TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MORELIA

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA-ELECTRÓNICA

ACADEMIA DE INGENIERÍA ELECTRÓNICA

MATERIA DE:

PROCESAMIENTO EN HARDWARE

PRÁCTICA #1

“ALU DE 4 BITS de 4 Operaciones”

Alumnos: Andrea Joanellie Hernández Alvarado 19121121

Pedro Eduardo Rojo Carrillo 19121155

Profesor: Arturo Méndez Patiño

Morelia Mich; a 22 de septiembre del 2023.

OBJETIVO (descripción de lo solicitado)

- Realizar la descripción de una unidad lógica aritmética de 4 bits en VHDL
- Instalar y tener funcional el IDE: Quartus Prime y un simulador (Questa o ModelSim)
- Comprender de manera cabal el uso del IDE: Quartus Prime
- Obtener los reportes de tiempos y de área del diseño
- Familiarizarse con la tarjeta que se usará en el curso (Cyclone IV)

INTRODUCCIÓN

Una unidad lógica aritmética es un componente digital combinacional de “n” bits que acepta dos operandos de entrada de “n” bits y un resultado también de “n” bits (además de cierta información extra que puede ser de utilidad como acarreo, desbordamiento, etc.)

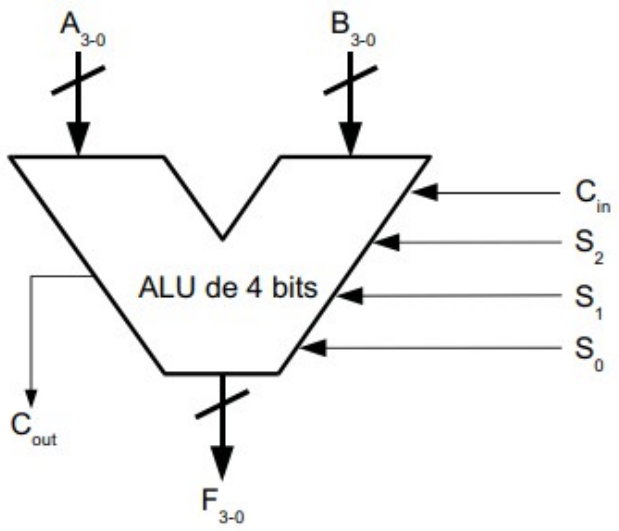


Figura 1.1: Símbolo de una ALU de 4 bits

Como se puede apreciar en la figura 1.1, extraída de [1], en el lado izquierdo se encuentra el operando A de 4 bits y de lado derecho el operando B de 4 bits también; por debajo del símbolo tenemos dos señalamientos marcados como Cout y F, los cuáles son el bit de acarreo de salida (para sumas binarias) y el resultado de la operación respectivamente, mientras que Cin, S0, S1, S2 son el bit de acarreo de entrada y los selectores de operación (Puede obtener información más detallada sobre las ALUs en [2]).

Quartus es una plataforma de diseño de hardware digital en PLD's y que cuenta con la integración necesaria para manejar tanto FPGA's como CPLD's, no obstante, en este curso se estará usando únicamente las funciones de integración enfocadas a FPGA's, más específicamente las de la Cyclone IV, dicha plataforma es proporcionada por la empresa de desarrollo de sistemas embebidos y digitales Altera, la cuál fue comprada por Intel. Quartus posee dos ediciones distintas entre sí y enfocadas a distintos dispositivos: Quartus ii y Quartus Prime, el primero se enfoca en los dispositivos más antiguos de la marca, tales como la Cyclone I a III, Stratix I a III, FPGA Arria GX, etc. mientras que el segundo cuenta con soporte para dispositivos más modernos como lo son: Cyclone IV a 10, Arria II, Stratix IV, etc. (Para información más detallada consulte [3]).

Una segunda definición del IDE puede ser revisada en [4] "La plataforma QUARTUS integra herramientas de desarrollo necesarias para procesar diseños en forma amigable e incluso manejar proyectos jerárquicos. También cuenta con métodos poderosos de síntesis lógica, compilación, partición, simulación funcional, simulación en tiempo y simulación enlazada con varios dispositivos".

Otro de los aspectos importantes que se revisarán durante el curso es el reporte llamado "propagation delay", para el cuál podemos encontrar una definición bastante congruente y sólida en [5]. "Propagation delay" se puede interpretar como el tiempo que le toma a una señal viajar de un registro al siguiente, esta definición nos permite comprender la importancia de este reporte sobre todo en una aplicación de alto desempeño, debido a que nos permite conocer la eficiencia del diseño creado en la FPGA.

En [5] se menciona también que el tiempo de propagación no debe de superar el tiempo del reloj (en caso de que se tenga uno), ya que esto ocasionaría que el diseño no funcionase de forma adecuada, añadiéndole mucha más importancia a este reporte.

Por otro lado el reporte de área es también una parte importante para conocer que tanto exige nuestro diseño de las capacidades de la FPGA, ya que este nos muestra cuántos elementos lógicos está utilizando el diseño así como registros, pines, multiplicadores embebidos y PLLs. Por lo que también se estará reportando a lo largo del curso.

