



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Informe Tarea

Lenguajes de Descripción de Hardware

Grupo 22

Martín Arancibia 201973517-9
Miguel Soto 201973623-K

21 de noviembre de 2022

Índice

1. Resumen	1
2. Operaciones	1
3. Desarrollo	2
4. Casos y Pruebas	5
5. Conclusión	6

1. Resumen

La intención de esta tarea es realizar una unidad lógica capaz de operar con dos entradas, dando una salida y 'flags' que permitan determinar la naturaleza del resultado dada la operación en cuestión.

Esto se logró a través del lenguaje SystemVerilog, el cuál maneja el puente entre los circuitos que hemos visto en el curso, y un lenguaje de programación más tradicional (es importante recalcar que verilog no es un lenguaje de programación. sino más bien de descripción de hardware).

2. Operaciones

Lo fundamental de la tarea es desarrollar un ALU (Arithmetic Logic Unit) que pueda cumplir una serie de operaciones binarias, las cuales hemos tenido que crear con el sistema descriptivo de Hardware SystemVerilog. Junto con esto, deben desarrollarse un número de 'Flags', las cuáles sirven para indicar la naturaleza del resultado de la operación realizada.

Las operaciones junto con su unidad de control correspondiente están indicados de la siguiente forma en el enunciado:

Tabla de operaciones

Op-code	Operación	Resultado	Descripción
000	Suma simple	Entero	Suma A y B en Complemento-2
001	Resta simple	Entero	Resta A y B en Complemento-2
010	Suma positiva	Natural	Suma A y B en Magnitud
011	Resta positiva	Natural	Resta A y B en Magnitud
100	Rotación izquierda	Entero	Rota A a la izquierda B veces
101	Rotación derecha	Entero	Rota A a la derecha B veces
110	Duplicación	Natural	Duplica A una cantidad B de veces
111	División binaria	Natural	Divide A por 2 una cantidad B de veces

Sumado a esto, las 'Flags' que pueden producirse tras realizar una operación:

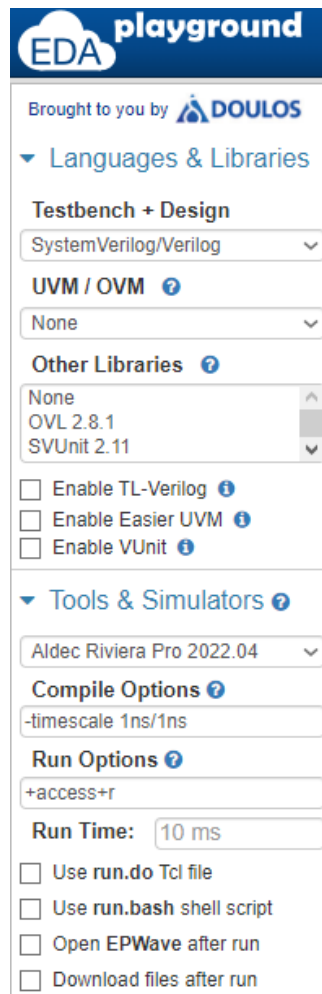
Flags de la ALU

Flag	Condición	Descripción
N	Negative	El valor de salida es negativo
Z	Zero	El valor de salida es cero
C	Carry	La operación produce un carry de salida
V	Overflow	La operación produce un overflow numérico
G	Greater	El valor de A es mayor que el valor de B
Q	Equal	El valor de A es igual al de B
O	Odd	El valor de salida es impar
P	Pairs	El valor de salida tiene la misma cantidad de 1s y 0s

Éstas 'Flags' aparecerán como un arreglo de 8 bits en donde el estado de dicha flags se determina en base a la posición de dicha variable y si su posición corresponde a un 1 (la flags se cumple) o un 0 (la flags no se cumple). Para lograr esto, se utilizó la posición de cada flag como aparece ordenada en la tabla del enunciado, por ejemplo, el estado 'Negativo' será la primera posición, el estado 'Zero' la segunda, y así consecutivamente.

