Ana Cristina Ballesteros Herrera
A01637832

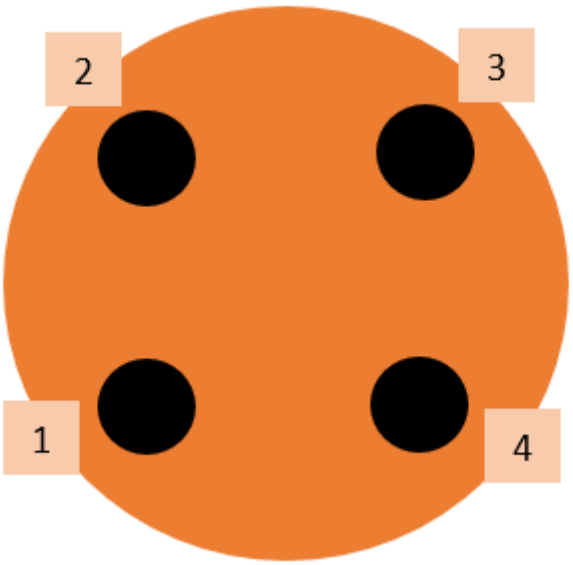
Situación Problema

Contexto

Se propuso una problemática por parte de la empresa Bayer de México, esta se basa en introducir un diseño de un sistema de control para llenado automático de botes. Esto optimizará el proceso de la empresa, ya que el llenado y sellado de estos sería automatizado. Esto se piensa llevar a cabo mediante sensores y actuadores disponibles en la máquina envasadora. Y al tener el análisis de la solución, utilizar herramientas para programar en VHDL el desarrollo.

Análisis y proceso

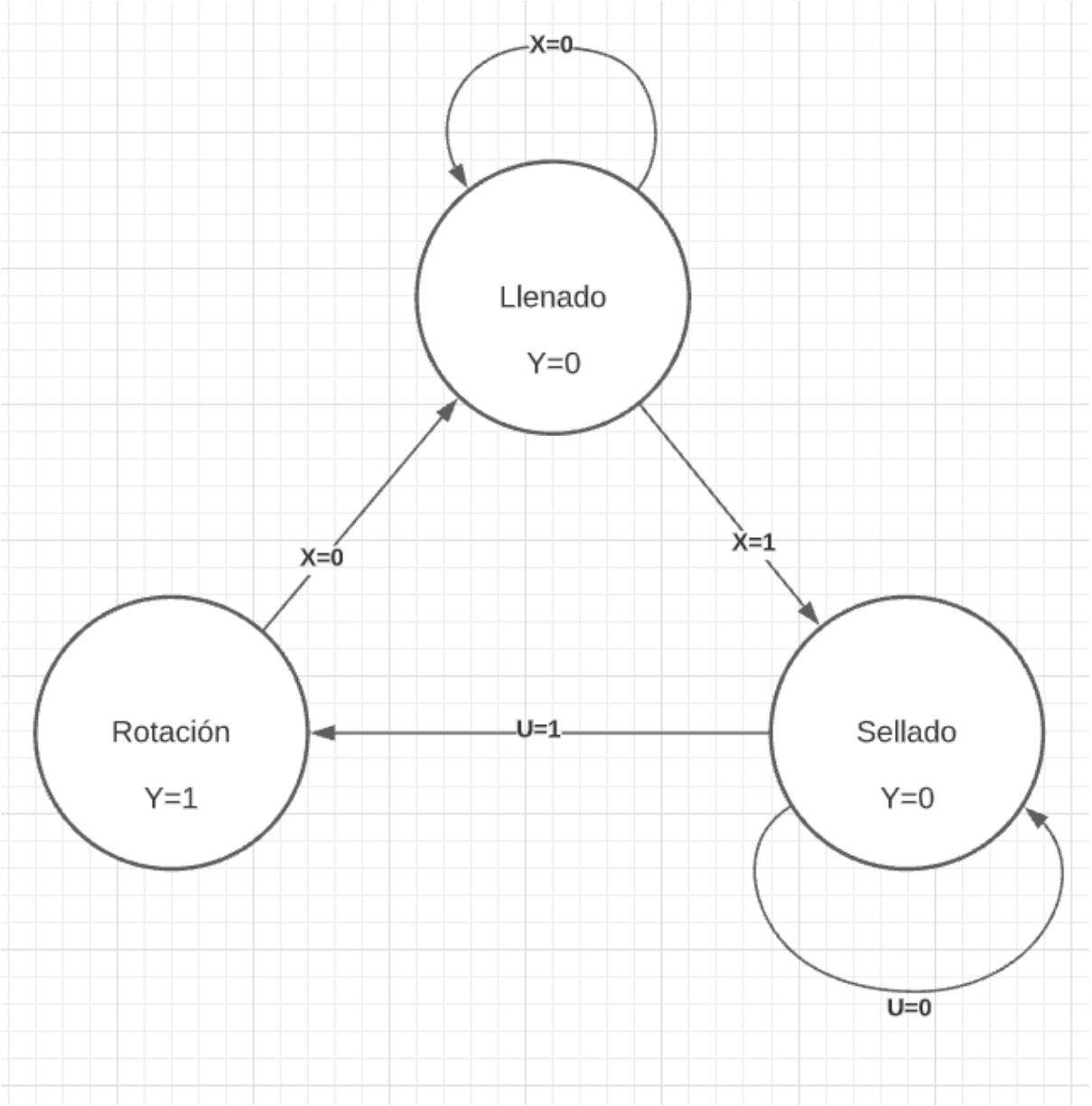
Para el proceso se eligieron tres etapas: llenado, sellado y recolección. Por otra parte, se piensa que la base sea circular, en la que se posicionan cuatro botes, como se muestra en la siguiente imagen.