La Cyclone IV es una tarjeta que fue lanzada en el año 2009, siendo relativamente moderna en comparación con otros dispositivos antes mencionados como la Cyclone II, cuenta con 6000 elementos lógicos, 15 bloques de procesamiento de señal digital (DSP), y 30 multiplicadores embebidos de 9 bits. [6]

En cuánto a la tarjeta de desarrollo, se utilizará la Cyclone IV mostrada en la figura 1.2

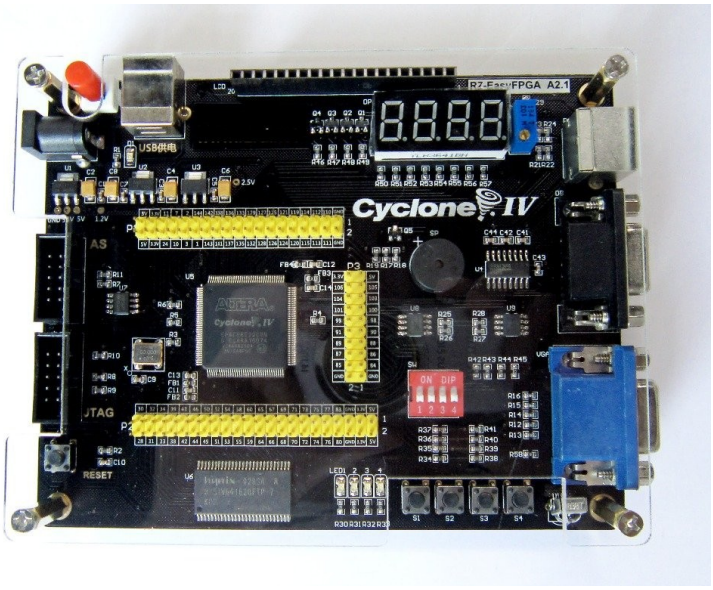


Figura 1.2: Tarjeta de desarrollo Cyclone IV

Y la distribución de las tiras de pines macho se puede observar en 1.3

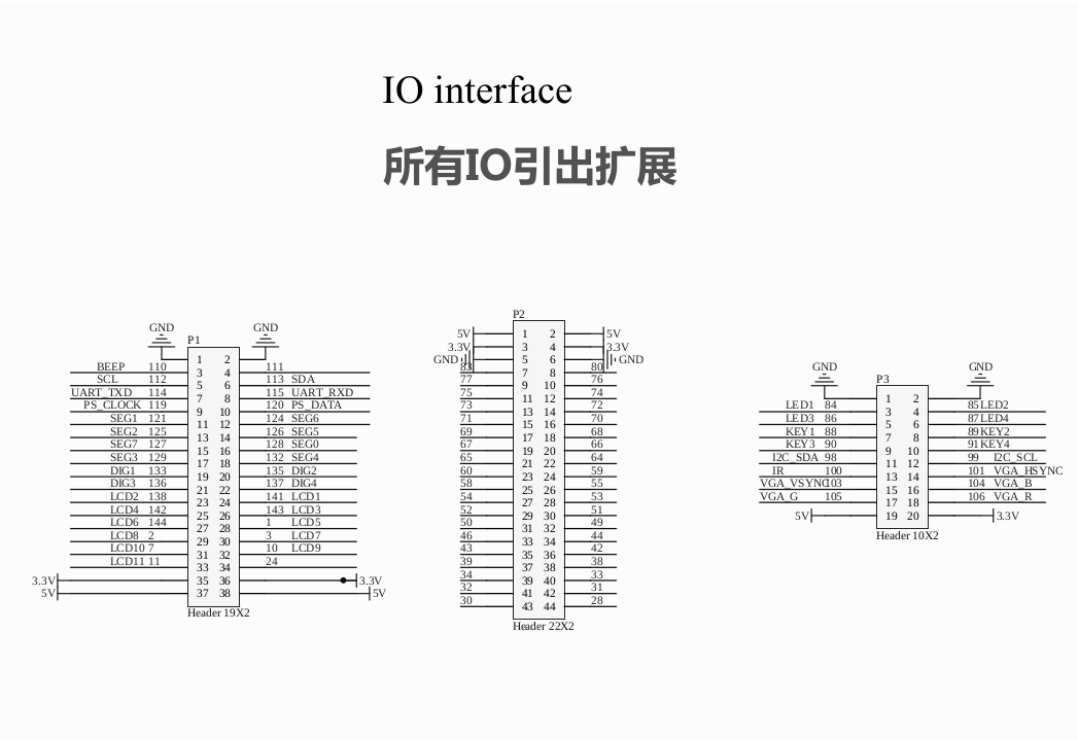


Figura 1.3: Conexión de la interfaz IO

CÓDIGO

```
1  Library IEEE;
2  use IEEE.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use IEEE.numeric_std.all;
5  library work;
6  use work.ALL;
7
8
9  entity ALU4Bits is
10     port (
11        INA : in  std_logic_vector(3 downto 0);
12        INB : in  std_logic_vector(3 downto 0);
13        SEL : in  std_logic_vector(1 downto 0);
14        RES : out std_logic_vector(3 downto 0)
15     );
16 end entity ALU4Bits;
17
18 architecture rtl of ALU4Bits is
19    signal r_RES : std_logic_vector(3 downto 0);
20 begin
21
22    with SEL select
23       r_RES <= INA+INB                              when "00",
24                INA-INB                              when "01",
25                INA(1 downto 0) * INB(1 downto 0)    when "10",
26                INA or INB                           when others;
27
28    RES <= not r_RES;
29
30 end architecture;
```

Figura 1.4: Código de la ALU de 4 bits

En las primeras líneas de código (1 a la 4) tenemos las declaraciones de uso de las librerías estándar de la IEEE las cuáles son std_logic_1164, std_logic_unsigned y numeric_std, así mismo para poder arrancar el simulador Questa desde la línea de comandos sin problemas se importó la librería work, que permite que los archivos puedan ser codificados y compilados por los comandos vcom y vsim de questa.

Después, de las lineas 9 a 16 se tiene la declaración de la entidad principal la cuál para este caso es la ALU de 4 bits, siendo INA e INB los operandos de la ALU y vectores de 4 bits cada uno, SEL el selector de la operación; dado que serán 4 operaciones las que se ejecutarán sólo se necesita un vector de 2 bits, por último RES, también de 4 bits y es en donde se tendrá el resultado de la operación.

Más abajo en el código, desde la línea 18 hasta el final se encuentra la descripción de la arquitectura en donde se está utilizando una sentencia “with” “select” para poder seleccionar la operación que se realizará entre los operandos.

