Guess a number game

Universidad Latina de Panamá

Gabriela Santamaría 4-793-245

Diseño de Sistemas Digitales

***Resumen*—**

Este proyecto se basa en utilizar una FPGA, en este caso la Basys II programándola con un código en VHDL con el programa ISE Design Suite de Xilinx, el programa se basa en un juego para adivinar un numero entre el 00 y el 99 con ayuda de los botones, y leds sabrás si el numero que has ingresado es mayor, menor o igual al numero random que ha selecionado la tarjeta. El manual de uso corto esta adjunto a este documento.

Palabras Claves

I. **FPGA**
II. **Basys 2**
III. **Random number**

I. OBJETIVOS

Realizar un proyecto con FPGA funcional aplicando lenguaje VHDL

II. Dispositivos a utilizar

PC con Windows 7 en adelante
Tarjeta Basys 2
Cable USB

III. ¿QUÉ ES UNA FPGA?

Una matriz de puertas lógicas programable en campo o FPGA (del inglés field-programmable gate array), es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada en el momento, mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

Las FPGA se utilizan en aplicaciones similares a los ASIC sin embargo son más lentas, tienen un mayor consumo de energía y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGA tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Ciertos fabricantes cuentan con FPGA que sólo se pueden programar una vez, por lo que sus ventajas e inconvenientes se encuentran a medio camino entre los ASIC y las FPGA. Históricamente las FPGA surgen como una evolución de los conceptos desarrollados en las PAL y los CPL.

IV. HISTORIA

Las FPGA fueron inventadas en el año 1984 por Ross Freeman y Bernard Von Der Schmitt, cofundadores de Xilinx, y surgen como una evolución de los CPLD.

Tanto los CPLD como las FPGA contienen un gran número de elementos lógicos programables. Si medimos la densidad de los elementos lógicos programables en puertas lógicas equivalentes (número de puertas NAND equivalentes que podríamos programar en un dispositivo) podríamos decir que en un CPLD hallaríamos del orden de decenas de miles de puertas lógicas equivalentes y en una FPGA del orden de cientos de miles hasta millones de ellas.

Aparte de las diferencias en densidad entre ambos tipos de dispositivos, la diferencia fundamental entre las FPGA y los CPLD es su arquitectura. La arquitectura de los CPLD es más rígida y consiste en una o más sumas de productos programables cuyos resultados van a parar a un número reducido de biestables síncronos (también denominados flip-flops). La arquitectura de las FPGA, por otro lado, se basa en un gran número de pequeños bloques utilizados para reproducir sencillas operaciones lógicas, que cuentan a su vez con biestables síncronos. La enorme libertad disponible en la interconexión de dichos bloques confiere a las FPGA una gran flexibilidad.

Otra diferencia importante entre FPGA y CPLD es que en la mayoría de las FPGA se pueden encontrar funciones de alto nivel (como sumadores y multiplicadores) embebidas en la propia matriz de interconexiones, así como bloques de memoria.

V. Evolución del mercado de las FPGA

1985: primera FPGA comercial por Xilinx, XC2064
1987: 14 millones de dólares
1993: más de 385 millones de dólares
2005: 1 900 millones de dólares
2010: alrededor de 2 750 millones de dólares
2013: 5 400 millones de dólares
2020: alrededor de 9 800 millones de dólares

VI. Programación

En la FPGA no se realiza programación tal cual como se realiza en otros dispositivos como DSP, CPLD o microcontroladores. La FPGA tiene celdas que se configuran con una función específica ya sea como memoria (FLIP-FLOP tipo D), como multiplexor o con una función lógica tipo AND, OR, XOR. La labor del programador es describir el hardware que tendrá la FPGA. Por consiguiente, la tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos.
El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en una FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL o lenguajes de descripción de hardware. Los HDL más utilizados son:
VHDL
Verilog
ABEL
En un intento de reducir la complejidad y el tiempo de desarrollo en fases de prototipado rápido, y para validar un diseño en HDL, existen varias propuestas y niveles de abstracción del diseño. Los niveles de abstracción superior son los funcionales y los niveles de abstracción inferior son los de diseño al nivel de componentes hardware básicos. Entre otras, National Instruments LabVIEW FPGA propone un acercamiento de programación gráfica de alto nivel.

