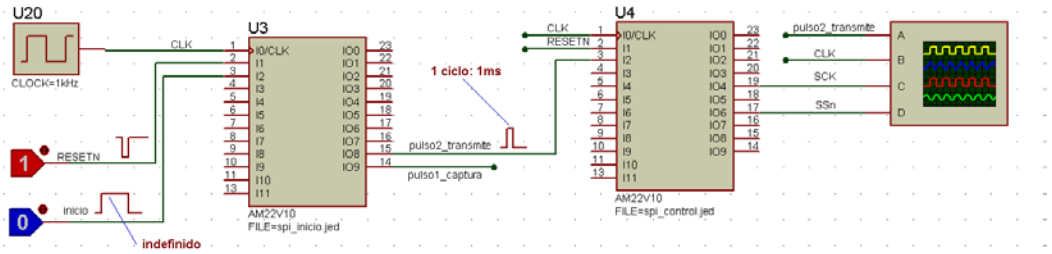
- **reposo**: Permanece en este estado mientras la entrada *transmite* está a nivel bajo. Cuando cambia a nivel alto –un ciclo de reloj-, sale de reposo y va al estado *byte0*.
- **Byte0**: Este estado se mantiene mientras la señal (interna) *fin_8bits* valga 0. Esta señal la activará el otro autómata (llamado *Bits)*. Cuando la señal *fin_8bits* cambia a '1' entonces pasa al estado *separa*.
- **separa**: Este estado será de un ciclo de reloj y pasará al estado *Byte1*. En este estado dará un pulso en la señal *ini_8bits* para que el autómata *Bits* comience un nuevo ciclo para enviar 8 bits.
- **Byte1**: Funciona igual que el estado *Byte0.* Se diferencia en que pasa al estado *reposo* cuando la entrada *fin_8bits* da un pulso de un ciclo de reloj.

Bits: Recibe como entrada la señal interna *ini_8bits* que genera el autómata *Trama* y tiene como salidas las señales *fin_8bits* y SCK. Se apoyará en la señal interna *cuenta_s* para contar 8 y, de esta forma, generar 8 pulsos en la salida de reloj SCK. Los estados del autómata *Bits* son los siguientes:

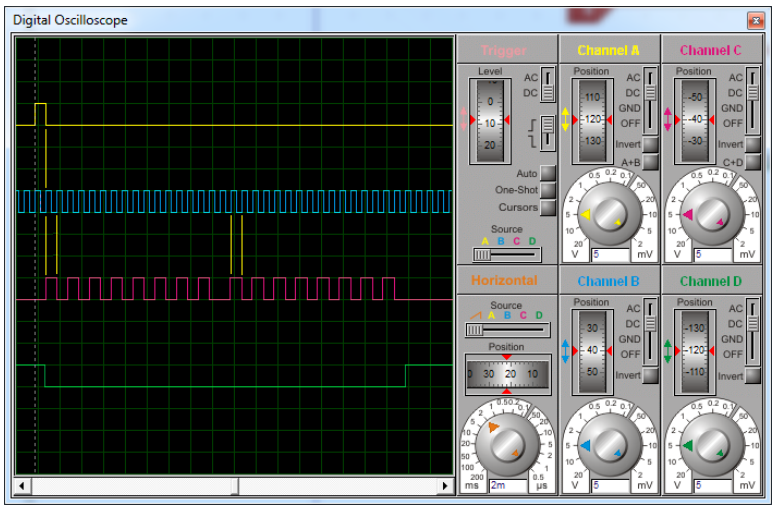
- **reposo**: Permanece en este estado mientras la entrada *ini_8bits* está a nivel bajo. Cuando cambia a nivel alto –un ciclo de reloj-, sale de reposo y va al estado *reloj_1*.
- **reloj_1**: Estado de un ciclo de reloj en el que la salida se pone a SCK='1'. También incrementa la cuenta. Pasa al *estado_0* de forma incondicional.
- **reloj_0**: Este estado será de un ciclo de reloj. La salida SCK se pone a '0'. Si la señal que realiza la cuenta no ha desbordado (ha contado 8) entonces va al estado *reloj_1*, si no va al estado *reposo.* La señal *fin_8bits* se activa en este estado cuando la cuenta ha desbordado.

