



CAMPUS QUERÉTARO

**TE2002B.1 DISEÑO CON LÓGICA PROGRAMABLE
(GRUPO 1)**

“Evidencia 2. Implementación y diseño de una Unidad de Control”

Evidencia presentada por la estudiante:

A01706095 - Naomi Estefanía Nieto Vega

Profesores:

Jesús Esteban Cienfuegos Zurita

Rick Leigh Swenson Durie

Fecha de entrega:

Viernes 26 de febrero de 2021

I. Introducción

En esta evidencia se desarrollará una Unidad de Control (UC) con las especificaciones que se muestran a continuación en la figura 1, como sabemos la unidad de control es un componente muy importante de la unidad central de procesamiento o mejor conocida como CPU, que se encarga de darle instrucciones a la unidad aritmética y lógica, la memoria y al sistema de entradas y salidas como responder ante las instrucciones. [2] A continuación, se desarrollará una unidad de control con las indicaciones que se muestran en la figura 1.

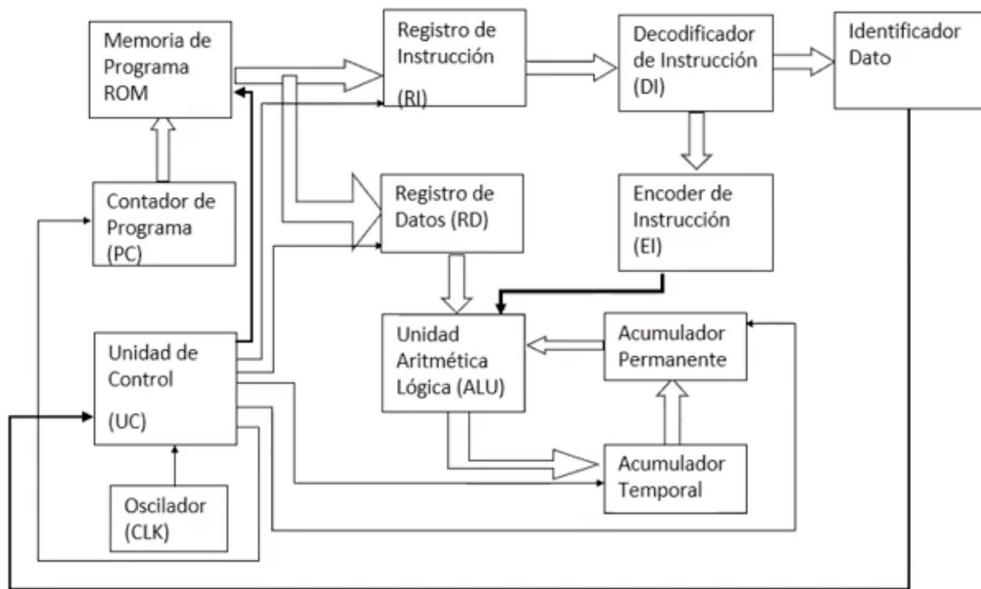


Figura 1. Diagrama de bloques que ilustra las instrucciones de la unidad de control.

II. Desarrollo

Para el desarrollo de esta evidencia se planteó primeramente el diseño de la unidad de control con un diagrama de estados (ver figura 2).

A. Diseño y modelado

En el diagrama de estados mostrado en la figura 2 se plantean los dos posibles casos que podría recibir la unidad de control, como podemos observar el primer recorrido es cuando inicias con dato, es decir indicando un 1 en el switch de dato, cuando esto sucede comienza desde el estado inicial después pasa a la memoria, de la memoria pasa al IR (Instruction Register), posteriormente pasa de nuevo a la memoria pero en este caso se pasa a una memoria auxiliar que se utiliza solo como apoyo en el diagrama ya que realmente no existe esta memoria auxiliar y se usa para evitar un ciclo sin fin, ya que de hacer lo contrario no avanzaría correctamente. Después pasa el registro de datos, luego al acumulador temporal y después al permanente donde será almacenado para después ser enviado a la ALU (la ALU se implementó anteriormente en la evidencia 1). Por otra parte cuando el ciclo comienza y el switch indicamos

0, es decir que no hay dato, el recorrido es un poco diferente, ya que ahora parte del estado inicial que es el “000000”, después pasa a la memoria, luego al IR, después al acumulador temporal y luego al permanente para ser almacenado.