VII. BASYS 2

La placa Basys2 es una plataforma de diseño e implementación de circuitos que cualquiera puede usar para adquirir experiencia construyendo circuitos digitales reales. Construido alrededor de un Xilinx Spartan-3E Field Programmable Gate Array y un controlador USB Atmel AT90USB2, la placa Basys2 proporciona hardware completo y listo para usar adecuado para alojar circuitos que van desde dispositivos lógicos básicos hasta controladores complejos. Se incluye una gran colección de dispositivos de E / S integrados y todos los circuitos de soporte FPGA necesarios, por lo que se pueden crear

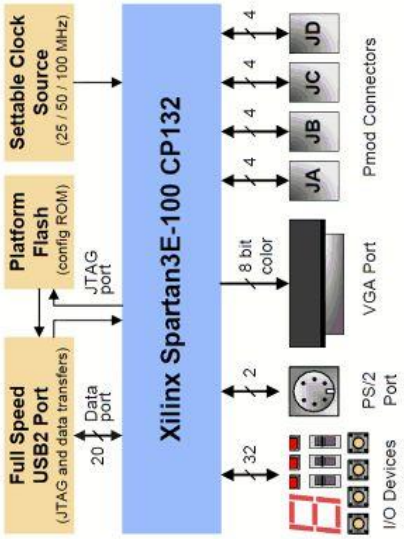
innumerables diseños sin la necesidad de ningún otro componente.

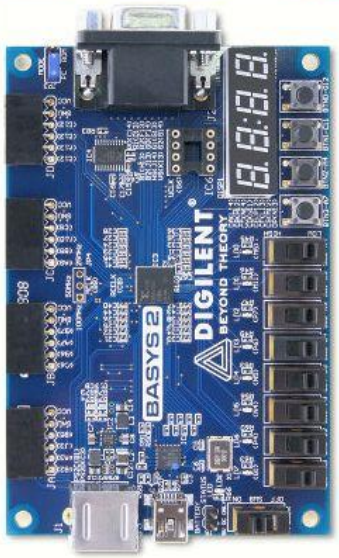
Cuatro conectores de expansión estándar permiten que los diseños crezcan más allá de la placa Basys2 utilizando placas de prueba, placas de circuito diseñadas por el usuario o Pmods (los Pmods son módulos de E / S digitales y analógicos económicos que ofrecen conversión A / D & D / A , controladores de motor, entradas de sensor , y muchas otras características).

Caracteristicas
Xilinx Spartan-3E FPGA, puertas de 100K
FPGA presenta multiplicadores de 18 bits
72 Kbits de RAM de bloque de puerto dual rápido
Operación de 300 MHz +
Requiere Adept 2.0 o posterior para funcionar
ROM Flash de la plataforma XCF02 que almacena configuraciones FPGA indefinidamente
Frecuencia del oscilador configurable por el usuario (25, 50 y 100 MHz), además del zócalo para un segundo oscilador
Tres reguladores de voltaje integrados (1.2V, 2.5V y 3.3V) que permiten el uso de suministros externos de 3.5V-5.5V
Se envía con un cable USB que proporciona alimentación y una interfaz de programación
8 LEDs, 4 botones, 8 interruptores deslizantes
Pantalla de cuatro dígitos y siete segmentos
Puerto PS / 2
Puerto VGA de 8 bits
Puerto USB2 de velocidad completa para configuración de FPGA y transferencias de datos (usando el software Adept 2.0 disponible como descarga gratuita)
Cuatro cabezales de 6 pines para E / S de usuario y conexión de placas de circuito de accesorios Digilent Pmod
Las señales en los conectores de 6 pines están protegidas contra daños por ESD y cortocircuitos, lo que garantiza una larga vida útil en cualquier entorno.
La placa Basys2 funciona a la perfección con todas las versiones de las herramientas Xilinx ISE, incluido el WebPack gratuito. Se envía con un cable USB que proporciona alimentación y una interfaz de programación, por lo que no se requieren otras fuentes de alimentación o cables de programación.

La placa Basys2 puede consumir energía y ser programada a través de su puerto USB2 incorporado. El software Adept para PC de Digilent, disponible gratuitamente, detecta automáticamente la placa Basys2, proporciona una interfaz de programación para FPGA y ROM Flash de plataforma , y permite transferencias de datos de usuario (consulte www.digilentinc.com para obtener más información).

La placa Basys2 está diseñada para funcionar con el software CAD ISE WebPack gratuito de Xilinx. WebPack se puede utilizar para definir circuitos utilizando esquemas o HDL, para simular y sintetizar circuitos y para crear archivos de programación. Webpack se puede descargar de forma gratuita desde www.xilinx.com/ise/ .