Antes de inicializar la arquitectura, es decir, en la línea 19 se declara una “signal” o registro que guardará el resultado de las operaciones para después negarlo, más adelante se dejará en claro el porque de esta decisión.

Después del begin en la línea 20 se puede ver que si el vector SEL tiene el valor de “00” entonces se realizará una suma, si es “01” se realizará una resta con los números codificados en complemento A2, si es “10” se realizará la multiplicación de los bits menos significativos de los vectores INA e INB, esto es porque para poder almacenar el resultado de una multiplicación entre dos vectores de 4 bits se requieren al menos 8 bits y dado que el vector de salida es de 4, se tiene que reducir dicho número; cabe destacar que la operación de multiplicación se realiza con los números codificados en complemento A2. Por último cuando SEL es igual a “11” se realiza una or entre los operandos.

En la línea 28 se le da el resultado a RES utilizando la negación del registro r_RES, esto se tuvo que realizar debido a la conexión de los LEDs, (como se puede apreciar en la figura 1.5) ya que si por el pin de la FPGA se envía un 1 lógico entonces el LED se apaga y si se envía un 0 lógico el LED se enciende; por ello, para evitar trabajar con esta lógica negada e interpretar de una forma más sencilla los resultados arrojados en hardware se negó el resultado final.

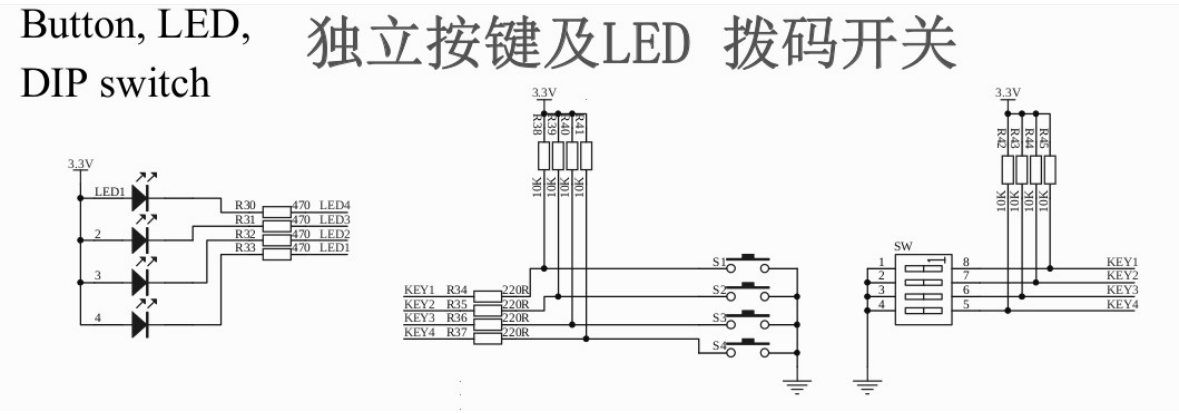


Figura 1.5: Conexión de LEDs y botones

REPORTE DE TIEMPOS

	Input Port	Output Port	RR	RF	FR	FF
1	SEL[1]	RES[1]	8.557	8.619	8.760	8.966
2	SEL[1]	RES[3]	8.663	8.704	8.837	9.035
3	SEL[1]	RES[0]	9.011	9.861	9.967	9.416
4	SEL[1]	RES[2]	9.104	9.346	9.364	9.563
5	INA[1]	RES[1]	9.712	9.952	10.045	10.234
6	INA[1]	RES[3]	9.918	10.019	10.193	10.356
7	INA[0]	RES[0]	10.143	10.219	10.243	10.514
8	INA[3]	RES[3]	10.179	10.278	10.449	10.582
9	INB[3]	RES[3]	10.425	10.527	10.697	10.819
10	INA[0]	RES[1]	10.269	10.441	10.710	10.944
11	INA[0]	RES[3]	10.346	10.447	10.787	10.950
12	INB[1]	RES[1]	10.526	10.763	10.799	10.983
13	INA[2]	RES[3]	10.508	10.609	11.017	11.180
14	INA[1]	RES[2]	11.025	11.319	11.266	11.354
15	INB[0]	RES[0]	11.001	11.126	11.267	11.435
16	INB[1]	RES[3]	11.106	11.212	11.322	11.485
17	SEL[0]	RES[0]	11.197	11.322	11.406	11.524
18	INA[2]	RES[2]	11.186	11.363	11.372	11.724
19	INB[2]	RES[3]	11.340	11.503	11.634	11.797
20	INB[0]	RES[1]	11.322	11.556	11.631	11.865
21	INB[0]	RES[3]	11.399	11.562	11.708	11.871
22	INA[0]	RES[2]	11.453	11.747	11.716	11.948
23	SEL[0]	RES[1]	11.518	11.752	11.720	11.954
24	SEL[0]	RES[3]	11.595	11.758	11.797	11.960
25	INB[2]	RES[2]	11.781	12.047	12.032	12.341
26	INB[1]	RES[2]	12.213	12.507	12.409	12.703
27	INB[0]	RES[2]	12.328	12.605	12.637	12.871
28	SEL[0]	RES[2]	12.524	12.801	12.726	13.010

Figura 1.6: Reporte de propagación del retraso

Los tiempos manejados por la figura 1.6 están dados en nano segundos y como se puede apreciar, el tiempo máximo de retraso es de 13.010ns, lo que quiere decir que esta aplicación puede correr como mínimo a este periodo de reacción, visto en frecuencia, la ALU puede como máximo trabajar y ver un cambio en las entradas a 76.86MHz.