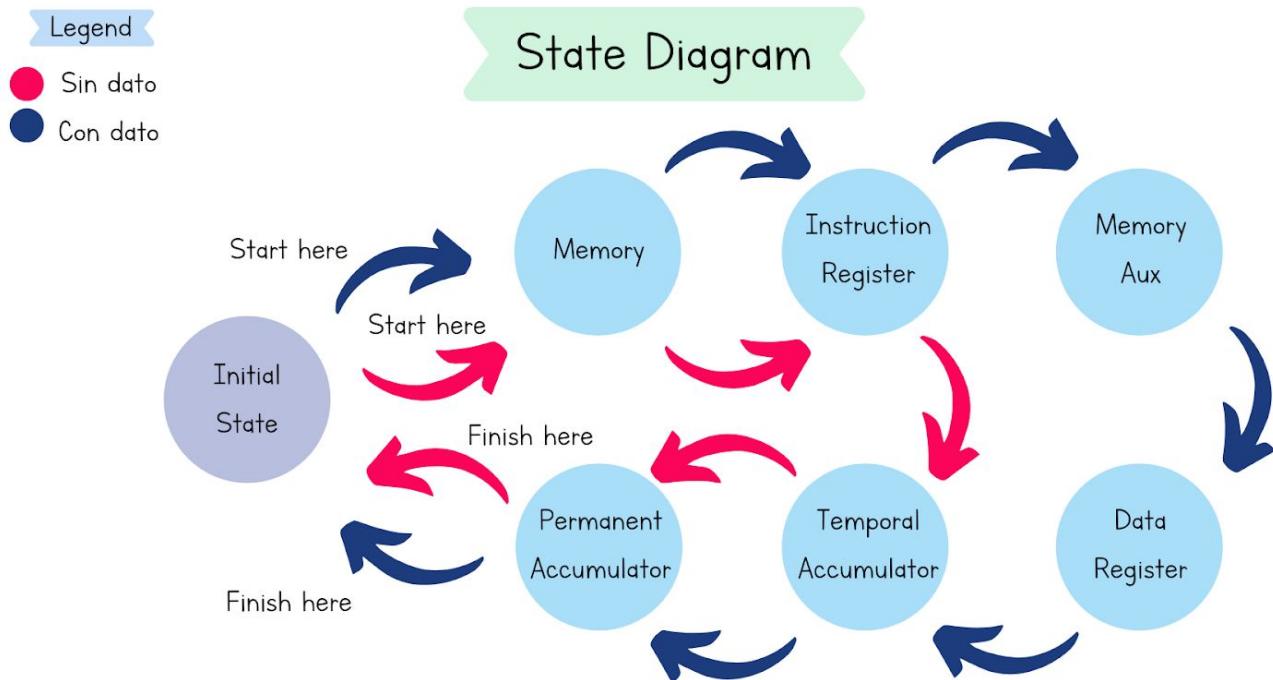


Figura 2. Diagrama de estados de la unidad de control que muestra el proceso cuando la entrada es con dato y cuando es sin dato.

Después realicé el diseño de la unidad de control indicando sus entradas que son el “Clk” clock, “DataId” que almacena si hay dato o no hay dato, y el “Rst” que es el reset. Y sus 6 salidas que son todas las salidas a los diferentes registros. Todo esto se ilustra en la figura 3.

