Revista Científica y Tecnológica UPSE

Simulación electrónica del microprocesador GAL22V10 mediante el software Proteus basado en VHDL para virtualizar circuitos integrados

Electronic simulation of the GAL22V10 microprocessor using Proteus software based VHDL to virtualize integrated circuits

Luis Chuquimarca Jiménez * https://orcid.org/0000-0003-3296-4309, Paúl Suárez Ricardo https://orcid.org/0000-0001-1957-6563, Franklin López Silva https://orcid.org/0000-0003-4306-4058.

Universidad Estatal Península de Santa Elena, Ecuador.

Resumen

El presente artículo está enfocado hacia los dispositivos electrónicos FPGA que pueden ser usados dentro de la industria; para este caso se usó un microprocesador GAL22V10 como controlador de un proceso industrial de llenado de tanques, programado en código VHDL desarrollado en la herramienta de software llamada Cypress Warp Galaxy, el cual generó un archivo (JED) que se utilizó en el software simulador electrónico Proteus, el cual proporcionó una virtualización de los sistemas integrados. Luego del cual, se parametrizó las entradas y salidas del sistema de control, las cuales vienen siendo controladas por el microprocesador GAL22V10, donde se desarrolló el algoritmo del proceso industrial que se va a controlar, tomando en consideración las características técnicas de la hoja de datos del microprocesador para la respectiva configuración en el simulador Proteus. Los resultados obtenidos entre la vinculación del algoritmo en código VHDL en el microprocesador GAL22V10 simulado en Proteus se realizó de forma satisfactoria, cumpliendo con las condiciones del proceso industrial basado en circuitos integrados virtualizados.

Palabras clave:

FPGA, GAL22V10, VHDL, JED, Máquina Virtual.

Abstract

This article is focused on FPGA electronic devices that can be used within the industry, for this case a GAL22V10 microprocessor was used as a controller of an industrial process of filling tanks, programmed in VHDL code developed in the software tool called Cypress Warp Galaxy, which generated a file (JED) that was used in the electronic simulator software Proteus, which provided a virtualization of integrated systems. After which, the inputs and outputs of the control system were parameterized, which are being controlled by the GAL22V10 microprocessor, where the algorithm of the industrial process to be controlled was developed, taking into consideration the technical characteristics of the microprocessor data sheet for the respective configuration in the Proteus simulator. The results obtained between the linking of the algorithm in VHDL code in the GAL22V10 microprocessor simulated in Proteus were satisfactorily fulfilling the conditions of the industrial process based on virtualized integrated circuits.

Keywords:

FPGA, GAL22V10, VHDL, JED, Vitual Machine

Recibido: abril 9/ 2021 **Aceptado:** mayo 19/2021 **Publicado:** junio 25/ 2021

Forma de citar: Chuquimarca Jiménez, L.; Suárez Ricardo, P.; López Silva, F. (2021). Simulación electrónica del microprocesador GAL22V10 mediante el software Proteus basado en VHDL para virtualizar circuitos integrados. Revista Científica y Tecnológica UPSE, 8 (1) pág. 107-115. DOI: 10.26423/rctu.v8i1.573

* *Autor para correspondencia:* lchuquimarca@upse.edu.ec

1. Introducción

El desarrollo tecnológico actual, propone una visión de automatizar los sistemas y procesos para proponer mejores tiempos en ejecución y en calidad de un producto.

Un sinnúmero de ventajas se abren con los sistemas lógicos programables, gran parte del desarrollo se fundamenta en usar un lenguaje de programación como VHDL (Very High Speed Integrated Circuit Hardware Description Language) el uso de este ayuda a crear líneas de código que activan instrucciones que ejecutan varios procesos (Sosa, Garcia, Salinas, Ortega, & Hernandez, 2018)

Los ambientes virtuales hoy en día son más activos en el desarrollo tecnológico, los simuladores PROTEUS (Garcia & Valdes, 2018) y máquinas virtuales como Virtual Box ayudan a los investigadores a recrear sistemas automáticos que demuestren resultados esperados y luego evaluarlos para ver las posibles ventajas o desventajas.

