PRÁCTICA DE SÍNTESIS nº2

Sistemas secuenciales: detector de trama y contador

1.- Introducción

En esta práctica hay que implementar un detector de trama a partir de una máquina tipo Mealy. El detector debe poner su salida a 1 cuando se reciba la trama 1011. Posteriormente, se debe implementar un contador que dé cuenta del número de veces que se detecta la trama.

La trama se generará a partir de un sencillo módulo que se comentará más adelante. Además, la salida del contador se conectará al display de 7 segmentos de la placa Spartan-3, por lo que habrá que implementar también un decodificador de 7 segmentos.

En la figura 1 aparece es sistema que debe implementarse en esta práctica, donde figuran los bloques mencionados anteriormente: el generador de tramas, el detector de trama, el contador y el decodificador de 7 segmentos.

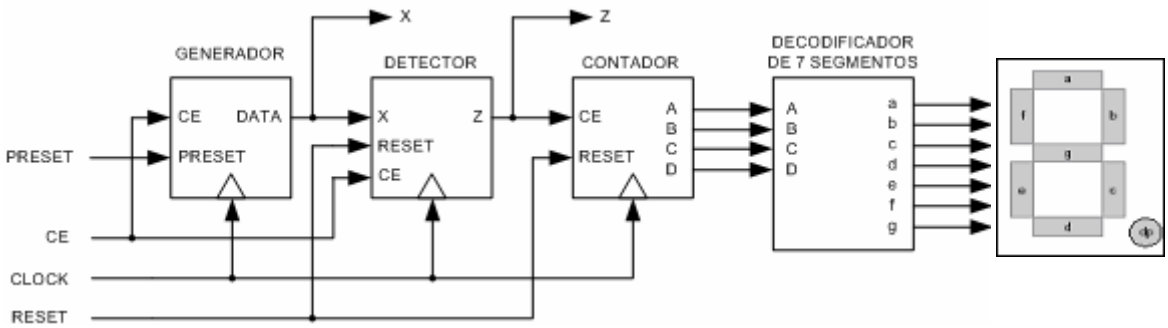

Figura 1. Esquema del sistema a implementar.

El sistema contiene 4 señales de entrada que, básicamente, son de control y sincronismo. La señal *ce* habilita el funcionamiento del generador y el detector, *preset* y *reset* inicializan el generador y el contador respectivamente, y *clock* provee la velocidad de funcionamiento del sistema.

Las salidas que nos interesan son las señales de la *a* a la *g* que atacan el display de 7 segmentos en que visualizaremos la cuenta del número de tramas válidas detectadas. No obstante, también visualizaremos los datos que proveen el generador (señal *x*) y el detector (señal *z*) a través de dos leds, para comprobar el correcto funcionamiento en la placa de desarrollo.

2.- Pasos a seguir

2.1. DISEÑO DEL GENERADOR DE TRAMAS

Para poder suministrarle datos al detector de tramas deberemos crearnos un generador de secuencias. El más popular y simple circuito generador de secuencias pseudo-aleatorias consiste en la realimentación de un circuito de desplazamiento. La realimentación se produce por medio de una puerta XOR que realiza la operación entre dos bits del registro de desplazamiento. El esquema del generador a implementar se muestra en la figura 2.