REPORTE DE ÁREA

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Sep 21 18:23:52 2023
Quartus Prime Version	22.1std.2 Build 922 07/20/2023 SC Lite Edition
Revision Name	ALU4Bits
Top-level Entity Name	ALU4Bits
Family	Cyclone IV E
Device	EP4CE6E22C8
Timing Models	Final
Total logic elements	20 / 6,272 (< 1 %)
Total registers	0
Total pins	14 / 92 (15 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

Figura 1.7: Reporte de área

En la figura 1.7 se puede apreciar que en total el diseño utiliza 20 elementos lógicos, es decir, menos del 1% del total disponibles en la FPGA; se utilizan 14 pins, 0 multiplicadores embebidos y 0 PLLs. Lo que quiere decir que la ALU utiliza multiplicadores por software, debido al bajo número de bits que esta requiere.

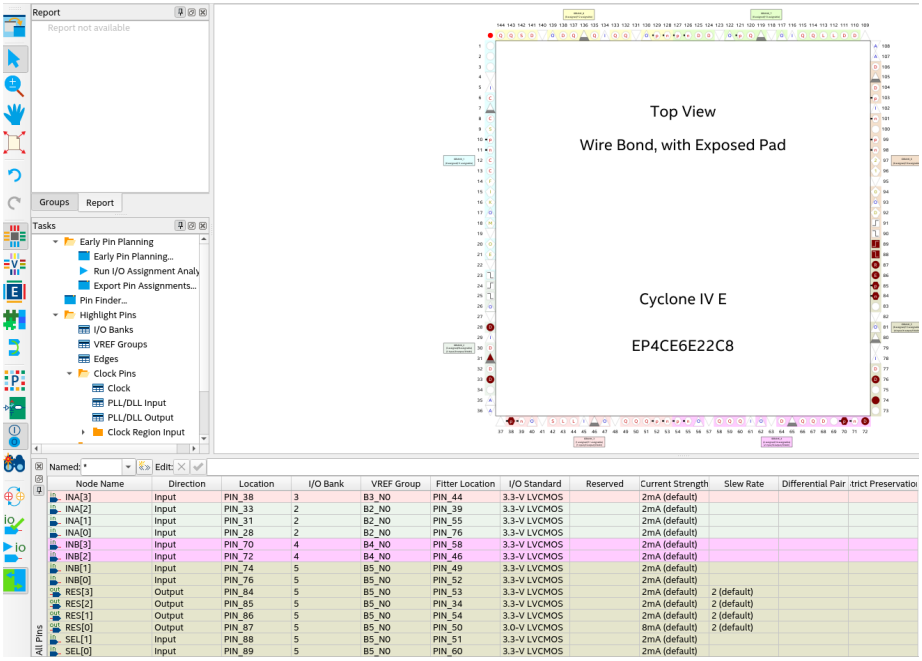
CÓDIGO DEL TESTBENCH

```vhdl
Library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;


entity tb_ALU4Bits is
end entity tb_ALU4Bits;

architecture rtl of tb_ALU4Bits is
    constant PERIOD      : time := 1 ns;
    signal INA, INB, RES   : std_logic_vector(3 downto 0) := (others => '0');
    signal SEL             : std_logic_vector(1 downto 0) := (others => '0');
    signal r_RES           : std_logic_vector(3 downto 0) := (others => '0');

    component ALU4Bits is
        port (
            INA : in  std_logic_vector(3 downto 0);
            INB : in  std_logic_vector(3 downto 0);
            SEL : in  std_logic_vector(1 downto 0);
            RES : out std_logic_vector(3 downto 0)
        );
    end component ALU4Bits;

begin

    --Instatiation
    U1 : ALU4Bits port map(
        INA => INA,
        INB => INB,
        SEL => SEL,
        RES => r_RES
    );

    stim_proc: process
    begin
    for i in 0 to 3 loop
        for j in 0 to 15 loop
          for k in 0 to 15 loop
             INA <= std_logic_vector( to_unsigned(j, INA'length));
             INB <= std_logic_vector( to_unsigned(k, INB'length));
             SEL <= std_logic_vector( to_unsigned(i, SEL'length));
             wait for PERIOD;
           end loop;
         end loop;
       end loop;
       wait;
    end process;

    RES <= not r_RES;

end architecture rtl;
```

Figura 1.8: Testbench para ALU de 4 bits

En el código anterior se destaca el uso de una constante llamada period, la cuál es el tiempo de duración de cada señal, dado en este caso por 1 ns. En el proceso de estimulo se utilizaron 3 ciclos for anidados para así poder simular y observar todas las combinaciones posibles. Dentro de los for para poder usar los iterados como valor en los vectores se casteó dicho entero signado a std_logic_vector sin signo, es decir, primeramente el entero se convirtió a un entero sin signo, usando la longitud del vector, para después convertirlo al tipo std_logic_vector. Similar al diseño general se negó la salida para evitar errores de interpretación de las operaciones.

ASIGNACIÓN DE PINES EN LA FPGA



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	trict Preservatio
INA[3]	Input	PIN_38	3	B3_N0	PIN_44	3.3-V LVCMOS		2mA (default)			
INA[2]	Input	PIN_33	2	B2_N0	PIN_39	3.3-V LVCMOS		2mA (default)			
INA[1]	Input	PIN_31	2	B2_N0	PIN_55	3.3-V LVCMOS		2mA (default)			
INA[0]	Input	PIN_28	2	B2_N0	PIN_76	3.3-V LVCMOS		2mA (default)			
INB[3]	Input	PIN_70	4	B4_N0	PIN_58	3.3-V LVCMOS		2mA (default)			
INB[2]	Input	PIN_72	4	B4_N0	PIN_46	3.3-V LVCMOS		2mA (default)			
INB[1]	Input	PIN_74	5	B5_N0	PIN_49	3.3-V LVCMOS		2mA (default)			
INB[0]	Input	PIN_76	5	B5_N0	PIN_52	3.3-V LVCMOS		2mA (default)			
RES[3]	Output	PIN_84	5	B5_N0	PIN_53	3.3-V LVCMOS		2mA (default)	2 (default)		
RES[2]	Output	PIN_85	5	B5_N0	PIN_34	3.3-V LVCMOS		2mA (default)	2 (default)		
RES[1]	Output	PIN_86	5	B5_N0	PIN_54	3.3-V LVCMOS		2mA (default)	2 (default)		
RES[0]	Output	PIN_87	5	B5_N0	PIN_50	3.0-V LVCMOS		8mA (default)	2 (default)		
SEL[1]	Input	PIN_88	5	B5_N0	PIN_51	3.3-V LVCMOS		2mA (default)			
SEL[0]	Input	PIN_89	5	B5_N0	PIN_60	3.3-V LVCMOS		2mA (default)			