Con la aplicación de las herramientas de software mencionadas se logra tener una vista preliminar antes de aplicar el sistema basado en microprocesadores y circuitos digitales, con lo cual se disminuye tiempo al momento de evaluar los errores en programación o en esquema electrónico, ya que la configuración y elaboración de estas aplicaciones de simulación poseen herramientas con códigos VHD y microprocesadores y simulan los resultados más cercanos a la realidad de todos los componentes electrónicos.

El proceso de llenado de líquido en tanques forma parte activa en la industria, se pueden automatizar con FPGA (field-programmable gate array) con sistema de programación en VHDL (Chavez & Vasquez , 2017), desde hace mucho tiempo el llenado de tanques ha ido evolucionando proponiéndose para este fin muchos sistemas de control desde los más básicos que solo usan una boya flotante para abrir o cerrar una válvula de llenado (Vargas , Cogua, & Rincón, 2017) hasta los más complejos con el uso de sensores entre niveles va desde un nivel bajo hasta un nivel alto, o con sensores de presión.

El sistema al que se refiere este proyecto se compone de sensores que miden el nivel y envían una señal que permiten ver y evaluar en qué nivel se encuentra el líquido y ejecuta la apertura o cerrado de una válvula. (Mogollón Baca, 2019)

El propósito de este proyecto es el uso de una herramienta de simulación de FPGA para simular el control de nivel en tanques de líquido. La propuesta sirve para usar la investigación en laboratorios virtuales de sistemas digitales, la importancia del mismo es crear el ambiente perfecto entre el lenguaje máquina programado en VHDL y un sistema real de control de llenado de tanques.

2. Marco teórico

2.1. FPGA

Los dispositivos electrónicos avanzados FPGA se basan en una matriz CLB (configurable logic block) conectados mediante interconexiones programables. Los FPGA se pueden reprogramar para la aplicación deseada tomando en consideración los requisitos de funcionalidad de fabricación del dispositivo. Esta característica distingue a los FPGA de los ASIC (application-specific integrated circuit), que se fabrican a medida para tareas de diseño específicas. Aunque se encuentran disponibles FPGA programables OTP (one-time programmable), los tipos dominantes están basados en SRAM (static random access memory), que pueden reprogramarse a medida que evoluciona el diseño. (Agrawal, Akhilesh, Chankyu, & kaushik, 2018)

Los ASIC y los FPGA (Torres, Perez, Jose, Zambrano, & Sepúlveda, 2017) tienen diferentes propuestas de valor y deben evaluarse cuidadosamente antes de elegir uno sobre el otro. Mientras que los FPGA solían seleccionarse para diseños de menor velocidad o complejidad, en la actualidad superan fácilmente la barrera del rendimiento de 500 MHz. Con aumentos de densidad lógica sin precedentes y una serie de otras características, como procesadores integrados, bloques DSP, reloj y serie de alta velocidad a precios cada vez más bajos, los FPGA son una propuesta atractiva para casi cualquier tipo de diseño (Roman, Calviz, Amoedo, & Romero, 2018).

Debido a su naturaleza programable, los FPGA son ideales para muchos mercados diferentes. Como líder de la industria, Xilinx ofrece soluciones integrales que consisten en dispositivos FPGA, software avanzado y núcleos IP configurables y listos para usar para mercados y aplicaciones (Risco, 2017).

Los FPGA también permiten un mayor grado de conectividad integrada debido a la fácil integración de protocolos IP de terceros para redes industriales, soporte de bus heredado y ruta a protocolos basados en Ethernet, junto con la capacidad de admitir múltiples protocolos en un solo dispositivo. En el control de motores, los ahorros en el costo total de propiedad se logran mediante la conservación de energía mediante un control más estricto de la velocidad, el par y la aceleración, mientras que la eficiencia mejorada permite motores más pequeños y menos costosos (Barrios Alfaro, 2017).

Los FPGA ofrecen soluciones particularmente poderosas para satisfacer las necesidades de visión artificial, redes industriales, control de motores y videovigilancia. Por ejemplo, la flexibilidad de los FPGA permite a los diseñadores adaptarse rápidamente a las cambiantes interfaces del sensor de imagen y los requisitos de procesamiento de imágenes, desarrollar

capacidades de análisis para mantenerse al día con los requisitos del mercado y agregar características y funciones mucho después de la implementación (Montoya Vásquez & Portilla, 2017).