La imagen está proyectada desde un punto de vista superior, y se enumeraron las posiciones (1,2,3,4) en las que se puede encontrar los botes. En la posición 3 es en donde se llevará a cabo el llenado y sellado. Para poder saber con certeza que el llenado se llevó a cabo se utilizará un sensor de peso, que si llega a cierto pesaje, deje de llenar el bote. Y para la parte de sellado se utilizará un sensor ultrasónico, para así verificar que la tapa está puesta y el bote está cerrado. Cuando estas dos etapas están listas, la plataforma rota para que un bote vacío pueda ser llenado y sellado, y así consecutivamente. Para la parte de la recolección del bote, se piensa que se realicé de manera manual.

En la siguiente imagen se puede observar el ciclo de estados.

Ana Cristina Ballesteros Herrera
A01637832



Desarrollo en ModelSim

Para el desarrollo en ModelSim se tomó en cuenta el ciclo de estados, y se programó a partir de estos. Sin embargo, primeramente se tiene un botón de reset, para que cuando este sea 0, el programa no tenga cambios. Y al tener programado este, se pudo seguir con la máquina de estados. Tomando en cuenta los sensores mencionados anteriormente, se tomó de base el siguiente código, en el que se utilizan 4 estados. Sin embargo, en esta situación se utilizarán 3 estados, por lo tanto se modifica.

Estados	Estado siguiente	Estado siguiente	Y	Código

	(Si X=0)	(Si X=1)		

S0 (estado inicial)	S1	S0	0	`when S0 =>`
				`Y <= '0';`
				`if X then`
				`NextState <= S0;`
				`else`
				`NextState <= S1;`
				`end if;`

S1	S1	S2	0	`when S1 =>`
				`Y <= '0';`
				`if X then`
				`NextState <= S2;`
				`else`
				`NextState <= S1;`
				`end if;`

S2	S3	S0	0	`when S2 =>`
				`Y <= '0';`
				`if X then`
				`NextState <= S0;`
				`else`
				`NextState <= S3;`
				`end if;`

S3	S1	S0	1	`when S3 =>`
				`Y <= '1';`
				`if X then`
				`NextState <= S0;`
				`else`
				`NextState <= S1;`
				`end if;`

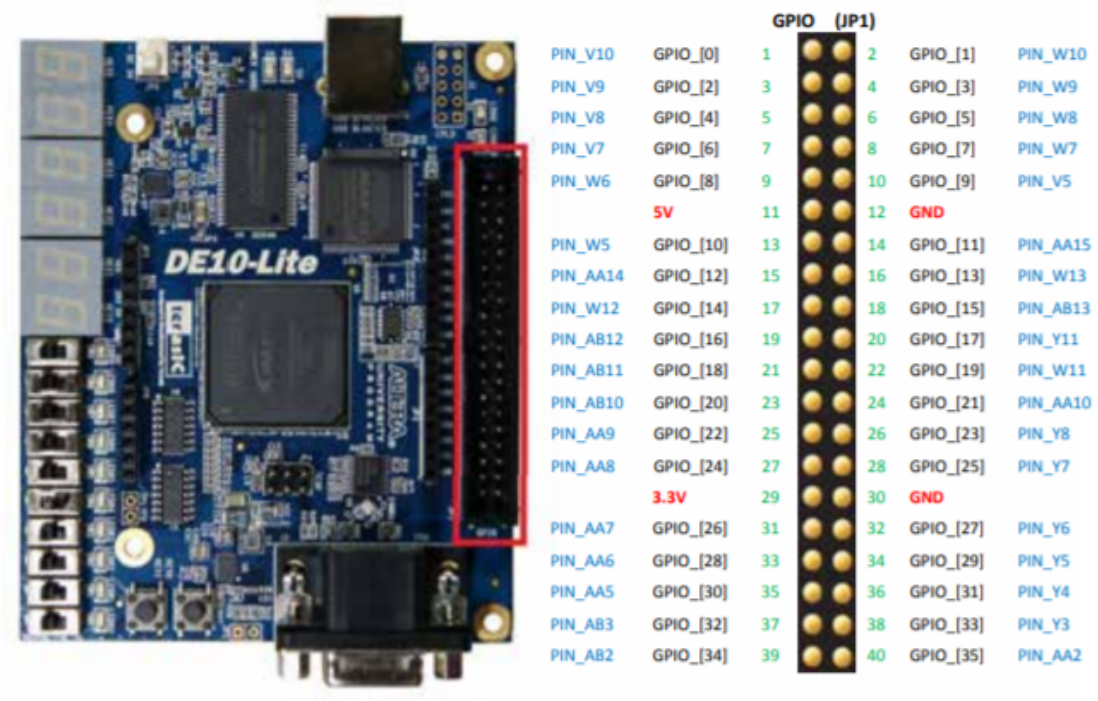
Posterior a esto se fueron agregando las diferentes variables que se deseaban, algunas de ellas siendo sensores y reset.

