Circuitos Digitales I

Proyecto Verilog

1 Indicaciones Generales

1. El proyecto Verilog requiere como entregables dos partes:
   (a) El informe escrito.
   (b) Los códigos fuente.
2. Son requisitos de admisibilidad del proyecto:
   (a) Que se presente el informe escrito completo.
   (b) Que se presenten todos los códigos fuente requeridos debidamente completados y que compilen apropiadamente en iverilog.
3. Si un proyecto no satisface los requisitos de admisibilidad el mismo no se revisará y se asignará una nota de cero.
4. El informe debe ser presentado de una manera ordenada y clara. Puede hacerse en manuscrito o con un procesador de texto. En cualquier caso se debe entregar en formato pdf.
5. Los códigos fuente serán compilados en iverilog y la persona estudiante debe asegurarse que dichos códigos compilen adecuadamente en este compilador HDL. Todo los códigos deben estar debidamente documentados, con los respectivos comentarios en las declaraciones de las estructuras de datos y en los encabezados, de tal manera que sea fácil de leer, entender y reutilizar.

2 Ejercicio 0

Realice diferentes pruebas cambiando los estímulos generados en los módulos de los multiplexores estudiados en clase. Muestre los diagramas de tiempo de ambos multiplexores funcionando al mismo tiempo y comente las diferencias entre cada uno y como se relacionan a la manera que se implementaron. El diagrama de tiempo debe aparecer en el reporte y puede escoger la prueba que más le llamó la atención.

3 Ejercicio 1

Implemente utilizando Verilog los siguientes módulos de manera que siga una definición estructural de la lógica que se necesita (conectar compuertas and, or, not, etc).

- Modulo para realizar la operación AND de 4 bits.
- Modulo para realizar la operación OR de 4 bits.

- Modulo para realizar la operación XOR de 4 bits.
- Modulo para realizar una suma de 4 bits con acarreo final.

Para realizar el modulo de suma puede utilizar como referencia la siguiente tabla de verdad para un full adder de 1 bit y desarrollar la lógica utilizando mapas de Karnaugh:

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 1: Tabla de verdad para full-adder de 1 bit.

Tome en cuenta que para crear el modulo de 4 bits es necesario conectar los modulos de 1 bit en cascada de la siguiente forma:

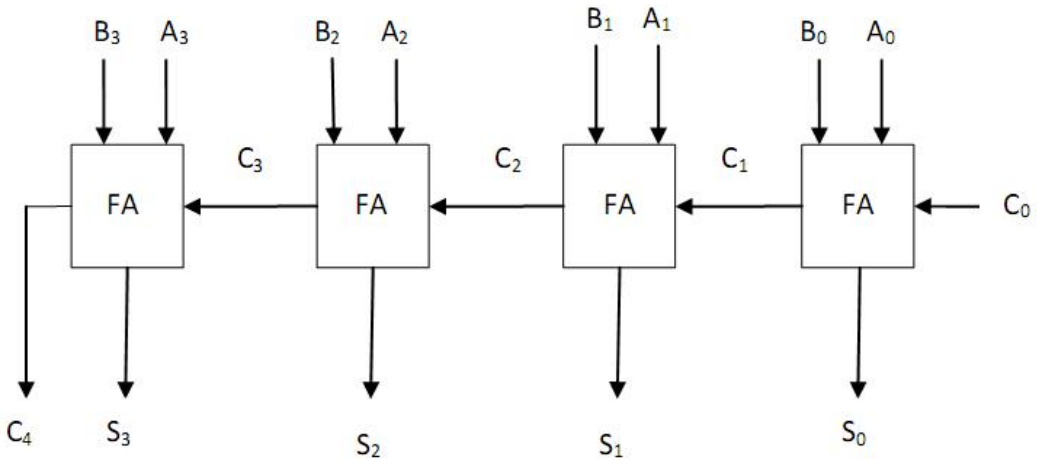

Figure 2: Sumador de 4 bits.