La placa Basys2 se entrega con una autocomprobación / demostración incorporada almacenada en su ROM que se puede utilizar para probar todas las características de la placa. Para ejecutar la prueba, configure el puente de modo (consulte a continuación) en ROM y aplique la alimentación de la placa. Si la prueba se borra de la ROM , se puede descargar y reinstalar en cualquier momento.

VIII. Código en VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.std_logic_arith.all;
use ieee.numeric_std.all;

entity muxdisplay_top is
port (
    clk50mhz:    in STD_LOGIC;
pin0 : in STD_LOGIC;
pin1 : in STD_LOGIC;
pin3 : in STD_LOGIC;
    display:    out STD_LOGIC_VECTOR(6 downto 0);
    cur_display:    out STD_LOGIC_VECTOR(3
downto 0);
leds : out STD_LOGIC_VECTOR(7 downto 0)
);
end muxdisplay_top;

architecture beh of muxdisplay_top is
    -- 50Mzh/100000=500Hz
    constant max_refresh_count: INTEGER :=
100000;
    signal refresh_count: INTEGER range 0 to
max_refresh_count;
    signal refresh_state: STD_LOGIC_VECTOR(1
downto 0) := (others => '0');
    signal display_sel: STD_LOGIC_VECTOR(3
downto 0) := (others => '0');
signal cuenta : integer := 0;
signal cuenta2 : integer := 0;
signal valorreal: integer :=0;
signal valormostrar : integer :=0;
signal valorc : integer := 0;
signal temporal: integer :=0;
begin
    cur_display <= display_sel;

    gen_clock: process(clk50mhz)
    begin
      if clk50mhz'event and clk50mhz='1' then
   -- contador 500Hz (para refresco del display)
    if refresh_count < max_refresh_count then
  refresh_count <= refresh_count + 1;
    else
refresh_state <= refresh_state + 1;
refresh_count <= 0;
if temporal=0 then
if valorc<99 then
valorc <= valorc +1;
else
valorc<=0;
end if;
end if;
        end if;
       end if;
    end process;

sumar: process(pin0)
begin
 if pin0'event and pin0='1' then
   if cuenta<9 then
    cuenta <= cuenta +1;
else
cuenta<=0;
end if;
 end if;
end process;

sumar2: process(pin1)
begin
 if pin1'event and pin1='1' then
   if cuenta2<9 then
    cuenta2<=cuenta2+1;
else
cuenta2<=0;
end if;
 end if;
end process;

verificador: process(pin3,cuenta,cuenta2,temporal)
begin
 if pin3'event and pin3='1' then
 valorreal<=cuenta2*10+cuenta;
   if valorreal = valorc then
  leds <= "01010101"; -- igual
temporal<=0;
 else
  temporal<=1;
  if valorreal > valorc then
   leds <= "11110000"; -- mayor
  else
leds <= "00001111"; -- menor
end if;
 end if;
 end if;
end process;

show_display: process(refresh_state,cuenta,cuenta2)
        begin
case refresh_state is
              when "00" =>
```

```
                display_sel <= "1110"; -- display 0
 valormostrar<=cuenta;
            when "01" =>
                display_sel <= "1101"; -- display 1
 valormostrar<=cuenta2;
when others =>
display_sel <= "1111";
valormostrar<=0;
          end case;


          -- mostrar digitos
          case valormostrar is
              when 0 =>
display <= "1000000"; -- 0
              when 1 =>
display <= "1111001"; -- 1
              when 2 =>
display <= "0100100"; -- 2
              when 3 =>
display <= "0110000"; -- 3
              when 4 =>
display <= "0011001"; -- 4
              when 5 =>
display <= "0010010"; -- 5
              when 6 =>
display <= "0000010"; -- 6
              when 7 =>
display <= "1111000"; -- 7
              when 8 =>
display <= "0000000"; -- 8
              when 9 =>
display <= "0010000"; -- 9
              when others =>
display <= "1111111";
     end case;

  --

end process;
end beh;
```

IX REFERENCIAS

(DIGILENT) Software download page
(SIDEOUS, 2014) Introduction to ISE DESIGN
Gonzáles Maxinez, David Jaime (2014). Grupo Editorial Patria, ed. *Programación de Sistemas Digitales con VHDL*. p. 330