Tarea 2 Arquitectura de Computadores

Juan Pablo León, 201473047-0

8 de Junio, 2016

1 HDLVerilog

1. Explique linea por linea lo que hace cada uno de los códigos.

   Para code1: declaramos la creación de un módulo code1 que tiene como entradas lógicas “clk”, “a”, “b”, “c” y salida “y”. Declaramos dentro del módulo la variable interna “x”. Luego tenemos un always el cual nos dejará pasar al “statement” solo en el borde positivo del “clk” (cuando sube a 1). Una vez dentro del always se asigna la suma de “a” y “b” a al valor de “x” y “x” módulo “c” a “y”. Finalmente se declara el fin del módulo.

   Code2 es prácticamente igual pero la asignación del “y” ocurre antes (en el código) que la de “x” dentro del always.

2. ¿Hacen lo mismo? si la respuesta es negativa, explique.

   Son casi iguales, pero, suponiendo que funcionasen, no. El code1 primero le asigna el valor a “x” y luego le asgina su valor a “y” utilizando el nuevo valor de “x”. En el caso de code2, al ser la asignación de “y” antes que la del “x” el valor de “y” será impredecible pues “x” todavía no tiene ningún valor.

3. ¿Qué sucede si se cambian los “=” por “ <= ”?

   Las asignaciones de “x” e “y” ocurren al mismo tiempo y ambos programas funcionan.

4. Use alguno de los módulos anteriores para programar una alarma que suene cada 1 [min]. Considere que el clock del sistema tiene una frecuencia de 1[Hz].

   (Para ”desactivar la alarma” por otro minuto se utiliza el reset)

```
module alarma(  input logic clk,
                input logic reset,
                output logic alarm);
        assign alarm = 0;
        reg [5:0] counter;

        always @(posedge clock) begin
                if (reset) counter <= 6'd0, alarm <= 0;
                else begin
                        counter <= code1( clk, 6'd1, counter, 6'd60);
                        if (counter == 6'd0) alarm <= 1;
                end
        end
endmodule
```

2 Representación decimal

1. sean $\alpha = 0 + 2 = 2, \beta = (0\%4) + 3 = 3$, calcule el número con la ecuación dada y exprese en punto flotante.

   Nuestro número es, aproximadamente, 12.529.065,25. Lo consideraremos como $1.252.906.25 * 10^3$

   Tenemos que la parte entera, en binario, se escribe 1100 0000 0010 0011 1100 0001 mientras que la parte decimal es 01, entonces nuestro número en binario es 1100 000 0010 0011 1100 0001.01 pero ahora debemos normalizar la mantissa, nos queda: 1.1000 0000 1000 1111 0000 0101 $*2^{22}$.

   Ahora calculamos el exponente: $127 + 23 = 150_{10} = 10010110_2$. Omitimos el primer 1 de la mantisa pues solo nos interesa la parte decimal, notamos que nos pasamos en un bit por lo que nuestra fracción queda: 1000 0000 1000 1111 0000 011 pues nuestro último bit era un 1. Sabemos que el número es positivo por lo que un 0 va al inicio. Juntamos todo y nos queda que nuestro número en punto flotante es:

   $$0\ 1001\ 0110\ 1000\ 0000\ 1000\ 1111\ 0000\ 011$$

   Donde el primer bit nos indica el signo, los siguientes 8 el exponente y los 23 restantes la fracción.

2. Transforme 1100 1011 0111 1011 0101 0110 1011 0111 a decimal considerando que este número está representado en:

   A) punto fijo con 16 bits para la parte decimal (en complemento 2)

   Primero le restamos 1 al bit menos significativo y luego damos vuelta los 0s y 1s del número, nos queda: 0011 0100 1000 0100 1010 1001 0100 1001 luego consideramos los primeros 16 bits como la parte entera y el resto la parte fraccionaria, nos queda:

   $$0011\ 0100\ 1000\ 0100_2 = 13.444$$
   $$1010\ 1001\ 0100\ 1001_2 = 661270141$$

   Finalmente sabemos que el número era originalmente negativo, por lo que unimos todo y nos da el número buscado: -13.444,661270141.

   B) Punto flotante de precisión simple. Separamos el número siguiendo las reglas del punto flotante en presición simple:

   Signo: 1 (-)

   Exponente: 1001 0110 (154) $\Rightarrow$ 27

   Mantissa: 1111 1011 0101 0110 1011 0111

   Juntando todo tenemos que el número es: $1.11110110101011010110111 * 2^{27} = -1.9635838270187378 * 2^{27} = -2.6354776 * 10^8$

3. Calcule el mayor número y menor número positivo representable en punto flotante con precisión simple en su forma normal.

   Sabemos que, en punto flotante, cuando el exponente está lleno de 1s, 1111 1111, se trata de menos infinito, más infinito o not a number. Es por esto que el exponente a considerar será 1111 1110. Para obtener el número más grande llenaremos la mantissa con 1s y luego el signo dependerá del primer bit. Tenemos entonces que:

   Mayor número: $0\ 1111\ 1110\ 111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = 1,11111111111111111111 * 2^{127} = (2 - 2^{-23}) * 2^{127} = 3,4028235 * 10^{38}$

   Menor número: $1\ 1111\ 1110\ 111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -3,4028235 * 10^{38}$