Figura 1.9: Asignación de pines en la FPGA

En la figura 1.9 se puede observar que todos lo pines están configurados en el estándar de voltaje 3.3-V LVCMOS debido a que el voltaje con el que se proporcionaba a los pines físicamente era 3.3V, y esta configuración fue añadida al selector de pines.

También durante la programación de la FPGA se encontró un problema referente al uso de los pines de la misma, problema que se puede ver en la sección “problemas referentes a hardware” en el problema 3.

CAPTURAS DE PANTALLA DE LA SIMULACIÓN

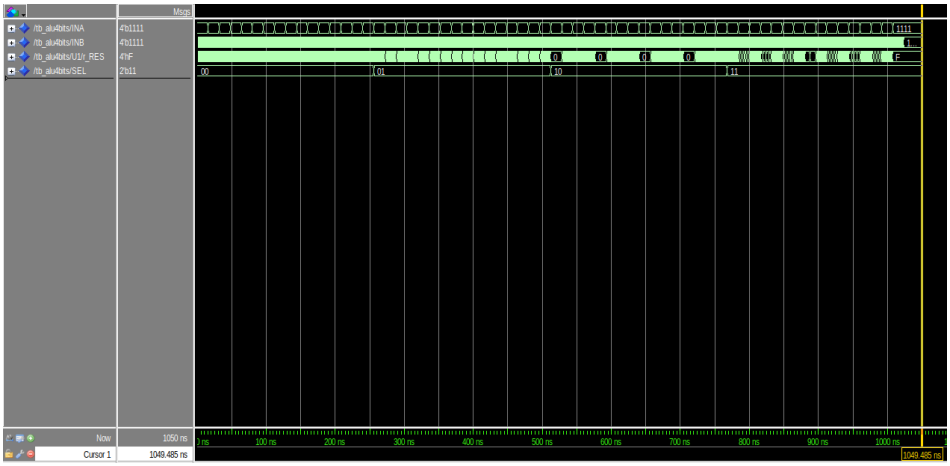


Figura 1.10: Vista general de la simulación

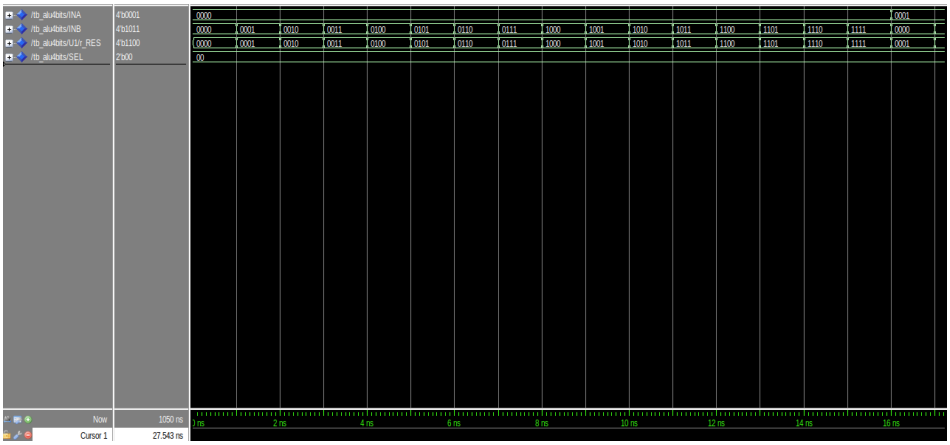


Figura 1.11: Ejemplo del caso de suma

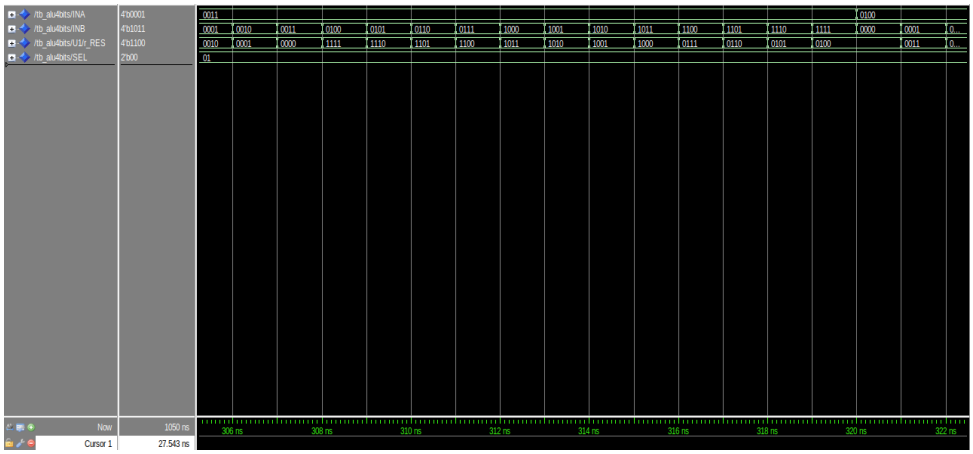


Figura 1.12: Ejemplo del caso resta

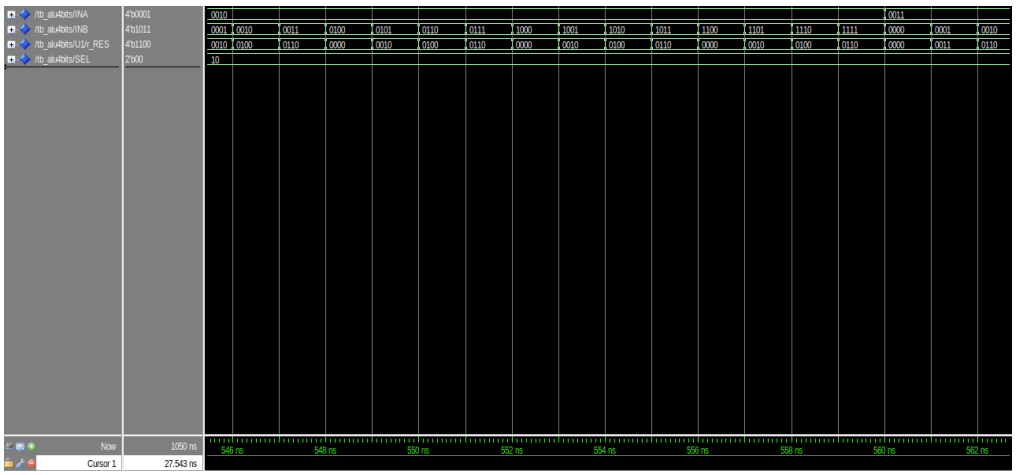


Figura 1.13 Ejemplo del caso Multiplicación

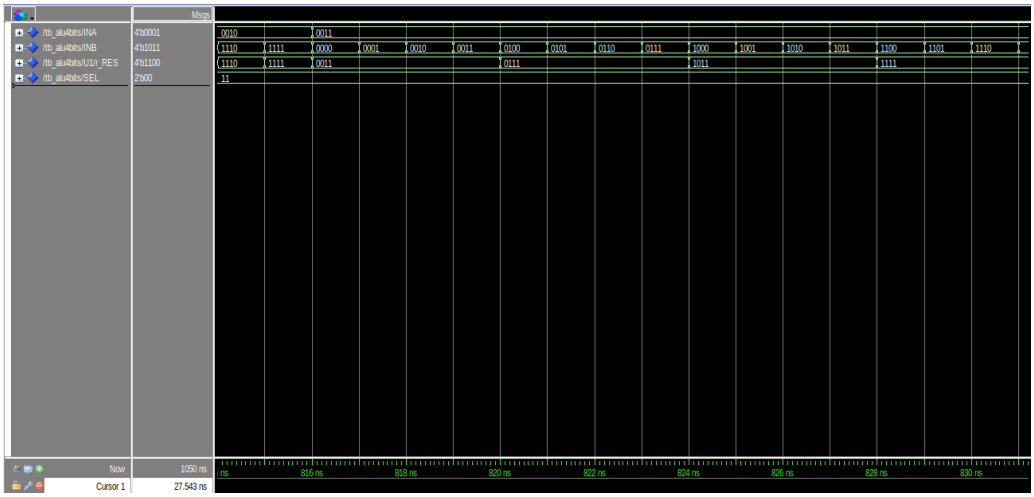


Figura 1.14: Ejemplo del caso OR

PROBLEMAS ENCONTRADOS

Problemas referentes a instalación de Quartus

1. Ejecución de script de instalación de quartus

Al momento de querer ejecutar en terminal el script de instalación este arrojaba errores, debido a que los permisos del archivo debían ser modificados

Solución: Utilizar el comando "*sudo chmod 777 QuartusLiteSetup-22.1std.2.922-linux.run*" para poder modificar los permisos y así ejecutarlo sin problemas con el comando "*./QuartusLiteSetup-22.1std.2.922-linux.run*"

2. Elección del directorio de instalación

Una vez modificados los permisos y ejecutado el cliente de instalación, se eligió como directorio de instalación /*home/usuario/altera/intelFPGA_lite/22.1.2* al momento de acceder al directorio y tratar de ejecutar el programa con el comando "*./quartus"* arrojaba un mensaje indicando que no podía acceder a las cabeceras estándar del kernel de linux y de librerías estándar tales como "libpng", etc.