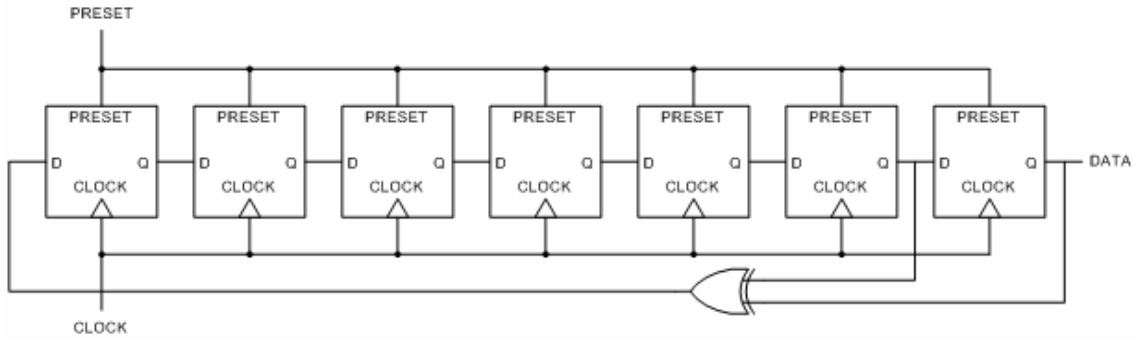

Figura 2. Esquema del generador de secuencias pseudos-aleatorias. En la figura no se muestra la señal CE, común a todos los biestables, por cuestiones de claridad en la figura, pero dicha señal hay que considerarla en el diseño.

Para implementar este circuito, primero hay que diseñar el flip-flop tipo D con PRESET asíncrono de la figura 2 (crear un código VHDL de nombre **ffD_preset**). Este tipo de componente se infiere usando el proceso explícito que aparece en la figura 3.

```
38  architecture Behavioral of ffD_preset is
39
40  begin
41
42  process (clk, preset)
43  begin
44     if preset='1' then
45        q <= '1';
46     elsif (clk'event and clk='1') then
47        if ce = '1' then
48           q <= d;
49        end if;
50     end if;
51  end process;
52
53  end Behavioral;
```

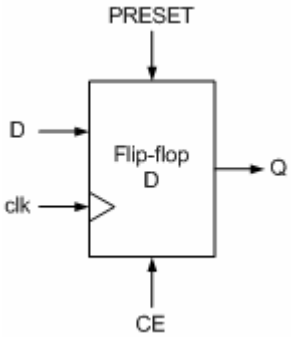

Figura 3. Extracto del código de la arquitectura que describe el biestable tipo D con CE y PRESET asíncrono; y esquema del componente con sus puertos de entrada y salida.

Sin entrar en pormenores, las ecuaciones booleanas que se implementaron en la práctica anterior, el multiplexor 2 a 1 de 1 bit, la suma, el acarreo de salida, etcétera; son procesos implícitos, aunque también pueden expresarse según sintaxis de la figura 3. En la línea 42 aparece la denominada lista de sensibilidad del proceso. Éste no se ejecutará a menos que haya un cambio en alguna de las señales presentes en esta lista.

Los procesos explícitos son muy importantes, ya que podemos describir un sistema digital de una manera algorítmica, facilitando la vida a los ingenieros electrónicos y reduciendo considerablemente el tiempo de diseño.

Una vez creado el componente **ffD_preset**, éste debe ser instanciado 7 veces acorde a la figura 2 en un nuevo archivo, que se puede llamar **generador**. Emplear para ello las cláusulas COMPONENT y PORT MAP, y señales internas de conexión, según vimos en el tutorial y en la práctica anterior.

NOTA IMPORTANTE: la señal de salida *data*, debe ser leída para evaluar la puerta XOR, pero no puede hacerse directamente pues, por la naturaleza de *data*, ésta sólo puede escribirse. Es por ello que hay que asignar la salida de este último biestable a una señal interna (que por tanto puede leerse y escribirse), y luego hacer la asignación de esta señal interna a la señal *data*.

Tras diseñar el componente, éste debe simularse para comprobar el correcto funcionamiento. En la figura 4 se muestra la simulación funcional del generador de tramas pseudos-aleatorias. Éste, obtiene tal denominación ya que, aunque la salida que produce parece aleatoria, resulta la repetición de un determinado patrón. Para este diseño, generador repite la trama cada 2^7=128 pulsos de reloj, más que suficiente para nuestra aplicación.

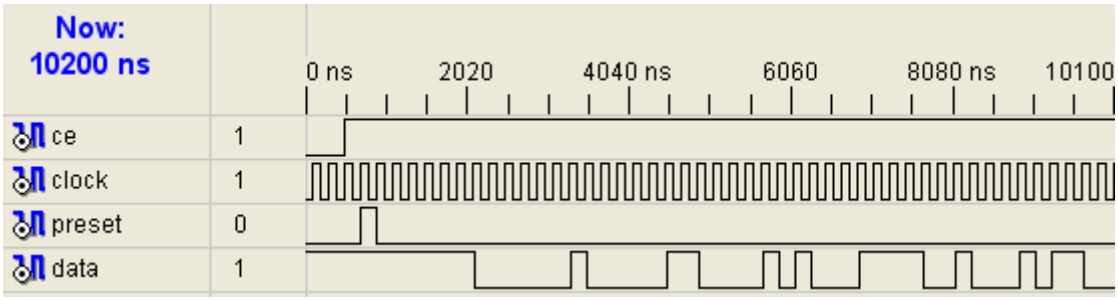

Figura 4. Simulación de la secuencia pseudos-aleatoria generada.