Ademas de los modulos anteriormente mencionados implemente un MUX 4x1 con entradas y salida de 4 bits utilizando una definicion por comportamiento de la lógica (conductual).

Finalmente realice un modulo de testbench para probar que toda la logica implementada funciona adecuadamente. Muestre los resultados en el informe.

4 Ejercicio 2

Conecte todos los módulos del ejercicio 1 de manera que genere una ALU(Unidad Lógico Aritmética) para entradas de 4 bits, el diagrama de como se debe hacer la conexión se muestra a continuación:

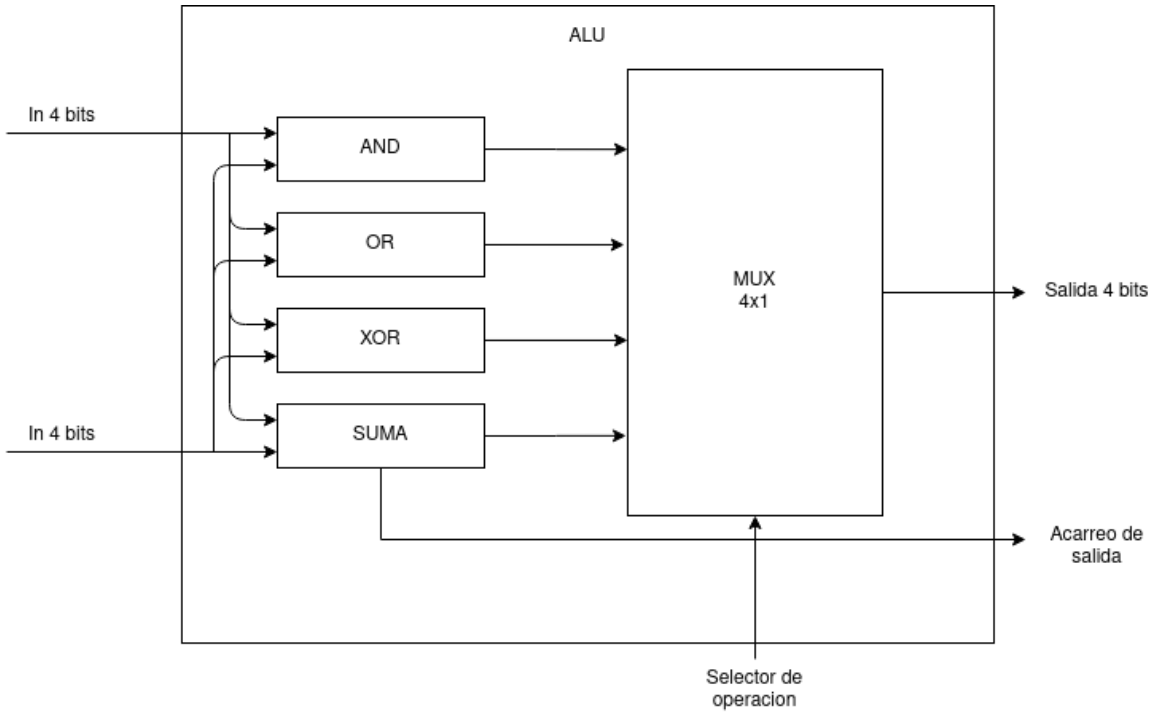

Figure 3: ALU de 4 bits.

Finalmente realice un modulo de testbench para probar que toda la lógica implementada funciona adecuadamente. Muestre los resultados en el informe.

5 Ejercicio Extra

Agregue la funcionalidad de realizar restas a la ALU así como realizar la operación lógica NOT. Piense en los cambio necesarios que se debe realizar a las conexiones para agregar estas dos operaciones. Nota: Recuerde que la resta en números binarios se realiza utilizando el complemento a 2 de un numero y una suma(A+-B) donde -B es el complemento a 2 de B.

Realice un modulo de testbench para probar que toda la lógica implementada funciona adecuadamente. Muestre los resultados en el informe.