En la siguiente figura se muestra la simulación de este autómata. Una vez que recibe un pulso en la señal *ini_8bits*.

El componente U4 de la siguiente figura contiene los dos autómatas de este apartado. La señal de entrada que dispara este bloque se genera con la caja U3 vista en el apartado 2.2. De esta forma el pulso de arranque (pulso2_transmite) es de un ciclo de reloj.



El resultado del osciloscopio se muestra en la siguiente figura. Tras el pulso de arranque (color amarillo), se generan 16 pulsos de la señal SCK (reloj de la comunicación serie) y la salida SSn se mantiene a nivel bajo.

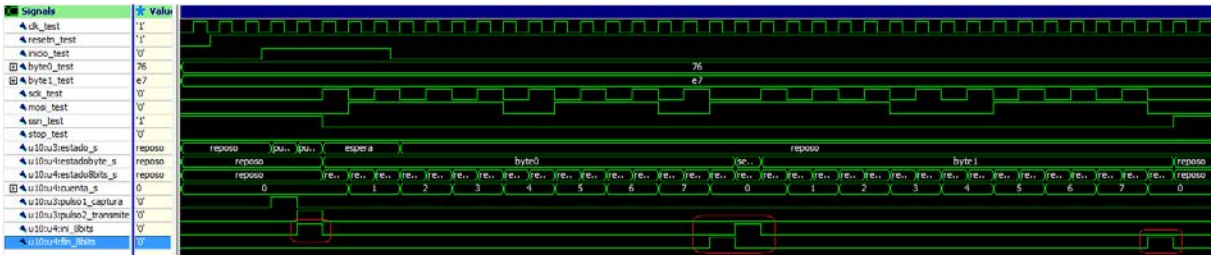


2.4. Simulación del sistema completo (spi_top.vhd)

Es necesario un diseño jerárquico en el que un nuevo componente sea la unión de las 4 cajas que hay en el diseño. Ese componente se llamará spi_top (ENTITY). Se deja el

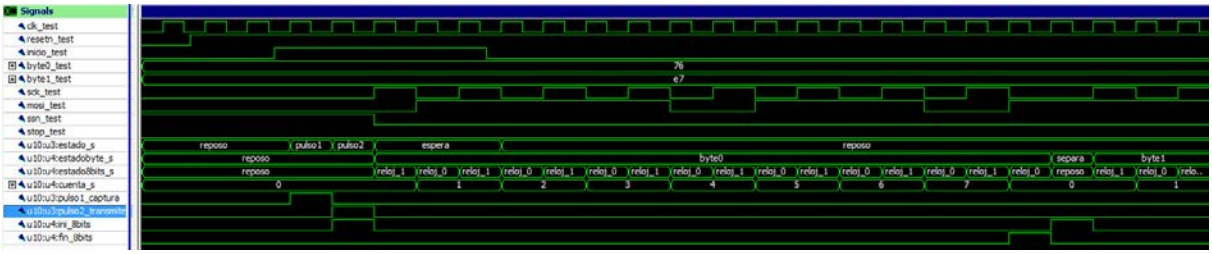
fichero con este componente ya resuelto. También se dejará el fichero de estímulos parcialmente resuelto para este nuevo componente que es el que se va a simular.

En la siguiente figura se muestra el resultado de la simulación. El sistema solo necesita un pulso en la señal *inicio* de cualquier tamaño. El autómata de control sale de reposo y manejará el resto de elementos para transmitir los dos bytes. El resultado de la simulación es el siguiente para el caso de que se transmitan los bytes 0x76 y 0xe7. Es muy importante hacer visibles los estados de los autómatas (señales internas para la simulación).



En color rojo se tienen los pulsos en la señal INI_8bits y FIN_8bits.