Por último, señalar que, en este componente, se emplea un PRESET para inicializar el circuito y generar la trama. Si usáramos un biestable con RESET, resulta evidente que la salida de la puerta XOR de la figura 2 sería 0 siempre, por lo que la salida tendría este mismo valor indefinidamente.

Las interfaces del generador, y de los otros bloques de los que consta esta práctica se pueden extraer de la figura 1, donde se muestra el sistema global. Se recuerda que estas interfaces deben serle suministrada al asistente de cada módulo VHDL creado para crear la ENTITY del código de cada componente.

2.2. DETECTOR DE TRAMAS

El detector de tramas es un circuito secuencial síncrono que detecta la secuencia de datos 1011 con solapamiento, esto es, se trata de un circuito con una salida Z tal que Z(t)=1 si, y sólo si, los cuatro últimos valores de la entrada X han sido, en este orden, X(t-3)=1, X(t-2)=0, X(t-1)=1 y X(t)=1. Debido a cómo el anterior módulo genera los valores de X, estos ya son síncronos, así que lo mejor es realizar el diseño mediante una

máquina tipo Mealy, pues el número de estados del diagrama del sistema será menor y la implementación será más sencilla.

En este caso vamos a emplear flip-flops tipo D con CE y RESET asíncrono, así que habrá que crear un código VHDL de nombre **ffD_reset** para poder instanciar este componente en la máquina de estados. La arquitectura que describe la funcionalidad de este componente se muestra en la figura 5. Puede apreciarse que el código es muy parecido al de la figura 3, salvo la distinta consideración respecto a las señales RESET y PRESET.

```
38  architecture Behavioral of ffD_reset is
39
40  begin
41
42  process (clk, reset)
43  begin
44     if reset='1' then
45        q <= '0';
46     elsif (clk'event and clk='1') then
47        if ce = '1' then
48           q <= d;
49        end if;
50     end if;
51  end process;
52
53  end Behavioral;
```

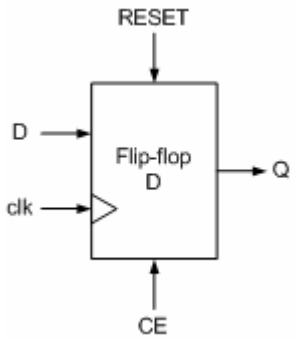

Figura 5. Extracto del código de la arquitectura que describe el biestable tipo D con CE y RESET asíncrono; y esquema del componente con sus puertos de entrada y salida.

El diseño de este detector de secuencias a través de una máquina tipo Mealy conlleva un diagrama de estados con 4 estados, así que necesitaremos 2 de los biestables mencionados anteriormente. El código realizado debe ser simulado para comprobar el correcto funcionamiento del mismo (figura 6).

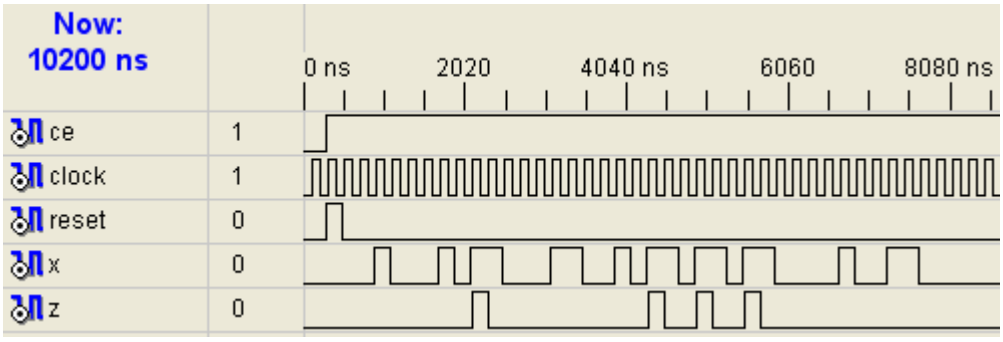

Figura 6. Simulación funcional del detector de tramas.

