Listado de ejercicios para aprobación

June 19, 2024

Nota 1: Todos los ejercicios deben ser resueltos en Verilog con su correspondiente testbench.

Nota 2: Durante la defensa de los ejercicios, los docentes pueden preguntar temas generales y/o específicos de cualquier tema del programa del curso, no sólo sobre los ejercicios que aquí se presentan.

Ejercicio 1 - De un texto del idioma castellano de máximo 1.000 caracteres, se quiere reconocer cuántas veces aparece la palabra "casa".

- Diseñe el AFD (cuántos registros de 1 bit se necesitan, y halle la función de traducción f: N $\rightarrow$ N que calcula el traductor, halle las funciones w y d.
- TIP: se considerará que solo pueden aparecer los siguientes símbolos en el texto: a, b, c, d, e, f, g, h, i, j, k, l, m, n, ñ, o, p, q, r, s, t, u, v, w, x, y, z, ' ' (espacio en blanco para separar). Es decir no hay signos de puntuación ni upper cases.

Ejercicio 2 - Implementar en Verilog el ejercicio 1. Establezca una codificación binaria para representar los caracteres. Simule con algún texto de 1000 caracteres incluído el espacio en blanco.

Ejercicio 3 - Sea x un entero expresado en 2C con N_x bits y sea y un entero expresado en 2C con N_y bits. Describir en Verilog un circuito que obtenga $m = x \times y$ expresado en 2C con $N_m = N_x + N_y$ bits tal que opere secuencialmente con codificación CSD. Verifique que opere correctamente para todos los casos posibles de entrada.
TIP: Se puede utilizar la siguiente codificación

x_i^{2C}	x_{i1}^{CSD}	x_{i0}^{CSD}
0	0	0
1	1	0
$\overline{1}$	1	1

De esta forma bit x_{i1}^{CSD} significa que o bien no hay que sumar nada o bien hay que restar o sumar. Bit x_{i0}^{CSD} significa que si es 0 se suma, si es 1 se resta (se puede usar para invertir el operando y y como entrada del carry in del sumador.

Ejercicio 4 - Implementar en Verilog un multiplicador serie-paralelo ambos operandos en 2C.

Ejercicio 5 - Implementar un divisor basado en binary search donde el divisor y el dividendo son ambos de una cantidad genérica de bits.

Ejercicio 6 - Implementar con un multiplicador sequencial de Booth, el filtro FIR que posee los siguientes coeficientes. El filtro es simétrico, $L = 16$. Los bits de entrada y salida son $B_x = B_y = 16$ bits.

k	$h[k]$
0 / 15	10
1 / 14	21
2 / 13	32
3 / 12	43
4 / 11	54
5 / 10	65
6 / 9	76
7 / 8	87

Ejercicio 7 - Implementar un CIC de orden genérico N, un ancho de palabra de entrada genérico BX y una tasa de decimacción genérica $R = 2^K$.

Ejercicio 8 - Implementar un CORDIC desenrrollado de 16 iteraciones. Mediante un parámetro PIPELINE se seleccionará si se instancias registros entre etapas. Cuando PIPELINE=1, se agregarán dichos registros, cuando PIPELINE=0, será combinacional puro. Agregar precordic para procesar todo el círculo. Finalmente, poseerá una entrada rot_0_vec_1 para seleccionar entre los modos rotación y vectorización. Tip: observar que en el caso pipeline, la entrada rot_0_vec_1 debe entrar a un shift register para que no se desincronize respecto de los datos, de forma tal que cada etapa CORDIC consuma el valor de rot_0_vec_1 desde el registro correspondiente del shift register.