2.2 Redes industriales

El área industrial es un lugar de trabajo cada vez más integrado que requiere interfaces para cruzar una amplia gama de aplicaciones, como controladores lógicos programables (PLC), módulos de E / S, motores, sensores, etc. Los protocolos de redes industriales proporcionan una comunicación fluida entre módulos, lo que permite componentes de diferentes fabricantes siempre que utilicen los mismos protocolos. Las comunicaciones de fábrica se pueden clasificar en tres niveles: el dispositivo, el proceso y los niveles de Ethernet.

El nivel de dispositivo proporciona comunicación entre módulos, como accionamientos de motor y sus sensores, y debe tener el menor tiempo de respuesta. El nivel de proceso es la comunicación de nivel medio entre PLC, que utiliza formatos peer-to-peer, que requiere un tiempo de respuesta corto pero que permite una latencia más alta en comparación con la comunicación a nivel de dispositivo

Finalmente, el nivel más alto de comunicación es el uso de Ethernet, que proporciona el mayor ancho de banda de datos y distancia para proporcionar comunicación entre varios sitios de fábrica. (Granado, Marín, & Pérez, 2010).

2.3 GAL22V10

El microprocesador GAL22V10, con un tiempo de retardo de propagación máximo de 4ns, combina un proceso CMOS (complementary metal-oxide-semiconductor) de alto rendimiento con borrable eléctrico (E2).

Tecnología de puerta flotante para proporcionar el mayor rendimiento disponible de cualquier dispositivo 22V10 en el mercado. Los circuitos CMOS permiten el GAL22V10 (ver Figura 1) para consumir mucha menos energía en comparación con dispositivos bipolares 22V10. E2 la tecnología ofrece alta velocidad (<100ms) borrar tiempos, proporcionando la capacidad de reprogramar o reconfigurar el dispositivo de forma rápida y eficiente.

La arquitectura genérica proporciona la máxima flexibilidad de diseño permitiendo que la macro célula lógica de salida (OLMC) sea configurada por el usuario. El GAL22V10 (ver figura 1) es totalmente funcional, compatible con dispositivos bipolares estándar y CMOS 22V10 (Quiles, Ortiz, Moreno, & Benavides Benítez).

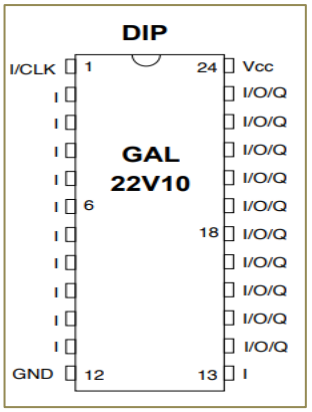

Figura 1. Integrado GAL22v10.

2.4 Máquina virtual

Usada para la aplicación de codificación de lenguaje assembler. Oracle VirtualBox (ver figura 2) es una aplicación de virtualización multiplataforma. Se instala en sus computadoras existentes basadas en Intel o AMD, ya sea que estén ejecutando los sistemas operativos Windows, Mac, Linux o Solaris. VirtualBox puede crear y ejecutar un sistema operativo "invitado" (máquina virtual) en una ventana del sistema operativo host. La máquina virtual proporciona un entorno autónomo en el que experimentar con nuevo software sin correr el riesgo de cambios dañinos en el sistema operativo del host (Panchi, Vinueza, & Manzaba, 2020).

Esta aplicación es principalmente requerida debido a que el programa compilador de VHDL (Very High Speed Integrated Circuit Hardware Description Language) no corre en sistemas operativos de alta gama, en la actualidad se utilizan otros tipos de compiladores mucho más avanzados, es la única aplicación disponible para el proyecto que se va a trabajar puesto que se usa un GAL22V10 y el compilador que se usara es el más recomendado para su programación.

El sistema operativo compatible con la aplicación es el Windows XP profesional con la arquitectura de 32 bits.

Figura 2. Sistema Operativo invitado instalado mediante Virtual Box.