2.3. CONTADOR

Un contador de 4 bits debe diseñarse para ir contando las veces que se cumple la secuencia, para ello usaremos 4 biestables tipo D iguales que en el anterior apartado. El contador debe recorrer la secuencia binaria de 0 a 9 de forma cíclica (de 9 pasa a 0 de nuevo). La figura 7 muestra cómo debe comportarse este módulo.

2.4. DECODIFICADOR DE 7 SEGMENTOS

La salida del contador la debemos de tratar para poder visualizar el número de cuentas en un display de 7 segmentos. Por tanto, el último bloque debe ser un decodificador de 7 segmentos como el visto en clase y que responde a la tabla de la verdad que aparece en la tabla 1.

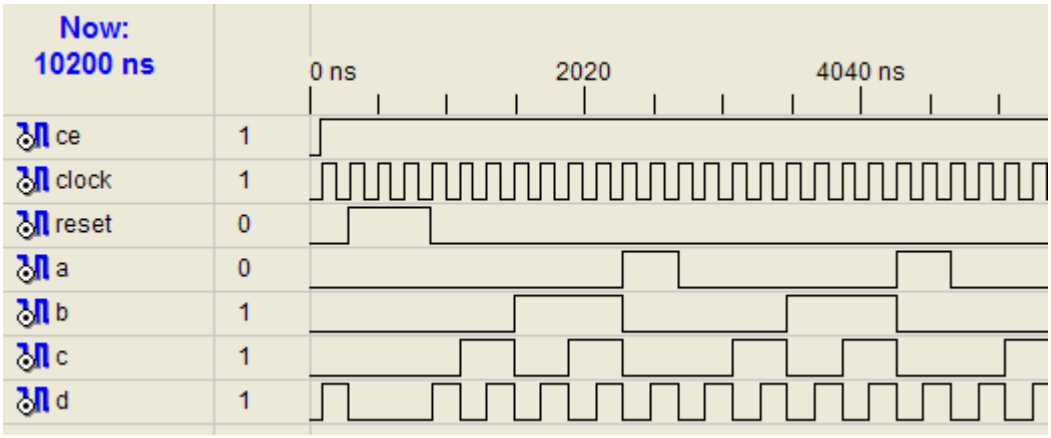

Figura 7. Simulación funcional del contador.

A	B	C	D	seg_a	seg_b	seg_c	seg_d	seg_e	seg_f	seg_g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

Tabla 1.Tabla de la verdad de un decodificador de 7 segmentos.

En nuestro caso, el display de 7 segmentos de la placa Spartan-3 tiene lógica negativa. Esto supone que cada led del display se enciende cuando el valor de su segmento es 0, y permanece apagado cuando vale 1. Esta consideración se tiene en cuenta en la tabla 1.

Crear un módulo VHDL llamado decodificador en la que estén implementadas las 7 funciones simplificadas usando mapas de Karnaugh. En la figura 8 se muestra la simulación de este módulo.

2.5. DISEÑO DEL SISTEMA GLOBAL

A continuación se colocan los 4 módulos creados en otro módulo jerárquicamente superior y que los engloba, tal y como se mostró en la figura 1. Para conectar cada bloque hay que emplear señales internas. Algunas de estas señales, *x* y *z* internas

concretamente, se asignan luego a las señales externas *x* y *z* que son señales de salidas definidas en la entidad del sistema total.

En la figura 9 se muestra la simulación funcional del sistema total.