Desarrollo en tarjeta FPGA

Para el desarrollo en la tarjeta se utilizó el mismo procedimiento que para el desarrollo en ModelSim presentado anteriormente, y se decidió agregar el display 16x2, utilizado en la evidencia 2. Al unir estos dos códigos, se pudo obtener un código base, sin embargo se agregaron switches, leds y push buttons a este. Estos elementos que se agregaron, se utilizaron para representar sensores, actuadores y salidas del código.

Ana Cristina Ballesteros Herrera
A01637832

Para poder hacer las conexiones de la manera correcta, se utilizó el documento de Terasic, de donde se obtuvieron los nombres de los pines.



(Terasic, 2016)

También se tomaron en cuenta las especificaciones de la tarjeta FPGA, para los pines específicos de los leds, push buttons y switches, como se presenta a continuación.

Switches

Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTL

(Terasic, 2016)

Push buttons

Ana Cristina Ballesteros Herrera
A01637832

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY0	PIN_B8	Push-button[0]	3.3 V SCHMITT TRIGGER"
KEY1	PIN_A7	Push-button[1]	3.3 V SCHMITT TRIGGER"

(Terasic, 2016)

Leds

Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR0	PIN_A8	LED [0]	3.3-V LVTTL
LEDR1	PIN_A9	LED [1]	3.3-V LVTTL
LEDR2	PIN_A10	LED [2]	3.3-V LVTTL
LEDR3	PIN_B10	LED [3]	3.3-V LVTTL
LEDR4	PIN_D13	LED [4]	3.3-V LVTTL
LEDR5	PIN_C13	LED [5]	3.3-V LVTTL
LEDR6	PIN_E14	LED [6]	3.3-V LVTTL
LEDR7	PIN_D14	LED [7]	3.3-V LVTTL
LEDR8	PIN_A11	LED [8]	3.3-V LVTTL
LEDR9	PIN_B11	LED [9]	3.3-V LVTTL

(Terasic, 2016)

Tomando todos los elementos anteriores, el pin planner de la tarjeta quedó de la siguiente manera:

