8 de Septiembre de 2020

Tecnológico de Monterrey

La Unidad Aritmético-Lógica

Laboratorio de arquitectura de Computadoras

ITESM Campus Monterrey

Profesor
Diego Fernando Valencia Martínez

Miguel Morales de la Vega
Javier Mondragón Martín del Campo

A00821541
A01365137

Introducción

En esta práctica se implementa un módulo ALU (Aritmetic Logic Unit por sus siglas en inglés) que componen los módulos básicos de un circuito digital con operaciones aritméticas como la suma, resta, multiplicación y corrimiento de bits. Esta se compondrá de entradas y salidas de 32 bits con una entrada de control para la selección del modo de operación requerido y una salida sensible a la respuesta del resultado si es igual a ‘0’.

Los más complejos circuitos electrónicos modernos llevan consigo implementados físicamente este tipo de componentes para agilizar las operaciones y puede llevar uno o más ALU por núcleo y a sí mismo uno o más núcleos.

Procedimiento

Módulo ALU.
La creación del módulo de ALU contiene varias funciones/operaciones lógicas en su interior. Las operaciones del módulo corresponden a un formato de 32 bits, contiene 3 entradas de las cuales una es un selector de operaciones y 2 salidas una como salida de 32 bits (resultado de las operaciones lógicas) y la otra salida corresponde a una bandera que indica si el resultado es cero.

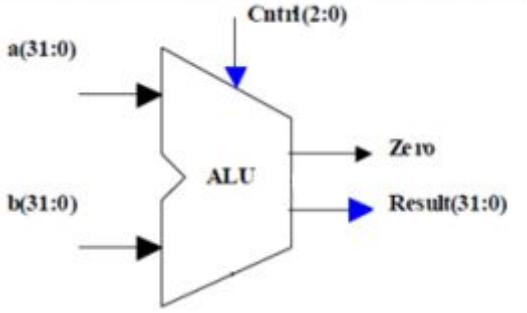

Figura 1.1. Esquema de ALU.

Dentro del módulo ALU que se muestra en la figura 1.1, se tienen diferentes operaciones lógicas las cuales serán ejecutadas dependiendo de la entrada “Cntrl”, esta entrada nos permite tener 8 opciones por el número de bits que está compuesto, no obstante solo se realizarán 7 operaciones. Las operaciones a realizar se indican en la tabla 1.1, dentro de la tabla se pueden observar las entradas, operaciones y el tipo de salida.

ALU Control Lines	Función	Operación
000	And	Result = a and b
001	Or	Result = a or b
010	Add	Result = a + b
011	Mov	Result = a
100	B upper	Result = b[15:0] & x"0000"
110	Substract	Result = a - b
111	Set less than	Consultar más adelante

Tabla 1.1. Operaciones de la ALU.

Para hacer lo posible la selección de operaciones se hizo uso del estatuto “with”, este es muy parecido al uso del “case”. El estatuto “with” hace referencia a una variable a un caso donde la variable referenciada tenga el valor indicado, el valor indicado es establecido por el comando “when” y el valor al cual se tiene que comparar la variable. Junto con cada “when” anteriormente es indicado los valores que se tendrá en la operación, esto se puede observar en la imagen 1.2.

```
with Cntrl select
        RES <=  A AND B when "000",
                A OR B  when "001",
                A + B   when "010",
                A       when "011",
                B(15 DOWNTO 0) & x"0000" when "100",
                A - B   when "110",
                DL      when "111",
                (others => '0') when others;
```

Figura 1.2. estatuto with.

Resultados

Las operaciones realizadas por la ALU son sencillas y solo se requiere el uso de la librería “NUMERIC_STD” y “LOGIC_1164”. En las siguientes imágenes podemos observar el test bench de la ALU al tenee diferentes valores en la entrada “Cntrl” y observamos la salida correspondiente a su operación seleccionada.

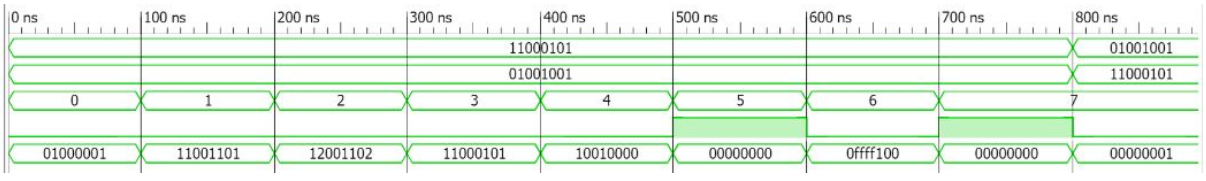

Figura 2.1. Test Bench.

Como se observa en la figura 2.1. se puede comprobar cada uno de los funcionamientos y son resumidos en la siguiente tabla para comprobar sus resultados.

A	11000101						
B	01001001						
Cntrl	0	1	2	3	4	6	7
Zero	0	0	0	0	0	0	1
Result	01000001	11001101	12001102	11000101	10010000	0FFFF100	00000000

Tabla 2.1. “Operaciones del Test Bench”.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ALU is
    Port ( A            :       in  STD_LOGIC_VECTOR (31 DOWNTO 0);
           B            :       in  STD_LOGIC_VECTOR (31 DOWNTO 0);
           Cntrl        :       in  STD_LOGIC_VECTOR (2 DOWNTO 0);
           Zero :       out  STD_LOGIC;
           Result :     out  STD_LOGIC_VECTOR (31 DOWNTO 0));
end ALU;

architecture Behavioral of ALU is
signal RES: STD_LOGIC_VECTOR (31 DOWNTO 0);
signal DL : STD_LOGIC_VECTOR (31 DOWNTO 0);
begin
        with Cntrl select
                RES <=  A AND B when "000",
                        A OR B  when "001",
                        A + B   when "010",
                        A       when "011",
                        B(15 DOWNTO 0) & x"0000" when "100",
                        A - B   when "110",
                        DL      when "111",
                        (others => '0') when others;
        DL <= x"00000001" when A<B else x"00000000";
        Zero <= '1' WHEN RES = x"00000000" else '0';
        Result <= RES;
end Behavioral;
```

Figura 2.2. Código de ALU 32 bits.

Conclusión

La ALU es uno de los componentes más esenciales de una computadora, ya que esta realiza todas las operaciones aritméticas como lo menciona su nombre, durante la práctica se pueden observar sus componentes y la clase de operaciones que esta ejecuta. Un módulo simple de crear, con algunas dificultades al momento de hacer uso del estatuto “with” pero fácil de implementar al momento de conocer su declaración y operación.

Reflexion Individual

Miguel Morales A00821541.

La segunda práctica me ayudó mucho a esclarecer ciertas dudas técnicas que tenía acerca de la ALU y qué operaciones realiza. Uno de los aspectos más complejos fue la realización de casos de operaciones, ya que se trató de utilizar el estatuto “case” y “when”. Al momento de realizar una investigación de cómo realizar casos en VHDL se vio que el estatuto “with” funcionaria para la operación que deseábamos.

Javier Mondragon A01365137:

En esta segunda práctica no tuvo dificultad implementar este módulo, ya que VHDL nos brinda herramientas de sintetización muy poderosas que nos ayudan a hacer este tipo de implementaciones de manera rápida. Creo que el único problema que tuvimos fue con la sintaxis al utilizar case en lugar de con y eso nos obligaba a agregar un proceso en el código, esto se arregló con un “with” encontrado en la referencia del xise.

Referencias

https://insights.sigasi.com/tech/signal-assignments-vhdl-withselect-whenelse-and-case/