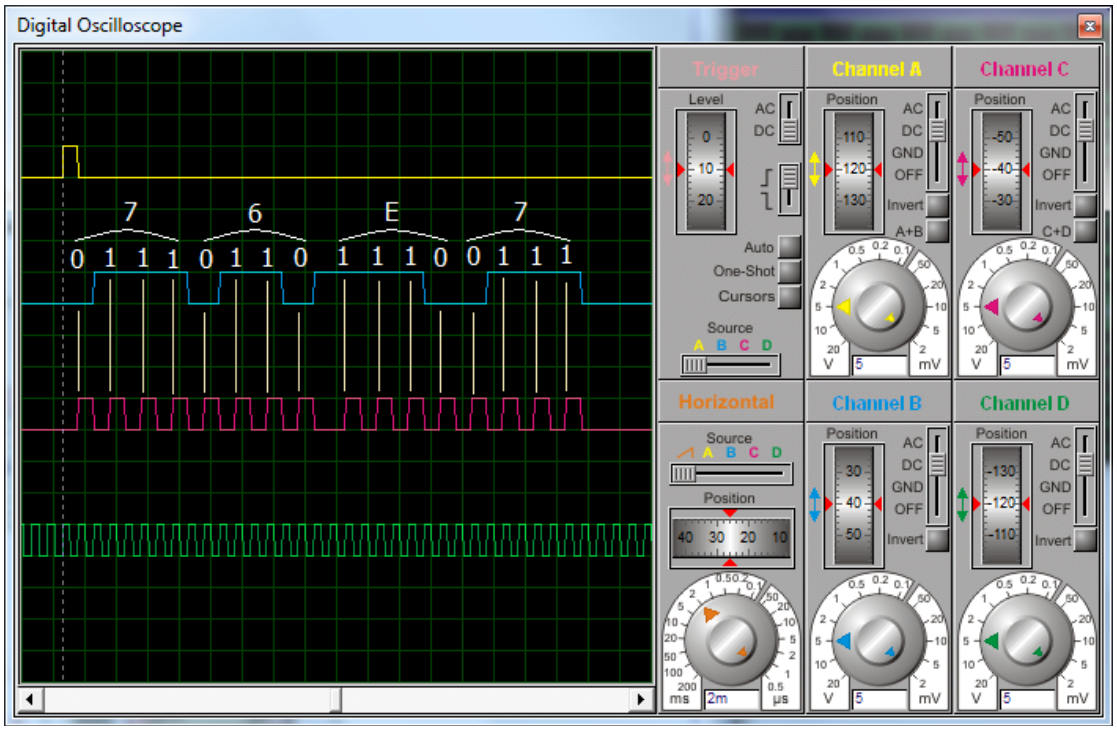
Para ver con mejor grado de detalle la evolución de los estados, se hará un zoom en la gráfica anterior de la parte inicial de la simulación.