2.5 Cypress Warp Galaxy

El compilador de síntesis Warp es un compilador VHDL de última generación para diseñar con PLD y CPLD. Warp utiliza un subconjunto de VHDL como lenguaje de descripción de hardware (HDL) para el diseño. Warp acepta la entrada de texto VHDL y luego sintetiza y optimiza el diseño para el hardware de destino. Warp luego genera un mapa jedec para programar PLD y CPLD, como se muestra en la Figura 3. El mapa jedec que produce Warp al apuntar a PLD y CPLD se puede utilizar para programar partes con un programador de dispositivos. El mapa también se puede utilizar como entrada para Nova ™ simulador funcional. Nova es un simulador gráfico interactivo que permite al usuario examinar el comportamiento de los diseños sintetizados (ACM, 2019).

2.6 Simulación

Una vez que se compile el código en la aplicación Cypress Warp Galaxy dentro de la máquina virtual que se esté trabajando se generaran los respectivos archivos para la simulación y diferentes herramientas que los requieran.

Vista del programa VHDL en el programa Cypress Warp Galaxy (ver figura 3) adentro de la máquina virtual con el sistema operativo Windows Xp profesional de 32 bits.

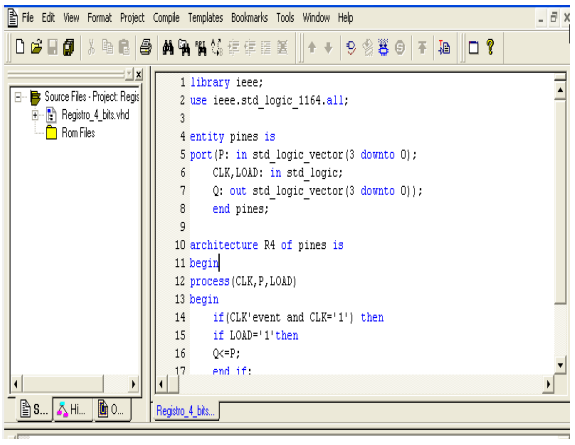

Figura 3. Compilador de síntesis Cypress Warp Galaxy

2.7 Código VHDL

A continuación, se muestra el código que permite obtener los niveles deseados de un tanque como entrada al microprocesador GAL22V10 realizado en el compilador Cypress Warp Galaxy (ver figura 4). Se detalla el código utilizado:

```
library ieee;
use ieee.std_logic_1164.all;

entity pines is
port(P: in std_logic_vector(3 downto 0);
    CLK,LOAD: in std_logic;
    Q: out std_logic_vector(3 downto 0));
    end pines;

architecture R4 of pines is
begin
process(CLK,P,LOAD)
begin
    if(CLK'event and CLK='1') then
    if LOAD='1'then
    Q<=P;
    end if;
    end if;
    end process;
    end R4;
```