Solución: Cambiar el directorio de instalación a */opt/intelFPGA_lite/22.1.2* y modificar los accesos y permisos del directorio con el comando "chmod -R 777 /o*pt*/intelFPGA" para que cualquier usuario no root pueda acceder al directorio

3. Falta de un acceso de aplicación para poder ejecutar el programa sin ir al directorio de instalación

Cuando el programa se logró ejecutar, este no se podía encontrar en el buscador de aplicaciones de Linux Mint ni en ningún otro lugar.

Solución: Crear y programar un archivo .desktop para que linux pueda detectar la aplicación y se pueda buscar y ejecutar como una sin ir al directorio de instalación, el programa se puede observar en la figura 1.15. Dicho archivo fue guardado en el directorio */home/user/.local/share/applications/* para que de este modo el sistema lo pueda detectar como programa.

Figura 1.15: Programa .desktop para la detección de quartus por el sistema

Cabe destacar que en la figura 1.15 se encuentra el programa final, ya que este tuvo que pasar por más procesos y solución de errores

Problemas referentes a uso de Quartus

1. Encontrar el reporte de propagación de retardos

Al momento de tratar de buscar este reporte, se encontraba en internet frecuentemente capturas del Quartus ii que mostraban dicho reporte en la sección de reportes generales, sin embargo, en esta aplicación no se encontraba el programa integrado “Timing Analyzer”, cosa que cambiaría en Quartus Prime y que, por ende modificaría la forma de acceder a este reporte.