2.5. Implementación y simulación en proteus del sistema completo

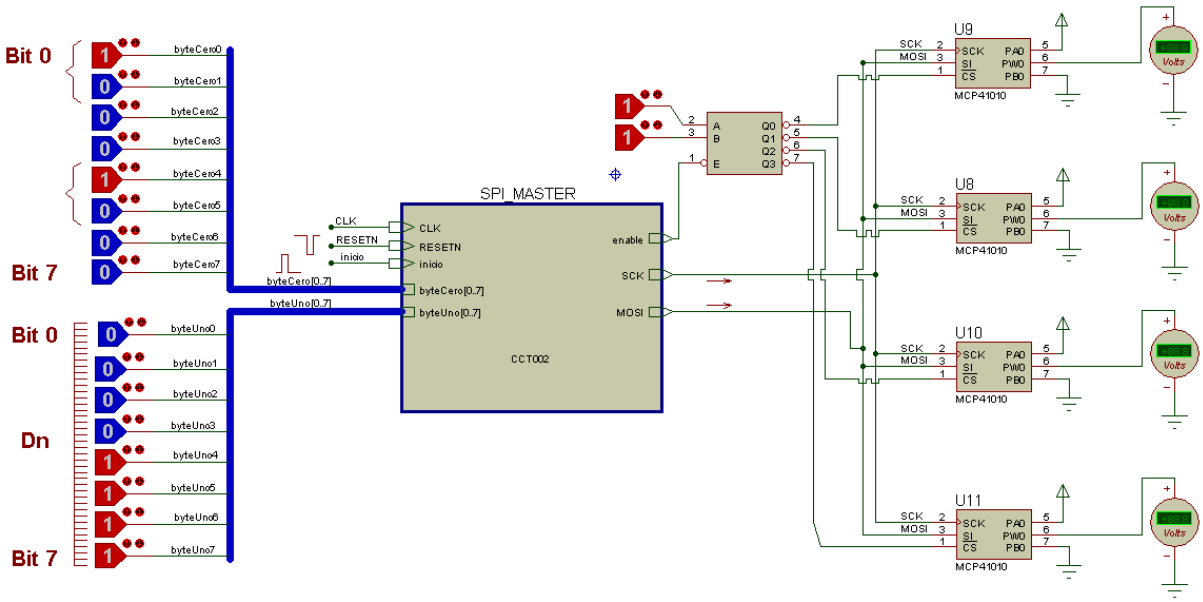
La comunicación comenzará cuando se reciba un flanco de subida en la entrada INICIO. Es muy importante dar un pulso de reset (solo uno) cuando se inicie una simulación del sistema. El esquema del fichero de proteus es el del figura de la página 3.

Se enviarán los bytes 0x76 y 0xe7 al exterior. Si hubiera conectado un esclavo, recibiría dichos bytes transmitidos en serie. Tras dar un reset y un pulso en *inicio*, se tiene el resultado en el osciloscopio con la opción de disparo (One-shot).

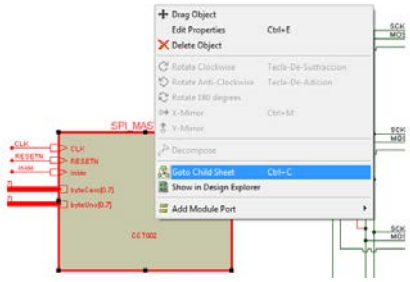


2.6. Diseño jerárquico en proteus para controlar dispositivos con protocolo SPI

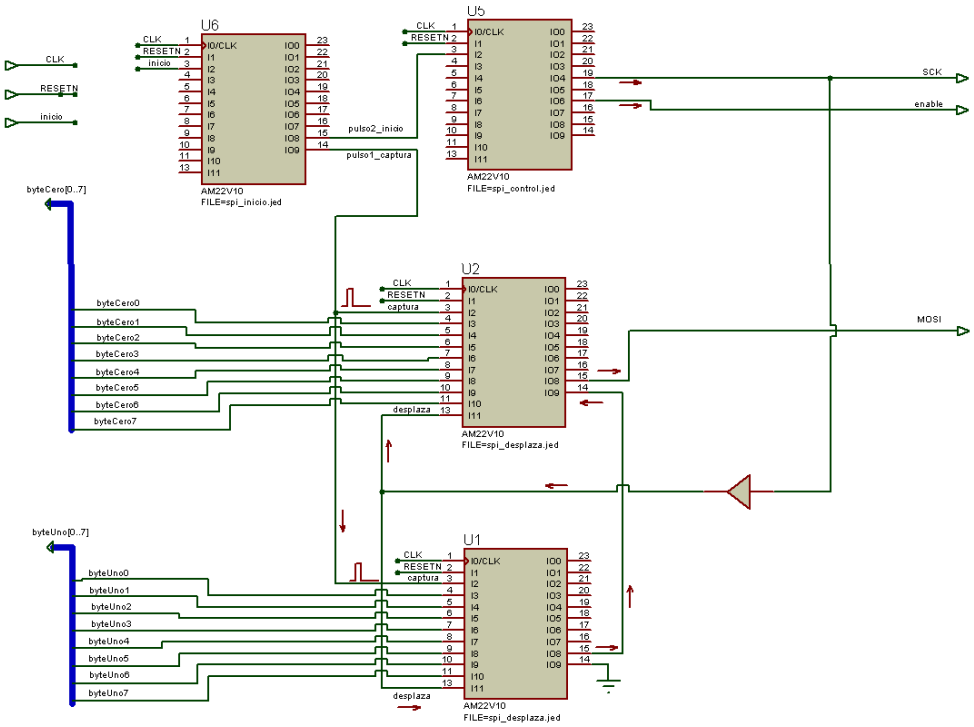
Este diseño podría hacerse en una única hoja en Proteus pero todos los componentes quedarían muy pequeños. En su lugar se ha creado un subcircuito (caja) que contiene el diseño completo de nuestro disposito maestro SPI. Los 4 dispositivos que aparecen a la derecha en la siguiente figura serían esclavos que esperan una comunicación SPI en la que se transmita 2 bytes para su configuración. El decodifidor de 2 a 4 permite elegir con cual dispositivo se va a comunicar nuestro maestro.