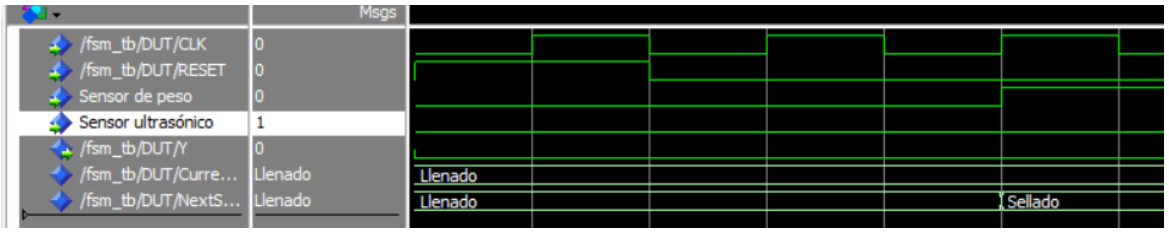
Node Name	Direction	Location	I/O Bank	VREF Group	itter Locatior	I/O Standard	Reserved	urrent Streng	Slew Rate	ifferential Pai	ict Preservati
CLK	Input	PIN_P11	3	B3_N0	PIN_P11	3.3-V LVTTL		8mA (default)			
data[7]	Output	PIN_V10	3	B3_N0	PIN_V10	3.3-V LVTTL		8mA (default)	2 (default)		
data[6]	Output	PIN_V9	3	B3_N0	PIN_V9	3.3-V LVTTL		8mA (default)	2 (default)		
data[5]	Output	PIN_V8	3	B3_N0	PIN_V8	3.3-V LVTTL		8mA (default)	2 (default)		
data[4]	Output	PIN_V7	3	B3_N0	PIN_V7	3.3-V LVTTL		8mA (default)	2 (default)		
data[3]	Output	PIN_W6	3	B3_N0	PIN_W6	3.3-V LVTTL		8mA (default)	2 (default)		
data[2]	Output	PIN_W5	3	B3_N0	PIN_W5	3.3-V LVTTL		8mA (default)	2 (default)		
data[1]	Output	PIN_AA14	4	B4_N0	PIN_AA14	3.3-V LVTTL		8mA (default)	2 (default)		
data[0]	Output	PIN_W12	4	B4_N0	PIN_W12	3.3-V LVTTL		8mA (default)	2 (default)		
en	Output	PIN_AB12	4	B4_N0	PIN_AB12	3.3-V LVTTL		8mA (default)	2 (default)		
RESET	Input	PIN_C10	7	B7_N0	PIN_C10	3.3-V LVTTL		8mA (default)			
rs	Output	PIN_AB11	4	B4_N0	PIN_AB11	3.3-V LVTTL		8mA (default)	2 (default)		
rw	Output	PIN_AB10	4	B4_N0	PIN_AB10	3.3-V LVTTL		8mA (default)	2 (default)		
U	Input	PIN_A7	7	B7_N0	PIN_A7	3.3-V LVTTL		8mA (default)			
X	Input	PIN_B8	7	B7_N0	PIN_B8	3.3-V LVTTL		8mA (default)			
Y	Output	PIN_A8	7	B7_N0	PIN_A8	3.3-V LVTTL		8mA (default)	2 (default)		
<<new node>>											

Como se puede observar en este, el LED0 se utilizó para representar la variable “Y” que define si se terminó el ciclo o no. El primer switch de la tarjeta se utilizó para modificar el RESET. Y los dos push buttons que hay se utilizaron para representar los sensores.

Simulaciones

Comenzando con la simulación en ModelSim se pudo observar a través de un Testbench que los estados en este se realizaban de manera correcta, al ir variando los dos sensores propuestos, mencionados anteriormente. Y esto fue lo que se visualizó en el programa ModelSim.

Ana Cristina Ballesteros Herrera
A01637832



Como se puede observar en la gráfica, primeramente los cambios se realizan cuando el CLK cambia de manera ascendente.

Se definió primeramente que el RESET fuera 1, por lo tanto se queda en la fase de llenado. Cuando el RESET se iguala a 0, el sensor de peso y ultrasónico son igual a 0, por lo tanto la etapa de llenado se queda de esta manera.

Posteriormente, el sensor de peso se enciende, y se pasa a la etapa de sellado. Que esto es correcto ya que el bote ya pesa lo que debe por la cantidad de pastillas.

En esta imagen, se puede observar primeramente que el estado esta en el de sellado, y este cambia al estado de rotación hasta que el sensor ultrasónico esta prendido, cabe mencionar que cuando esta en esta etapa, Y=1 porque el ciclo fue completado. El estado cambia nuevamente cuando el sensor ultrasónico está apagado y cambia a la etapa de llenado.

Para finalizar se prende el RESET, y se mantiene en etapa de llenado por esto mismo.

Para evidencia de la parte en ModelSim y Quartus, se adjuntaron los archivos de código, como videos y fotografías a la siguiente carpeta.

Situación Problema

Conclusiones

Para poder llevar a cabo esta práctica, se tuvo que entender de manera correcta el funcionamiento del LCD y las máquinas de estado. Al realizar el código, se tomaron de base el código para la evidencia 2, y el código en ModelSim para la situación problema. Y esto fue de gran ayuda, ya que al entender bien estos dos, se pudieron hacer aditamentos al código cuando se utilizó la tarjeta.

Primeramente se había optado por representar con switches los sensores, pero al ser switches, no regresan solos a su estado natural, o a estar apagados. Por lo tanto, se cambiaron a push buttons. Pero al cargar el programa a la tarjeta, sin estar presionando los push buttons, estaba

en etapa de rotación, y esto significaba que los push buttons en su estado natural eran igual a 1. Por lo tanto se cambió esto en la codificación, y se aprendió esto de estos componentes.

Por otra parte, el acercamiento con la tarjeta FPGA en la situación problema fue importante, ya que se pudo comprender mejor el funcionamiento y el desenvolvimiento de esta en este contexto.

Referencias

Terasic (2016). DE10-LITE User Manual. Terasic. Recuperado de. https://www.intel.com/content/dam/www/programmable/us/en/portal/dsn/42/doc-us-dsnbk-42-2912030810549-de10-lite-user-manual.pdf