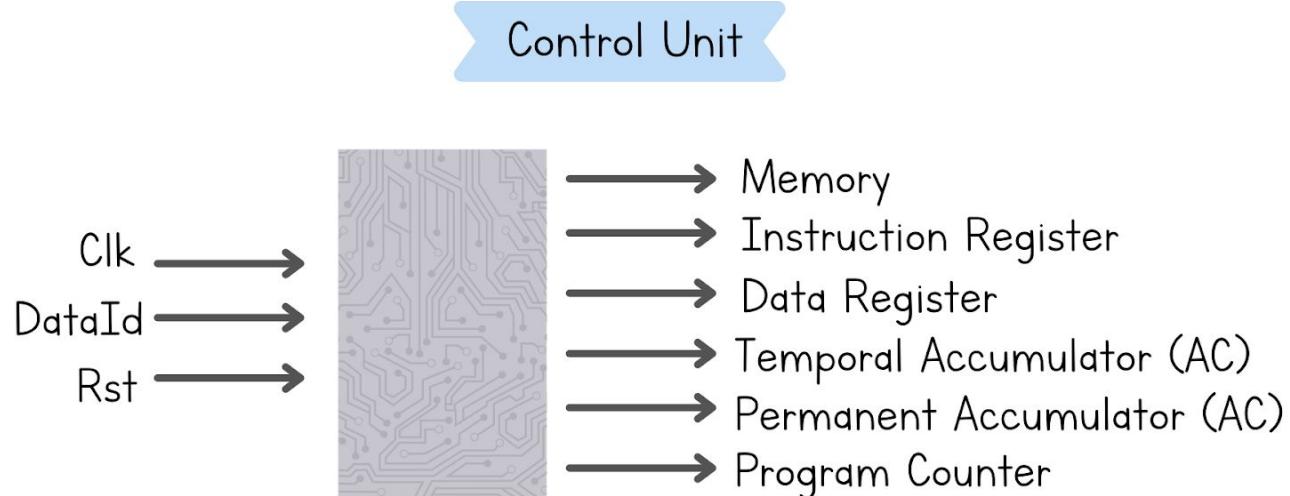


Figura 3. Diagrama esquemático de la unidad de control, en el podemos observar las entradas y salidas para su funcionamiento.

Posteriormente realicé otro diagrama de estados para cuando hay dato y cuando no hay dato, pero ahora indicando la salida que debería tener dependiendo de cuál ciclo inicie como se puede observar en la figura 4. Con este diagrama podemos observar los estados que se verán finalmente en el FPGA, es decir, los LEDs que se prenderán y los que permanecerán apagados simulando la unidad de control. Asimismo en la figura 5 podemos ver un diagrama esquemático obtenido de la visualización RTL de la unidad de control.

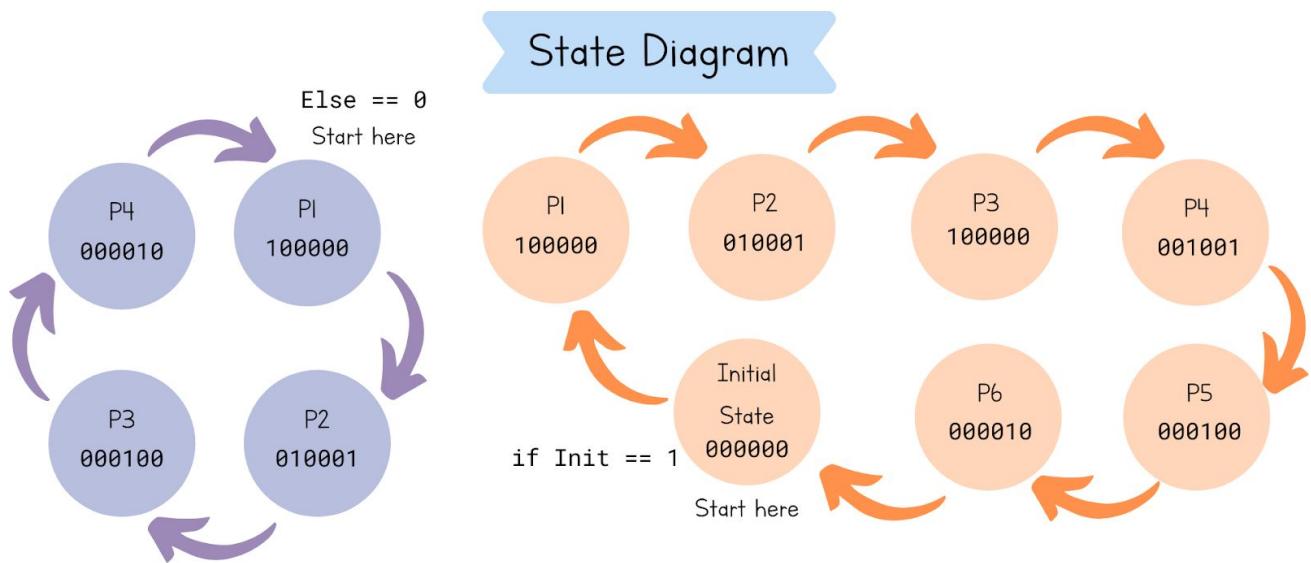


Figura 4. Diagrama de estados en el que se muestra el ciclo que recorrería si hay dato o no.