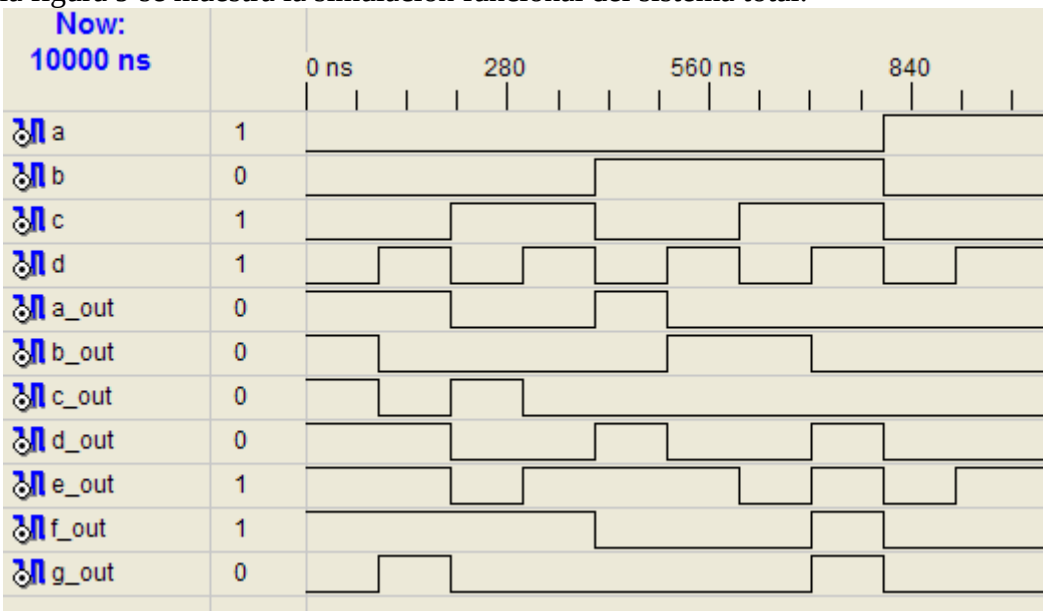

Figura 8. Simulación funcional del decodificador de 7 segmentos.

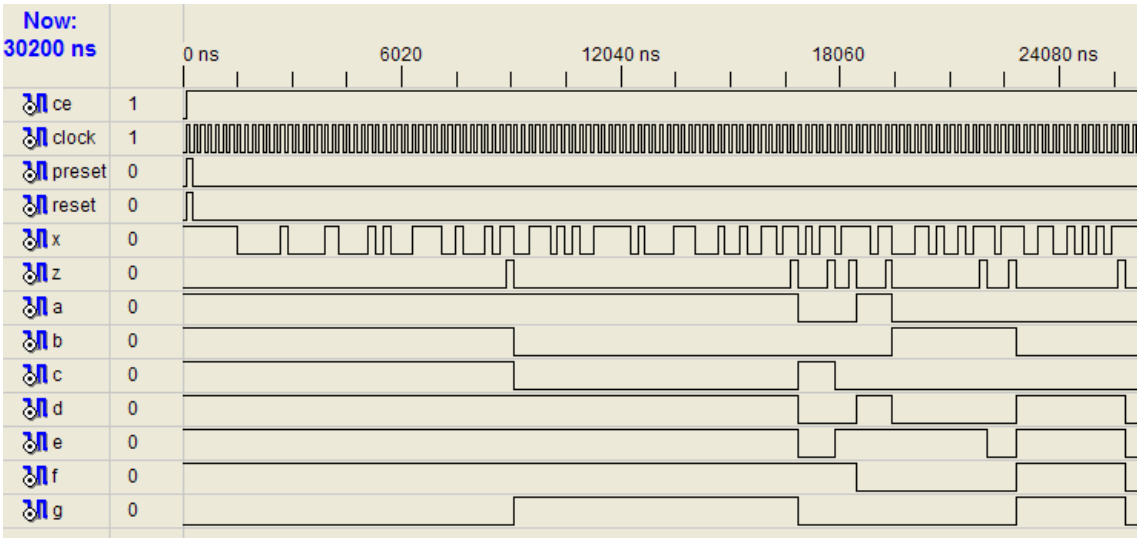

Figura 9. Simulación funcional del sistema total.