Solución: Después de una amplia búsqueda por la red, se encontró que para Quartus Prime dicho reporte estaba en el programa “Timing Analyzer” en la sección de “Datasheet”, también se encontró que primero se debía de generar la lista de redes de tiempos, leer el archivo SDC y actualizar la lista de redes de tiempos, para acto seguido ir la sección “Datasheet” y encontrar el reporte.

Problemas referentes a Questa

1. El simulador se abría y cerraba sólo

Para el primer intento de simulación al momento de dar click en Tools → Run simulation tool → RTL simulation el software se abría y se cerraba con código 0 (sin errores), por lo que no indicaba ningún problema en terminal ni tampoco en la bandeja de mensajes y errores de Quartus Prime

Solución: Ejecutar Quartus Prime en una terminal inspectora con el comando source y redirigir todas las salidas e impresiones de la terminal del programa (stderror, stdin y stdout) hacia la terminal inspectora para observar que el error era que hacía falta una licencia para utilizar Questa.

2. Obtención de la licencia de Questa

Al observar que ese era el error y que era necesario la obtención de la licencia se siguieron los pasos redactados en la solución

Solución:

1. Inscribirse con la cuenta institucional al sistema de auto activación de licencias de intel (Self-Service Licensing Center),

2. Una vez inscrito se proporcionó la dirección MAC o física de las máquinas, siendo el caso de las computadoras de escritorio el controlador de enrutamiento ethernet y en el caso de las Laptops la tarjeta de WIFI (existen tantas MAC como dispositivos de interconexión en un equipo)

3. Solicitar la licencia de Questa y esperar a que esta llegue por correo

4. Descargarla con la extensión .dat

3. Instalación de la licencia de Questa

El hecho de tener la licencia dentro del directorio de instalación no implicaba que esta fuera utilizada por el programa, ya que al observar el error de la terminal de inspección utilizado en el problema 1 de esta sección (y también algunos archivos readme.txt dentro del directorio de Questa) se observó que debía de haber una variable de entorno llamada “LM_LICENSE_FILE” que apuntara o bien fuera igual al directorio donde se encuentra la licencia.

Solución: Definir la variable de entorno LM_LICENSE_FILE en el archivo .zshrc (para zshell) o .bashrc (para bash) que es un archivo que se ejecuta en el momento en el que una nueva instancia de la shell es llamada. Añadir al archivo QuartusPrime.desktop la definición de la variable de entorno “LM_LICENSE_FILE” e igualarla al directorio donde se encuentra la licencia, esto se puede apreciar en la sección “Exec=env LM_LICENSE_FILE=….” de la figura 1.15.

4. Ejecución y simulación de testbench's en Questa

El poder darle la licencia a Questa permitió la ejecución del simulador mediante la opción Tools → Run simulation tool → RTL simulation, sin embargo, en la opción Gate level simulation (para correr un testbench) el simulador no respondía adecuadamente, ya que daba el error:

** Error (suppressible): (vsim-19) Failed to access library 'tb_ALU4Bits' at "tb_ALU4Bits".
No such file or directory. (errno = ENOENT)

Esto sucedía debido a que Questa no encontraba un archivo llamado "work.tb_ALU4Bits" el cuál contiene toda la información necesaria para poder compilar y simular lo indicado en el testbench. Dicho archivo no se podía encontrar porque la librería work de Questa no estaba compilada.

Solución: Compilar por medio de la terminal la librería work. (lamentablemente esto no solucionó todo).

5. Ajuste del entorno de Questa para su uso

Como se indicó en el problema anterior el compilar la librería work por linea de comandos no solucionaba el error y seguía apareciendo, ya que para la instancia de la shell que manejaba el programa dicha librería no estaba definida, sólo para la instancia dónde se había ejecutado el comando de compilación.

Solución: Añadir al PATH el directorio donde se encuentran los comandos vsim y vcom y controlar el arranque del programa desde la terminal siguiendo los pasos indicados a continuación:

1. Compilar los archivos de diseño y de testbench con el comando vcom file.vhd, si no da errores entonces se avanzar al siguiente paso, si es que arroja errores deben de atenderse, aunque sean warnings.

2. Ejecutar el comando vsim con los siguientes argumentos -t (tiempo de resolución de simulación) work.(archivo de testbench sin extensión), ejemplo: vsim -t 1ps work.tb_ALU4Bits.

Problemas referentes a VHDL

1. Desconocimiento sobre el sistema de multiplicación proporcionado por las librerías IEEE

Cuando se buscaba realizar la multiplicación se desconocía que el comportamiento de dicha operación en el diseño y porqué razón daba valores que no coincidían con lo esperado

Solución: Analizar los datos que el testbench arrojaba en la simulación para poder determinar que era lo que estaba multiplicando el diseño, el resultado fue que la multiplicación estaba codificada en complemento A2.

2. Casting de un entero a std_logic_vector

Cuando se diseño el testbench en un primer instante se buscó asignar el iterador de los ciclos for a los vectores lógicos, dando como resultado un error del tipado de las variables

Solución: Buscar por la red el error y observar que para poder asignar dicho iterador se debía de convertir a entero sin signo con la longitud del vector y luego castearlo a std_logic_vector.

Problemas referentes a hardware

1. Conexión de los push button, DIP Switch y LEDs

Cuando se trataba de enlazar a un pin físico los distintos elementos de la entidad se encontró que no había suficientes elementos en la placa de desarrollo para la conexión de los operandos.

Solución: Utilizar el IO interface mostrado en la figura 1.3 para poder conectar en protoboard los DIP Switch.

2. Cargar programa a la Cyclone IV

Durante la hora de laboratorio se tuvo un problema con el acceso del kernel a los puertos dev/tty impidiendo que se pudiera detectar el USB Blaster y programar la FPGA

Solución: Reiniciar la Laptop para que recargara los puertos y más tarde se desactivó el secure boot, ya que este mismo impedía el uso de puertos como el HDMI.