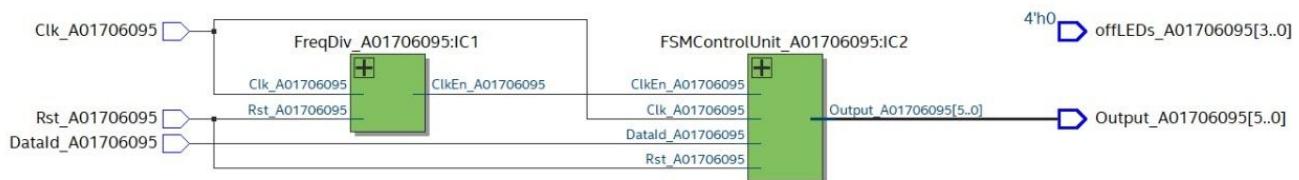


Figura 5. Vista RTL de la unidad de control.

Después en la figura 6 realicé un dibujo para ilustrar mejor la posición de los LEDs y los switches que asigne en el pin planner para explicar de forma más clara la descripción, en este caso los LEDs que no se usan se declaran apagados, y los que si se usan están en rojo, LEDs del 0 al 5. El switch 0 es el identificador del dato, con el indicamos si hay dato o no, y el switch 1 es el reset,

este debe estar siempre arriba para funcionar. Y posteriormente en la figura 7 observamos la asignación de estos pines desde la vista del pin planner. [Ver figura 6 y 7]

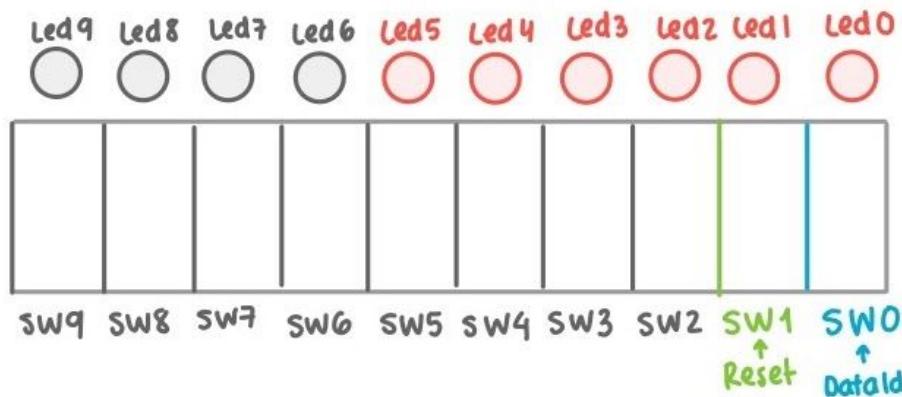


Figura 6. Diagrama esquemático del uso de la tarjeta FPGA de Intel.