3. Desarrollo

Para poder desarrollar las distintas operaciones se trabajará en la página entregada por los ayudantes, <https://www.edaplayground.com/home>. En ella, se ocuparán las siguientes opciones que nos solicitan de antemano.



The image shows the configuration sidebar of the EDA Playground website. At the top is the 'EDA playground' logo. Below it, it says 'Brought to you by DOULOS'. The main section is titled 'Languages & Libraries'. Under this, there is a 'Testbench + Design' section with a dropdown menu set to 'SystemVerilog/Verilog'. Below that is a 'UVM / OVM' section with a dropdown menu set to 'None'. Then there is an 'Other Libraries' section with a list containing 'None', 'OVL 2.8.1', and 'SVUnit 2.11'. Below the libraries are three checkboxes: 'Enable TL-Verilog', 'Enable Easier UVM', and 'Enable VUnit', all of which are currently unchecked. The next section is 'Tools & Simulators' with a dropdown menu set to 'Aldec Riviera Pro 2022.04'. Below this is a 'Compile Options' section with a text input field containing '-timescale 1ns/1ns'. Then there is a 'Run Options' section with a text input field containing '+access+r'. Below that is a 'Run Time' section with a text input field set to '10 ms'. At the bottom are four checkboxes: 'Use run.do Tcl file', 'Use run.bash shell script', 'Open EPWave after run', and 'Download files after run', all of which are currently unchecked.

Luego, para lograr las operaciones, se trabajó a partir del código entregado en aula. La ALU funciona de forma asíncrona y devuelve un resultado a partir de la unidad de control denominada 'opcode' y los dos inputs 'A' y 'B'.

La suma simple y resta simple se logran a través de la operación entre números en forma 'complemento de dos' como las vimos en clase. En particular, la resta se logra invirtiendo el número de 'B', sumándole un uno y luego sumándolo al otro input 'A', mientras que la suma se logra haciendo ésta operación invertida.

La suma y resta positiva se logran usando las operaciones por defecto en el lenguaje.

Las rotaciones se logran a través de un 'shifteo' de 'A' en 'B' veces, esto está dentro de un 'OR' ya que si la salida es cero, entonces se realiza un 'shifteo' por la cantidad de espacios restantes, así logrando que los bits roten de forma natural.

Finalmente, la duplicación y división se logran haciendo un 'shifteo' simple hacia la izquierda y derecha respectivamente.

```

// Code your design here

// La ALU definida como sale en el enunciado
// con los mismos nombres para los parametros

module ALU(input logic [7:0] A,B,
           input logic [2:0] opcode ,
           output logic [7:0] salida ,
           output logic [7:0] flags);

// El valor de la unidad de control maneja las
// operaciones a realizar

always @(*) begin

    case(opcode)

        // Suma Simple
        // Pasa a B a Complemento de Dos y
        // luego lo resta a A
        3'b000: begin
            salida = A - (~B+1);
        end

        // Resta Simple
        // Pasa a B a Complemento de Dos y
        // luego lo suma a A
        3'b001: begin
            salida = A + (~B+1);
        end

        // Suma Positiva
        // Suma los dos numeros en forma
        // Signo Magnitud
        3'b010: begin
            salida = A + B;
        end

        // Resta Positiva
        // Resta los dos numeros en forma
        // Signo Magnitud
        3'b011: begin
            salida = A - B;
        end

        // Rotacion Izquierda
        // Desplaza B veces a A a la izquierda ,
        // si se sale del arreglo aparece por
        // la derecha
        3'b100: begin
            salida = (A << B) | (A >> (7-B));
        end

        // Rotacion Derecha
        // Desplaza B veces a A a la derecha ,
        // si se sale del arreglo aparece por
        // la izquierda
        3'b101: begin

