T80 debugging con NoICE - Un breve tutorial

Leonardo Etcheverry

Junio 2010

Introducción

Este tutorial describe el procedimiento para el depurado de un programa de Z80 corriendo en un sistema basado en el core T80 con la herramienta NoICE.

Partiendo de un sistema sintetizable en la placa DE0, formado por un procesador T80, RAM, ROM y UART, se describe el proceso de preparación del programa **monitor** del debugger NoICE, la posterior carga de un programa sencillo a la RAM del sistema para su depuración desde la interfaz gráfica de NoICE.

Requisitos

Conjunto de herramientas y archivos necesarios supuestos por esta guia:

- Herramientas

  - Quartus 9.0
  - NoICE para Z80 target v9.3
  - PseudoSam 1.3 (ensamblador de Z80)

- Archivos provistos

  - t80_debug_noice.zip Proyecto de Quartus con el hardware del sistema.
  - monz80.zip monitor ajustado para este hardware (fuente y binario en formato Intel Hex)
  - testprog.zip programa simple de ejemplo para depurar (fuente, binario en formato Intel Hex y listing con información de debugging.)

Hardware

El proyecto de Quartus (t80debug.qpf) especifica un sistema con:

- procesador T80
- 4K RAM (mapeada en 0x8000)
- 1K ROM (mapeada en 0x0000)
- UART0 (IO 0x80)
- 1 puerto salida (0xA0)

Señales de entrada/salida del FPGA

Salidas

Señal	DE0
heightPuerto salida	LEDG[7..0]
UART_TXD	GPIO1_D8 (pin 13 del expansor J5)
LEDRX	LEDG8
LEDTX	LEDG9

Entradas

Señal	DE0
heightUART_RXD	GPIO1_D9 (pin 14 del expansor J5)
CLOCK_50	Oscilador de 50 MHz
RESET	BUTTON0

Prueba rápida del sistema.

1. Compilar el proyecto en Quartus y generar el archivo de programacion **t80debug.sof**.

   El proyecto ya incluye un archivo **monitor.hex** con el monitor ensamblado para este hardware.

2. Programar la placa con **t80debug.sof**.

3. Conectar la placa DE0 y el PC donde corre NoICE a través del cable serie.

4. Iniciar el NoICE, elegir como protocolo de comunicación **NoICE Serial Protocol**, y elegir el puerto COM adecuado.

   El monitor en el T80 está configurado para comunicarse con 19200 8N1.

5. Si la comunicación es exitosa, En el panel "Data" del NoICE debería aparecer la versíon del monitor: **NoICE Z80 monitor V3.0. Target type Z80/Z180. Target buffer 67 bytes**.

   En caso que NoICE de timeout al comunicarse con el target, es útil ver los LEDS 8 y 9 para descartar problemas de conexión del cable serie.

6. En NoICE, ir al menu "Processor", "Out". escribir FF en el puerto de salida A0. Los LEDS[7..0] deberían encenderse.

7. Finalmente, cargar el programa de prueba a RAM. Ir al menu "File", "Load" y seleccionar **test.hex**.

   El programa debería cargarse en RAM, y quedar listo para ejecutarse. Puede ejecutar el programa paso a paso con F8.

   Verificar que las instrucciones coinciden con el programa **test.asm**

8. Hasta ahora simplemente se cargó el binario en memoria y se ejecutó paso a paso.

   No hay información de símbolos asociada al programa. Para cargar la información de debugging, ir a "File", "Play" y seleccionar el archivo **test.noi**.

   Este archivo asocia número de linea en archivos fuente, símbolos y direcciones de memoria. Luego, por "View" elegir "Select Source File".

   En la ventana de selección de Source File, elegir **test.asm**. Ahora puede depurar el programa cargado contra el fuente original.

Flujo de trabajo: ensamblando y depurando un programa

Esta sección describe el flujo de trabajo para ensamblar y luego depurar un programa con NoICE asumiendo un monitor correctamente configurado.

Ensamblando el programa a depurar.

Primero se ensambla el programa que se desee depurar, generándose además información de debugging.

Archivos de entrada	**test.asm**
Archivos de salida	**test.obj, test.lst, test.hex**
Herramientas	Pseudosam (**a80z.com**)

Para ensamblar el programa, usando Pseudosam

```
a80z test.asm
```

El proceso de ensamblado generará dos archivos, **test.obj** que contiene el binario del programa en formato Intel HEX; y **test.lst** que contiene información de debugging, asociando número de línea, símbolos y direcciones de memoria.

Copiar (o renombrar) **test.obj** a **test.hex**. Esto permitirá cargar el programa en formato HEX desde el NoICE más facilmente.

Generando información de debugging para NoICE.

Archivos de entrada	**test.lst**
Archivos de salida	**test.noi**
Herramientas	**samsym.exe**

Este paso permite convertir la información de debugging generada por Pseudosam (lst) en un formato apropiado para el NoICE (.noi).

Para obtener **test.noi** a partir de **test.lst**, utilizar la herramienta **SAMSYM** distribuida con NoIce:

```
samsym.exe test.lst test.noi
```

Una vez obtenido el archivo hex con el binario del programa (**test.hex**) y la información de debugging de NoICE (**test.noi**); se puede cargar el programa en la RAM del target a través de los comandos LOAD y PLAY.

Modificando el monitor para otro hardware.

Esta sección explica como modificar el monitor. Puede ser necesario en caso de cambiar el hardware sustancialmente, o si se desea cambiar alguna otra configuración del monitor.

Archivos de entrada	**monz80.asm**
Archivos de salida	**monz80.obj, monitor.hex**
Herramientas	Pseudosam (**a80z.com**)

Para una configuración de hardware dada, el monitor debe ser modificado para incluir la información mínima necesaria que describe en donde están mapeados los componentes básicos para su funcionamiento (ROM, RAM, UART.)

Memoria

En **monz80.asm** deben ajustarse los siguientes EQUs que describen el mapeo de memoria.

ROM_START	Dirección donde comienza el código del monitor (base ROM)
USER_CODE	Dirección de comienzo de la memoria del usuario, (base RAM)
RAM_START	Dirección de comienzo de la RAM reservada para el monitor (algun lugar en RAM)

UART

Para la UART que utilizará el monitor deben configurarse la dirección de IO en la que está mapeada, y el divisor para generar el baudrate correcto de acuerdo al reloj del sistema.

El mapeo de IO se realiza asignando el siguiente EQU:

S16450	Dirección de IO base de la UART

El ajuste del divisor de baudrate se hace modificando el código de inicialización de la UART.

```
;  fixed baud rate of 19200:  crystal is 3.686400 Mhz.
;  Divisor is 3,686400/(16*baud)
        ld      a,H'A1                  ;fix at 19.2 kbaud
        out     (S16450+RXR),a          ;lsb
```

El valor del divisor, en este ejemplo 161 (H'A1), se calcula como

```
divisor = CLK / (16 * baudrate)
```

En el ejemplo, se está calculando el divisor de forma de obtener un baudrate de 19200 bps con un reloj del sistema CLK de 50MHz.

Ensamblado del monitor

Para ensamblar el monitor luego de modificarlo ejecutar el ensamblador Pseudosam [1]:

```
a80z monz80.asm
```

El binario resultante será **monz80.obj**, y esta en formato Intel HEX. Copiar este archivo (o renombrarlo) a **monitor.hex** y copiarlo a la estructura de directorios del proyecto de Quartus. Así,la ROM onchip del sistemá tendrá entonces el monitor.

[1] Asegurarse de usar PseudoSam 1.3 o posterior. Las versiones anteriores ensamblan mal los saltos relativos.