La caja SPI_MASTER contiene todo nuestro diseño. Si se hace clik-derecho en dicha caja entonces aparece un nuevo menú que permite bajar un nivel en la jerarquía. Así se accede a otro esquemático que contiene el trabajo realizado hasta ahora.



El esquema que aparecería sería nuestro diseño:



Para volver a la página principal de la jerarquía se hace clik-derecho sobre el esquema y se selecciona la opción “Exit to parent sheet” en el menú flotante que aparece.

3. Ejercicios

1) Diseño y simulación de un registro de desplazamiento. Tendrá señal de entrada de captura paralelo y habilitación de desplazamiento. Simular que tiene una entrada de 8 bits que será el número **0x93.** Recibe un pulso de captura y 8 pulsos de la señal de entrada desplaza (habilitación del desplazamiento) de forma consecutiva. Ambas señales de entrada tienen una anchura de un ciclo de reloj. El reloj del sistema será de 1khz. **(1 punto)**
2) Diseño y simulación de dos registros de desplazamiento conectados en cascada según el esquemático del sistema. Tras la señal de captura vendrán 16 pulsos consecutivos de la señal desplaza. Ambas tienen una anchura de un ciclo de reloj (1ms). Simular que el registro de desplazamiento de la caja U1 tiene una entrada de 8 bits que es el número **0x93** y el registro de desplazamiento de la caja U2 tiene una entrada de 8 bits que es el número **0x0F**. **(0.5 puntos)**.
3) Implementación del apartado 2. El resultado se capturará con el osciloscopio de proteus en modo disparo con el flanco de bajada de la señal *captura*. Se copiará y pegará la imagen del osciloscopio en la memoria. El resto de señales conectadas al osciloscopio serán *clk*, *desplaza* y *bit_o*. Se recuerda dar un *reset* antes de dar el pulso en la señal de *inicio* que disparará la captura del osciloscopio. Pegar el resultado del osciloscopio a la memoria. **(0.5 puntos)**
4) Simulación del autómata de la caja U3 (spi_inicio). Se mantiene en reposo mientras la entrada inicio está a '0'. Cuando ésta da un pulso de un tamaño indefinido, generará sendos pulsos consecutivos de una anchura de un ciclo de reloj en las salidas pulso1_captura y pulso2_transmite, respectivamente. **(2 puntos)**
5) Implementación y simulación en proteus del apartado 4. Se comprobará con el osciloscopio de proteus que las salidas son correctas. Pegar el resultado del osciloscopio a la memoria. **(0.5 puntos)**.
6) Simulación de los dos autómatas de la caja U4 (spi_control). El trabajo coordinado de los autómatas *Trama* y *Bits* permite que la comunicación comience cuando la entrada *transmite* cambie de '0' a '1'. Esta entrada procede de la salida de la caja *U3:pulso2_transmite* (anchura de un ciclo de reloj). Cuando ésta da un pulso, el sistema de la caja U4 dará 16 pulsos de la salida SCK de un ciclo de reloj y durante este tiempo mantendrá la salida *SSn* a '0'. **(2.5 puntos)**
7) Implementación y simulación en proteus del apartado 6 (spi_control). Se comprobará con el osciloscopio de proteus que las salidas son correctas cuando la entrada *transmite* recibe un pulso. Se pegarán los datos del osciloscopio en la memoria **(0.5 puntos)**.
8) Simulación del sistema completo mediante diseño jerárquico. Se transmitirá la trama constituidos por los bytes **0x93** y **0x0F**. El diseño jerárquico se deja completo (fichero sed_top.vhd) (1 **punto**).
9) Implementación del sistema completo utilizando las 4 cajas (U1, U2, U3 y U4). Se copiarán las curvas del osciloscopio en la memoria para el caso de transmitir los bytes **0x93** y **0x0F** **(0.5 puntos)**.
10) Hay que leer el catálogo del componente mcp41010. Se deja un PDF o se puede buscar otro en internet. Se dispone del fichero Proteus con un diseño jerárquico con 4 dispositivos mcp41010. Se configurarán los 16 bits que se van a transmitir (entradas del fichero proteus) para que el último potenciómetro SPI genere una salida de 4,74V. Hay que seleccionar el dispositivo con el decodificador. Se capturará la pantalla con los valores de las entradas (bytes) y decodificador asignadas **(1 punto)**.

