ARQUITECTURA DE COMPUTADORAS I

Guía de Ejercicios: Lógica Digital con HDL (Verilog)

Ejercicio 1.

Diseñe un módulo en Verilog que implemente un sumador completo de un bit con entradas *a*,*b* y *cin*(carry in), y salidas *sum* y *cout* (carry out). Describir y simular con Icarus Verilog + GTKWave.

Ejercicio 2.

Diseña un registro de desplazamiento de 4 bits con entrada serial *serial_ in*, señal de reloj *clk*, y salida en paralelo *q[3:0]*. En cada flanco positivo de *clk* debe desplazarse el dato serial hacia la derecha.

Ejercicio 3.

Realiza un módulo que implemente un multiplexor 4 a 1 con entradas de datos de 8 bits (*data0* a *data3*), señal de selección *sel* de 2 bits y salida *y* de 8 bits, usando descripción en nivel comportamiento.

Ejercicio 4.

Diseña un flip-flop tipo D con reloj síncrono y señal de reset asíncrono activo alto. El flip-flop debe almacenar el valor de la entrada D en el flanco positivo del reloj.

Ejercicio 5.

Diseña un sistema digital que integre un contador binario de 4 bits y un módulo convertidor binario a display de 7 segmentos para mostrar el conteo. El sistema debe contar de 0 a 15 con reloj síncrono y reset asíncrono. El display debe actualizarse con el valor actual del contador. Simula la funcionalidad completa mostrando señales internas y la salida para el display en GTKWave.

Ejercicio 6.

Diseña una ALU de 4 bits que realice las operaciones básicas: suma, resta, AND, OR entre dos operandos de 4 bits (a y b), seleccionando la operación mediante una señal de 2 bits opcode. Salida result y flag de cero zero.

Ejercicio 7.

Diseña un mini procesador simple con módulos integrados:

- Unidad Aritmético Lógica (ALU) de 4 bits (sumar, restar, AND, OR).

- Banco de registros con 4 registros de 4 bits.
- Unidad de control que tome instrucciones simples codificadas (ej. 2 bits opcode + registros).
- Ciclo básico de fetch-decode-execute simulado.

Proporciona un banco de pruebas que simule una secuencia simple de instrucciones (ejemplo: cargar datos, operar y almacenar).

Ejercicio 8.

Diseña un módulo Verilog para una memoria RAM simple y síncrona con capacidad parametrizable. La memoria debe tener:

- Entradas: reloj (*clk*), habilitación escritura (*we*), dirección (*addr*), dato de entrada (*data_ in*).
- Salida: dato almacenado en la dirección indicada (*data_ out*).
- Debe funcionar escritura en flanco positivo del reloj solo si *we* está activo.

Ejercicio 9.

Diseña una memoria ROM de 16 palabras de 8 bits cada una, que entrega el dato en la dirección addr. La memoria debe inicializarse con valores predefinidos en el bloque initial.

Ejercicio 10.

Diseña un módulo Verilog llamado io_decoder que reciba una dirección de 8 bits addr, una señal de lectura (rd) y escritura (wr), y active señales de chip select (cs) para 4 dispositivos de E/S mapeados en estas direcciones:

- Dispositivo 0: direcciones 0x00 a 0x0F
- Dispositivo 1: direcciones 0x10 a 0x1F
- Dispositivo 2: direcciones 0x20 a 0x2F
- Dispositivo 3: direcciones 0x30 a 0x3F

El módulo debe generar señales de selección cs0, cs1, cs2 y cs3 activas altas sólo cuando la señal rd o wr esté activa y la dirección esté en el rango correspondiente.

Luego, crea un ejemplo de placa *lab_ board* que integre este decodificador con:

- Módulos *dummy* simulados para cada dispositivo (simplemente un registro que puede leerse/escribirse).
- Señales de reloj y reset.
- Puertos para interactuar con las señales de E/S.