3. Asignación de pines en el “pin planner”

Al momento de tratar de compilar el proyecto salía un error que indicaba que se estaba creando una doble asignación en un pin a pesar de que en la figura 1.9 se podía apreciar que no había alguna doble asignación.

Solución: El pin en cuestión era utilizado como CE (chip enable) y para quitar dicha opción, click derecho en el dispositivo, seleccionar device → Device and Pin Options → Dual porpouse pins → nCEO y seleccionar dicha casilla como “Use as regular IO”.

CONCLUSIONES

Pedro Eduardo Rojo Carrillo

Durante la instalación de Quartus Prime fue ampliamente necesario el conocimiento adquirido de forma autodidacta sobre los sistemas operativos basados en Debian y como estos trabajan el sistema de directorios raíz, el sistema de permisos de usuario, etc. También fueron necesarios conocimientos adquiridos en la materia de sistemas operativos embebidos donde se revisó el funcionamiento de las terminales y el estándar de descriptores de archivos y programas en C (stdin, stdout, stderror), así como los código de salida, etc. Debido a que en internet no hay casi nada de información sobre como funciona el software de altera y como es que se puede llevar a cabo su instalación, fue necesario en la mayoría de los casos resolver los problemas de forma autodidacta. Así mismo es altamente valioso el poder contar con un apartado para establecer la infraestructura necesaria para poder programar una FPGA hacer los testbench’s, etc. Ya que si se hubiese optado por realizar algún diseño mucho más complejo, que requiriera un conocimiento más avanzado en materia de VHDL o del diseño digital, las complicaciones habrían sido mucho mayores. Optar por este sistema permitió al alumno en concentrarse en resolver los problemas de forma eficiente y definitiva, dejando todo el sistema listo y preparado para crear diseños mucho más complejos y que requieran un conocimiento más profundo sobre el procesamiento en hardware.

Andrea Joanellie Hernández Alvarado

Se tuvieron diversos problemas a lo largo de la práctica como se mencionó detalladamente en la sección de problemas encontrados de este reporte; sin embargo el primer problema presentado específicamente en mi caso, fue la descarga del software de quartus en una laptop propia que contaba con la versión de windows 11. El problema consistió en que al momento de querer crear o abrir archivos, el programa crasheaba sin más, por lo que se investigó en un foro de internet [7] y se llegó a que el software de quartus prime 21.1.1 no soportaba esta versión de windows, por ende el software se descargó en el sistema operativo de linux. Dado este cambio, se generaron algunos obstáculos como se vio más a detalle en los problemas referentes a la instalación de quartus.

Con respecto a la práctica desarrollada en laboratorio de la ALU de 4 bits, el primer problema surgió al utilizar ciertos pines de la FPGA que resultaron ser pines dedicados, los cuales se tuvieron que indicar que tendrían otro propósito, estos cambios se realizaron dentro del software. Con respecto a hardware se buscaba usar los componentes electrónicos dentro de la FPGA con la que se trabajo (cyclone iv) sin embargo no contaba con la cantidad necesaria de DIP switches para la práctica, ya que sólo contaba con uno y se tuvo que generar el circuito con otros dos DIP switches para la entrada A y B. El incorporado en la placa se utilizó para el selector de operación de la ALU. Un hecho menos relevante es que la protoboard utilizada generaba algunos falsos contactos con los DIP switch, por lo que generó incertidumbres de la veracidad en los datos obtenidos de los LED’s encendidos. Por último, se encontró que dentro de la sumatoria entre la entrada A y B, estos estaban en complemento a 2 y se disipó la creencia de un error de software.

BIBLIOGRAFÍA

[1] Departamento de Tecnología Electrónica. Accedido el 19 de septiembre de 2023. [En línea].
Disponible: http://www.dte.us.es/docencia/etsii/gii-ti/cedti/grupopilarparra/material/notas-tema-6/ALU_libroETC.pdf

[2] UNIGAL. “ALU (unidad lógica aritmética): definición, función y más | UNIGAL”. UNIGAL.MX . Accedido el 19 de septiembre de 2023. [En línea].
Disponible: https://unigal.mx/alu-unidad-logica-aritmetica-definicion-funcion-y-mas/

[3] intel. “Centro de descargas de software FPGA”. Accedido el 19 de septiembre de 2023. [En línea].
Disponible: https://www.intel.la/content/www/xl/es/collections/products/fpga/software/downloads.html?f:guidetm83741EA404664A899395C861EDA3D38B=[Intel®%20Stratix®;FPGA%20Stratix®%20IV]

[4] N. E. Chávez Rodríguez. “TUTORIAL DE DISEÑO DE10-Lite”. Páginas Personales. Accedido el 19 de septiembre de 2023. [En línea]. Disponible: http://profesores.fi-b.unam.mx/normaelva/Tutorial-DE10-Lite.pdf

[5] National Instruments. “Understanding Timing Considerations for FPGA VIs (FPGA Module)”. Engineer Ambitiously - NI. Accedido el 19 de septiembre de 2023. [En línea]. Disponible: https://www.ni.com/docs/en-US/bundle/labview-fpga-module/page/lvfpgaconcepts/registers.html#:~:text=Propagation%20delay%20is%20the%20time,not%20exceed%20the%20clock%20cycle

[6] intel. “FPGA Cyclone® IV E”. Accedido el 19 de septiembre de 2023. [En línea].
Disponible: https://ark.intel.com/content/www/xl/es/ark/products/series/141584/cyclone-iv-e-fpga.html?countrylabel=Latin%20America

[7] “Reddit - Dive into anything”. Reddit - Dive into anything. Accedido el 23 de septiembre de 2023. [En línea].
Disponible: https://www.reddit.com/r/FPGA/comments/y9wgx3/intel_quartus_prime_2111_on_windows_11/?rdt=37749