4. Observaciones

- Fecha de finalización: **12 de abril a las 24h**.
- Se evaluará la presentación del fichero PDF así como el código. Se penalizará si el código no está bien comentado ni indentado. Hay que sustituir tabuladores por espacios con el Notepad++ (Edit-> Opciones de limpieza -> TAB a espacios) en todos los ficheros
- Sólo se permitirán grupos de 3 o 4 miembros. Los grupos con 2 miembros o con 5 miembros tendrán una penalización del 5% de la nota. No se admiten grupos con un único alumno ya que no se cumplen los objetivos de la asignatura del fomento del trabajo en equipo
- Los diez primeros trabajos serán bonificados con un 10% extra de nota. Si se tiene un 10 se almacenará un 11 (para hacer media con el otro trabajo). Solo se permite subir los trabajos una sola vez a la enseñanza virtual. Hay que estar seguro de lo que se envía
- El trabajo se subirá a la EV con el título “TRABAJO_1”. Se adjuntará un fichero comprimido ZIP. Se incluirá la misma estructura de la plantilla sustituyendo el fichero MEMORIA_trabajo1.docx por el fichero MEMORIA_trabajo1.pdf obtenido a partir del anterior.
- Todos los miembros del grupo deben subir el trabajo para que el sistema permita evaluarlos. En la primera página del fichero MEMORIA_trabajo1.pdf se indicará el nombre de todos los miembros del grupo de forma clara. La fecha de entrega se considerará la del último miembro del grupo que suba la práctica
- Si el trabajo se entrega fuera de plazo se penalizará con un 10% por cada día de retraso respecto a la fecha límite
- Los ficheros entregados se pasarán por un programa que detecta el porcentaje de similitud con los ficheros entregados por otros grupos. Si se detecta que dos grupos se han copiado, la nota del trabajo se dividirá entre dos y se asignará a los miembros de ambos grupos (sin distinguir qué grupo lo hizo y cual lo copió). Si son 3 los grupos que copian, la nota se dividirá por 3 y así sucesivamente
- El principal medio de comunicación -para atender dudas- será el foro de la asignatura. Uno de los objetivos de los trabajos de la asignatura es fomentar el trabajo en equipo. Con el foro se consigue mejor estos propósitos ya que se comparten las dudas y las respuestas. Aquellas dudas que no sean contestadas o tengan respuestas confusas, las responderá el profesor. El foro permite mensajes anónimos. Cualquier tutoría atendida por el profesor (bien por correo electrónico o en el despacho) se subiría al foro