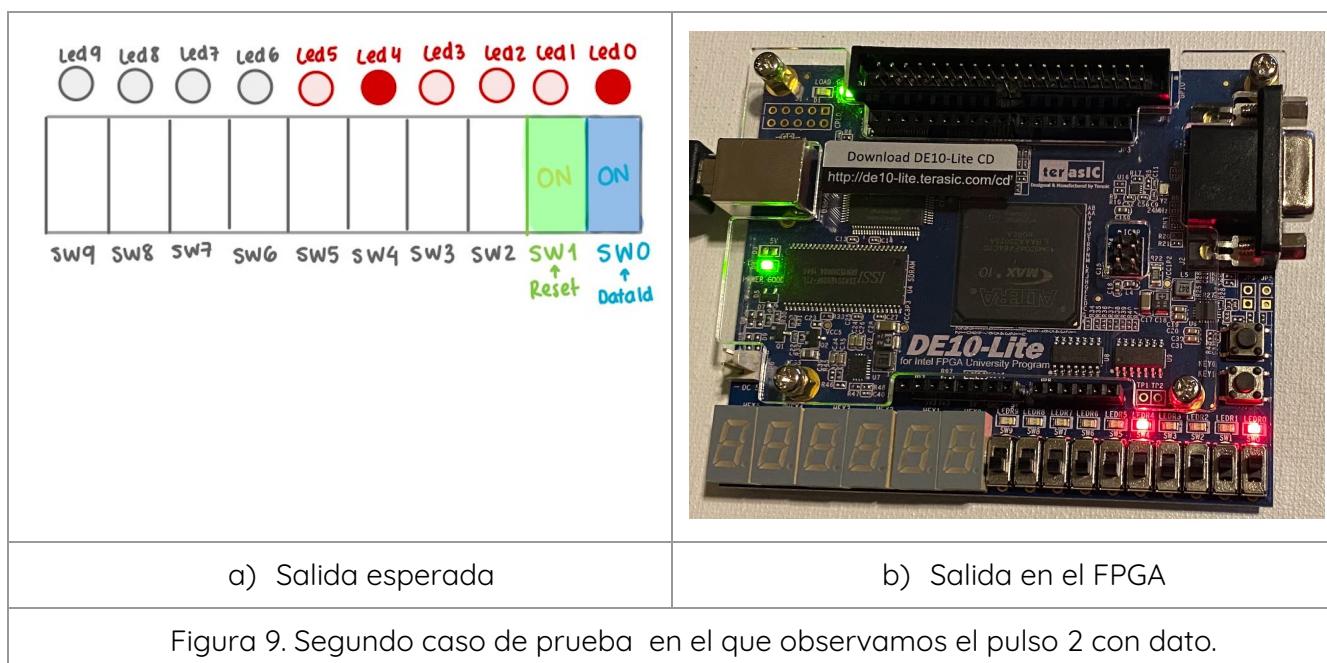
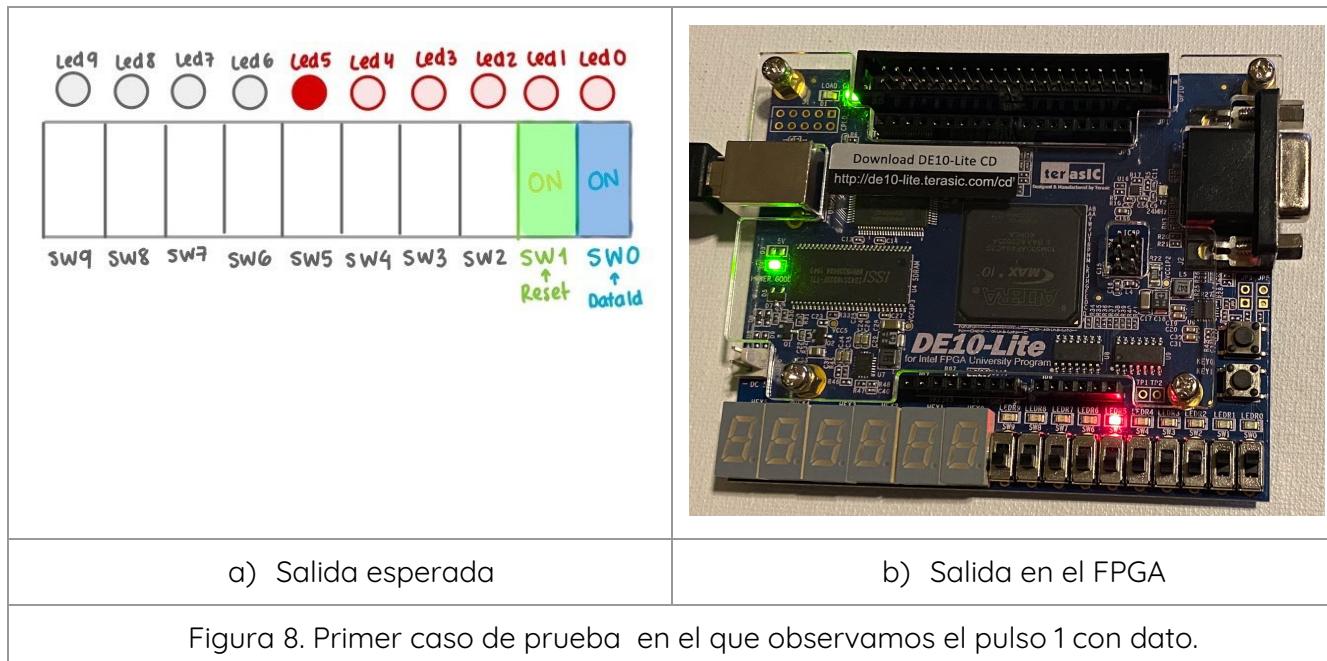
A continuación se adjunta el vínculo con dirección hacia la carpeta en drive en la que están todos los archivos del código fuente: [Evidencia2_TE2002B_A01706095](#).

