Arquitectura de computadoras
Trabajo Práctico N°1
ALU

Alumnos:

- Altuna, Julian
- Arce Giacobbe, Alejandro

Fecha: 11/9/2019

Introduccion

En este trabajo se busca implementar una Unidad Aritmética Lógica en Verilog, la cual debe soportar 8 operaciones distintas.
En primer lugar se realizará la simulación de la misma con edaplayground o ISE y luego se pasará a la placa de desarrollo Basys II.

Desarrollo

ALU

El módulo de ALU cuenta con dos parámetros, tres entradas y una salida. Los parámetros definen la longitud de los argumentos y el codigo de operacion. Las entradas son los datos A, B y el código de operación. La salida es el resultado obtenido luego de realizar la operación deseada.
Para calcular el resultado, contamos con un registro que es asignado dentro de un always. En el mismo determinamos la operación a realizar utilizando un case sobre el codigo de operacion.

```
always @ (*)
  begin
    case (i_op_code)
      ADD : res <= i_dato_a + i_dato_b;
      SUB : res <= i_dato_a - i_dato_b;
      AND : res <= i_dato_a & i_dato_b;
      OR  : res <= i_dato_a | i_dato_b;
      XOR : res <= i_dato_a ^ i_dato_b;
      SRA : res <= i_dato_a >>> i_dato_b ;
      SRL : res <= i_dato_a >> i_dato_b;
      NOR : res <= ~(i_dato_a|i_dato_b);
    default : res <= i_dato_a;
    endcase
  end
```

Testbench ALU

El testbench es usado para simular el módulo creado anteriormente. Lo que se hace es cambiar el código de operación cada dos ciclos de clock, generando números aleatorios como datos.

Una vez que la ALU calcula el resultado, el testbench verifica que el resultado obtenido sea el mismo que aplicar la misma operación sobre los datos enviados al módulo. De lo contrario, imprime un error en pantalla.

Ejemplo de la verificación de la suma:

```
#2//test suma
i_op_code_01 = 6'b100000;
i_dato_a_01 = $random;
i_dato_b_01 = $random;
#2
if(i_dato_a_01 + i_dato_b_01 != o_resultado_01)
  $display(" *ADD ERROR* ");
```

Implementación en la placa

Para la implementación en la placa debemos contar con un mecanismo para ingresar los datos. Se utilizan los 8 switches con los que cuenta la placa Basys2 para tomar datos A, B o código de operación. Luego, con los pulsadores se define a cuál de los tres parámetros se asigna el dato.
Contamos con un módulo que crea una instancia de la ALU. En este módulo almacenamos los valores que debe recibir el módulo en registros. La asignación se realiza en el siguiente always:

```
always @ (posedge i_clock)
begin
   casez (buttons)
       3'b1?? : i_dato_a_01 <= switch;
       3'b?1? : i_dato_b_01 <= switch;
       3'b??1 : i_op_01 <= switch;
   endcase
end
```

Como se ve en la figura, en este always la lista de sensibilización contiene un always. Esto se hace para que se sintetice un flip-flop en lugar de un latch.

Finalmente, el resultado que calcula el módulo de la ALU se muestra, en binario natural, en los LED de las placa.