2.6. ASIGNACIÓN DE PINES

Para realizar la asignación de pines hay que crear un archivo de restricciones de usuario donde se incluye esta información. Como vimos en el tutorial, *Processes* → *User Constraints* → *Assign Package Pins*, asegurándonos de tener seleccionada en la ventana de *Sources* el *top* del diseño. Los pines de la FPGA deben estar conectados a los siguientes componentes de la placa:

Entradas:
Preset, CE, reset: interruptores
Clock: pulsador

Salidas:
X, Z: leds de la placa
a, b, c, d, e, f, g: display de 7 segmentos

2.7. IMPLEMENTACIÓN DEL DISEÑO

Una vez que se han diseñado y simulado correctamente todos y cada uno de los bloques, y se ha realizado la asignación de pines de la FPGA, debemos sintetizar nuestro diseño. Seleccionando el módulo de jerarquía superior, hacemos doble-click en *Processes* → *Generate Programming File* (cerrar el cuadro de diálogo que aparece). Esto hará que se ejecuten todos los procesos anteriores para generar el archivo de programación. Ya con la placa conectada, el último paso es programarla (*Generate Programming File* → *Configure Device (iMPACT)*).

Switches and LEDs

Four-Digit, Seven-Segment LED Display

The Spartan-3 Starter Kit board has a four-character, seven segment LED display controlled by FPGA user-I/O pins, as shown in Figure 3-1. Each digit shares eight common control signals to light individual LED segments. Each individual character has a separate anode control input.

The pin number for each FPGA pin connected to the LED display appears in parentheses. To light an individual signal, drive the individual segment control signal Low along with the associated anode control signal for the individual character. In Figure 3-1, for example, the left-most character displays the value '2'. The digital values driving the display in this example are shown in blue. The AN3 anode control signal is Low, enabling the control inputs for the left-most character. The segment control inputs, A through G and DP, drive the individual segments that comprise the character. A Low value lights the individual segment, a High turns off the segment. A Low on the A input signal, lights segment 'a' of the display. The anode controls for the remaining characters, AN[2:0] are all High, and these characters ignore the values presented on A through G and DP.

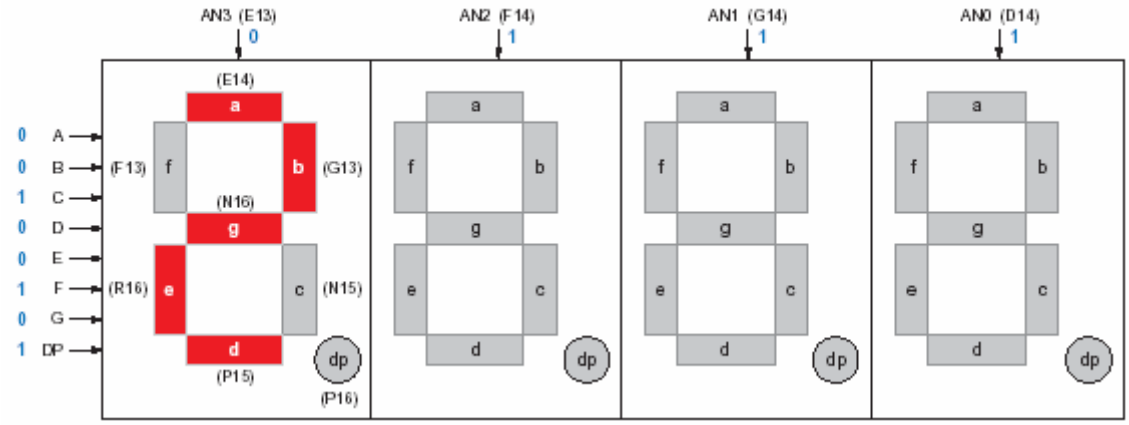

Figure 3-1: **Seven-Segment LED Digit Control**

Slide Switches

The Spartan-3 Starter Kit board has eight slide switches, indicated as 11 in Figure 1-2. The switches are located along the lower edge of the board, toward the right edge. The switches are labeled SW7 through SW0. Switch SW7 is the left-most switch, and SW0 is the rightmost switch. The switches connect to an associated FPGA pin, as shown in Table 4-1. A detailed schematic appears in Figure A-2.