Node Name	Direction	Location	I/O Bank	VREF Group
in Clk_A01706095	Input	PIN_P11	3	B3_NO
in DataId_A01706095	Input	PIN_C10	7	B7_NO
out offLEDs_A...706095[3]	Output	PIN_B11	7	B7_NO
out offLEDs_A...706095[2]	Output	PIN_A11	7	B7_NO
out offLEDs_A...706095[1]	Output	PIN_D14	7	B7_NO
out offLEDs_A...706095[0]	Output	PIN_E14	7	B7_NO
out Output_A...06095[5]	Output	PIN_C13	7	B7_NO
out Output_A...06095[4]	Output	PIN_D13	7	B7_NO
out Output_A...06095[3]	Output	PIN_B10	7	B7_NO
out Output_A...06095[2]	Output	PIN_A10	7	B7_NO
out Output_A...06095[1]	Output	PIN_A9	7	B7_NO
out Output_A...06095[0]	Output	PIN_A8	7	B7_NO
in Rst_A01706095	Input	PIN_C11	7	B7_NO

Figura 7. Captura de pantalla del Pin Planner con las ubicaciones en la tarjeta de cada una de las entradas y salidas.

B. Simulación en el FPGA DE-10 Lite

Para la parte de la simulación en Quartus Prime, primero dibujé un diagrama de salida esperada y después procedí a subir el programa a la tarjeta para probarlo. A continuación se muestran los casos de prueba cuando el switch de identificador de dato está en 1 como podemos observar en las figuras 8 a 13 las salidas en el FPGA tal como se planteó anteriormente para cada caso en el diagrama de estados.



<p>Led9 Led8 Led7 Led6 Led5 Led4 Led3 Led2 Led1 Led0 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 ↑ Reset SW0 ↑ DataIn</p>	
a) Salida esperada	b) Salida en el FPGA

Figura 10. Tercer caso de prueba en el que observamos el pulso 3 con dato.

<p>Led9 Led8 Led7 Led6 Led5 Led4 Led3 Led2 Led1 Led0 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 ↑ Reset SW0 ↑ DataIn</p>	
a) Salida esperada	b) Salida en el FPGA

Figura 11. Cuarto caso de prueba en el que observamos el pulso 4 con dato.

a) Salida esperada	b) Salida en el FPGA

Figura 12. Quinto caso de prueba en el que observamos el pulso 5 con dato.

a) Salida esperada	b) Salida en el FPGA

Figura 13. Sexto caso de prueba en el que observamos el pulso 6 con dato.

A continuación en las figuras 14 a 16 podemos observar la salida esperada y la salida obtenida en el FPGA una vez que se cargó el programa, pero en este caso las salidas son cuando el identificador de dato esta en 0, es decir que no hay dato, esto se puede comprobar con el diagrama de estados presentado en la figura 4.

a) Salida esperada	b) Salida en el FPGA

Figura 14. Primer caso de prueba en el que observamos el pulso 1 sin dato.

a) Salida esperada	b) Salida en el FPGA

Figura 15. Segundo caso de prueba en el que observamos el pulso 2 sin dato.

a) Salida esperada	b) Salida en el FPGA

Figura 16. Tercer caso de prueba en el que observamos el pulso 3 sin dato.

c) Salida esperada	d) Salida en el FPGA

Figura 16. Cuarto caso de prueba en el que observamos el pulso 4 sin dato.

III. Conclusiones

En conclusión, gracias a esta actividad pude comprender mejor el funcionamiento de una unidad de control y el flujo de los registros, asimismo pude comprender mejor la programación en VHDL y reforzar mis conocimientos en la organización de proyectos para planearlos mejor de forma que al pasar a la implementación sea más sencillo una vez que comprendí bien las especificaciones.

IV. Referencias

- [1] Intel.com. 2016. Intel DE10-Lite User Manual. [online] Available at: <https://www.intel.com/content/dam/altera-www/global/en_US/portal/dsn/42/doc-us-dsnbk-42-2912030810549-de10-lite-user-manual.pdf> [Accessed 21 February 2021].
- [2] GeeksforGeeks. (2019, 22 agosto). *Introduction of Control Unit and its Design.* <https://www.geeksforgeeks.org/introduction-of-control-unit-and-its-design/>