```

```

        salida = (A >> B) | (A << (7-B));
    end

    // Duplicacion
    // Desplaza B veces a A a la izquierda
    3'b110: begin
        salida = A << B;
    end

    // Division
    // Desplaza B veces a A a la derecha
    3'b111: begin
        salida = A >> B;
    end

    // Si el caso es invalido retorna cero
    default: salida = 8'b0000_0000;
    endcase

    assign flags = {( salida >= 255) ? 1'b1 : 1'b0,
        (salida == 0) ? 1'b1 : 1'b0,
        1'b0,
        ( (A + B > 255) | (A - B < 255) ) ? 1'b1 : 1'b0,
        (A > B) ? 1'b1 : 1'b0,
        (A == B) ? 1'b1 : 1'b0,
        (salida % 2) ? 1'b1 : 1'b0,
        (salida == 15 | salida == 51 | salida == 60 | salida == 85 | salida ==
    end
endmodule

```

Finalmente, las 'flags' se designaron la concatenación de múltiples operaciones lógicas.

El estado negativo se determina si es que el bit más significativo de la salida es igual a uno.

El estado cero se determina si es que la salida es cero.

El estado carry queda a ejercicio para el lector.

El estado overflow se determina si es que la salida da un output que no cabe en los 8 bits.

El estado greater se determina si es que A es mayor a B.

El estado equal se determina si es que A es igual a B.

El estado odd se determina si es que el bit menos significativo es igual a 1.

El estado pairs toma todos los casos posibles y evalúa si es que el output calza con alguno de ellos.

4. Casos y Pruebas

Hecha ya la parte de código e implementación de las operaciones, hacemos el código que nos permitirá probar el ALU, quedándonos lo siguiente:

```
// Code your testbench here
// or browse Examples
module test ();

    logic [7:0] A,B,salida , flags ;
    logic [2:0] opcode;

    // Las funciones implementadas dentro del ALU
    // El número de la operación sigue aquel indicado
    // en el enunciado de la Tarea

    // Instancia al ALU
    ALU alu(A,B,opcode , salida , flags );

    initial begin
        A = 8'b0000_1111;
        B = 8'b1000_0000;
        opcode = 3'b100; #10;

        // Imprime el resultado por pantalla
        $display("\nTarea_3:_SystemVerilog\n");
        $display("__Entrada_A:_%b",A);
        $display("__Entrada_B:_%b",B);
        $display("__Operacion:_%b",opcode);
        $display("__Resultado:_%b",salida);
        $display("__Flags:_%b",flags);
        $display("\n");
    end
endmodule
```

Las pruebas dieron los resultados a continuación (copiados directamente de EDA Playground):

```
# KERNEL:      Entrada A: 00000001
# KERNEL:      Entrada B: 11111111
# KERNEL:      Operacion: 000
# KERNEL:      Resultado: 00000000
# KERNEL:      Flags: 01010000

# KERNEL:      Entrada A: 00000001
# KERNEL:      Entrada B: 10000000
# KERNEL:      Operacion: 000
# KERNEL:      Resultado: 10000001
# KERNEL:      Flags: 00000010

# KERNEL:      Entrada A: 00001111
# KERNEL:      Entrada B: 00000011
# KERNEL:      Operacion: 100
# KERNEL:      Resultado: 01111000
# KERNEL:      Flags: 00011000

# KERNEL:      Entrada A: 00001111
# KERNEL:      Entrada B: 00000011
# KERNEL:      Operacion: 111
# KERNEL:      Resultado: 00000001
```

KERNEL: Flags: 00011010

5. Conclusión

Tras la elaboración de este informe, podemos conluir con el aprendizaje del lenguaje de descripción de hardware, el cuál nos permitió aterrizar funciones matemáticas y operaciones algebraicas a un nivel mucho más bajo.

A través de este puente también se hizo más aparente la relación entre la computación y el área eléctrica, dándonos a entender que los sistemas que manejan las computadoras del día a día realizan la tarea de traducir las instrucciones que se le solicitan a un ritmo impresionante.

Finalmente, podemos aseverar que el práctico cumplió su objetivo de dar más comprensión de la materia de forma tangible.