Table 4-1: **Slider Switch Connections**

Switch	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA Pin	K13	K14	J13	J14	H13	H14	G12	F12

Push Button Switches

The Spartan-3 Starter Kit board has four momentary-contact push button switches, indicated as13 in Figure 1-2. These push buttons are located along the lower edge of the board, toward the right edge. The switches are labeled BTN3 through BTN0. Push button switch BTN3 is the left-most switch, BTN0 the right-most switch. The push button switches connect to an associated FPGA pin, as shown in Table 4-2. A detailed schematic appears in Figure A-2.

Table 4-2: **Push Button Switch Connections**

Push Button	BTN3	BTN2	BTN1	BTN0
FPGA Pin	L14	L13	M14	M13

LEDs

The Spartan-3 Starter Kit board has eight individual surface-mount LEDs located above the push button switches, indicated by 12 in Figure 1-2. The LEDs are labeled LED7 through LED0. LED7 is the left-most LED, LED0 the right-most LED. Table 4-3 shows the FPGA connections to the LEDs.

Table 4-3: **LED Connections to the Spartan-3 FPGA**

LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA Pin	P11	P12	N12	P13	N14	L12	P14	K12

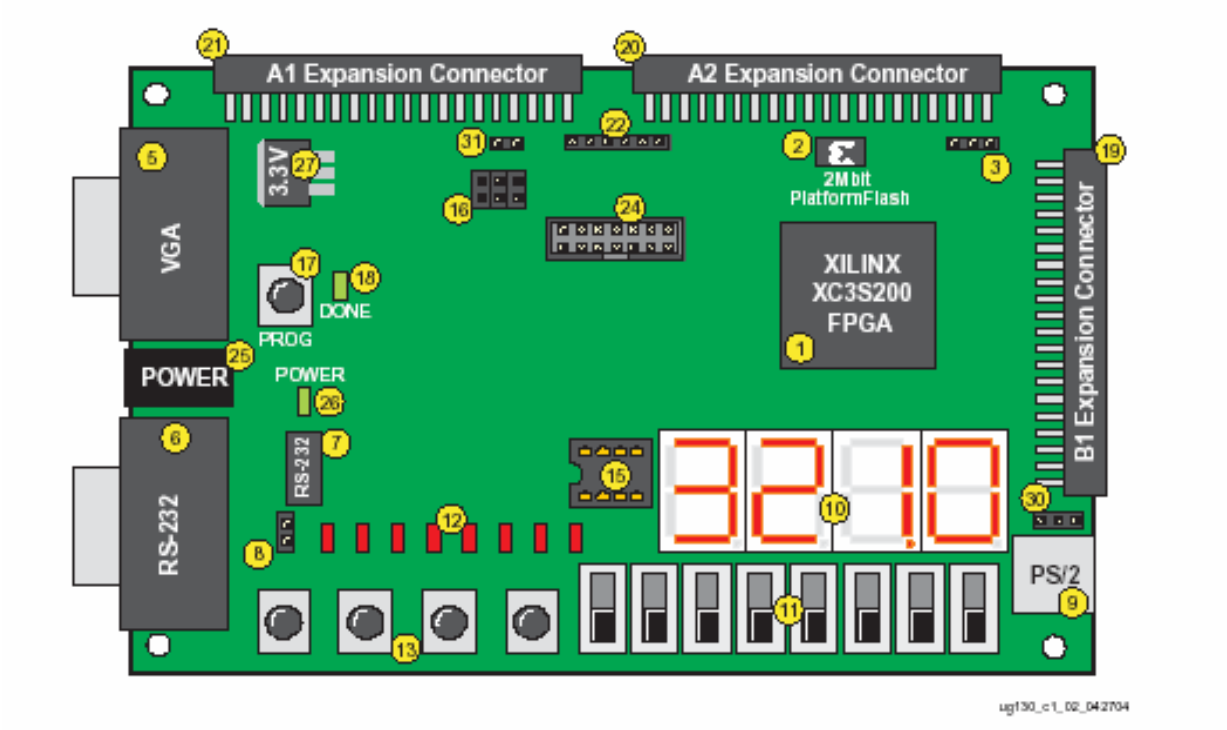

Figure 1-2: **Xilinx Spartan-3 Starter Kit Board (Top Side)**