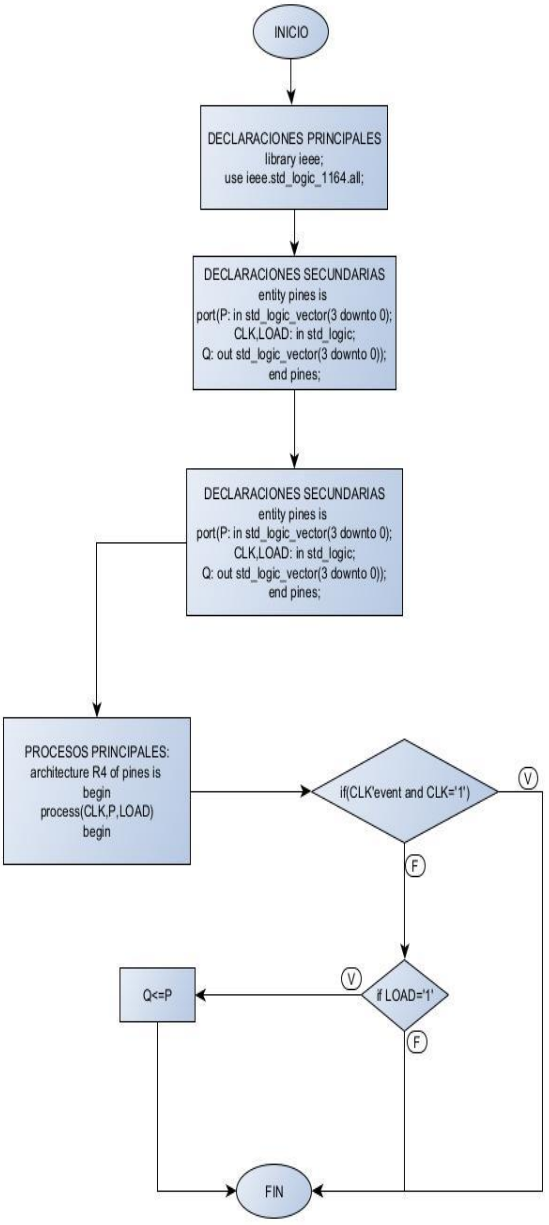

Figura 4. Diagrama de flujo del compilador Warp Galaxy

2.8 Archivos JED

El tipo de archivo jed está asociado principalmente con la especificación de programación jedec. En este caso la extensión es generada por la aplicación Cypress Warp Galaxy que es capaz de convertir la programación VHDL en un archivo de extensión jed para su posterior uso en simulador PROTEUS.

Antes de ser usado en el software PROTEUS es necesario que el archivo sea convertido a un archivo jedec que la aplicación reconozca ya que de no hacerlo dará inconvenientes a la hora de la simulación, esto se lo logra con la siguiente herramienta: "JEDEC convert to proteus".

Luego de ingresar el nombre del archivo jed generado a partir de la aplicación Cypress Warp Galaxy automáticamente tendremos el archivo con la extensión jed optimizada para que el programa PROTEUS lo reconozca y se pueda trabajar con este archivo en el FPGA (ver figura 5).

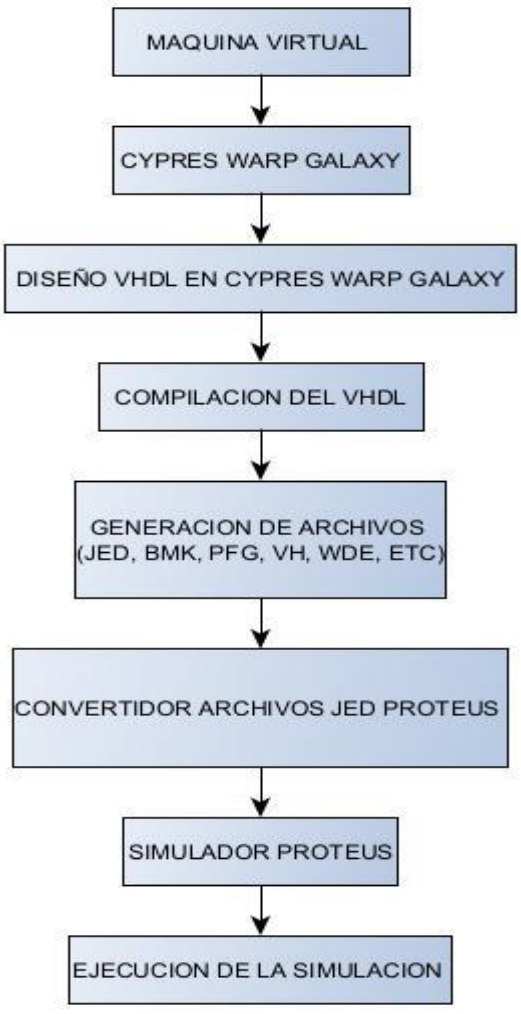

Figura 5. Diagrama de flujo del proceso de adaptación del archivo para PROTEUS.

Como ejemplo se tiene un diseño del sistema de tanques para simular el proceso de neutralización para siembra de algas:

Las microalgas marinas unicelulares se cultivan como alimento para las diferentes etapas del cultivo en criadero de moluscos de valor comercial. Hasta hace poco tiempo las algas vivas eran la única fuente de alimentación de las larvas y juveniles de bivalvos, pero esta situación está empezando a cambiar ahora como resultado de recientes investigaciones sobre el desarrollo de dietas artificiales e inertes apropiadas. Sin embargo, la producción de algas vivas va a seguir siendo un aspecto fundamental en el éxito de la gestión de criaderos en el futuro inmediato, aunque sólo sea como alimento vivo que complemente los alimentos más innovadores.

Simular un proceso de neutralización para mezclar agua, vitamina C y cloro para llegar a un punto de PH7, el óptimo para sembrar algas.

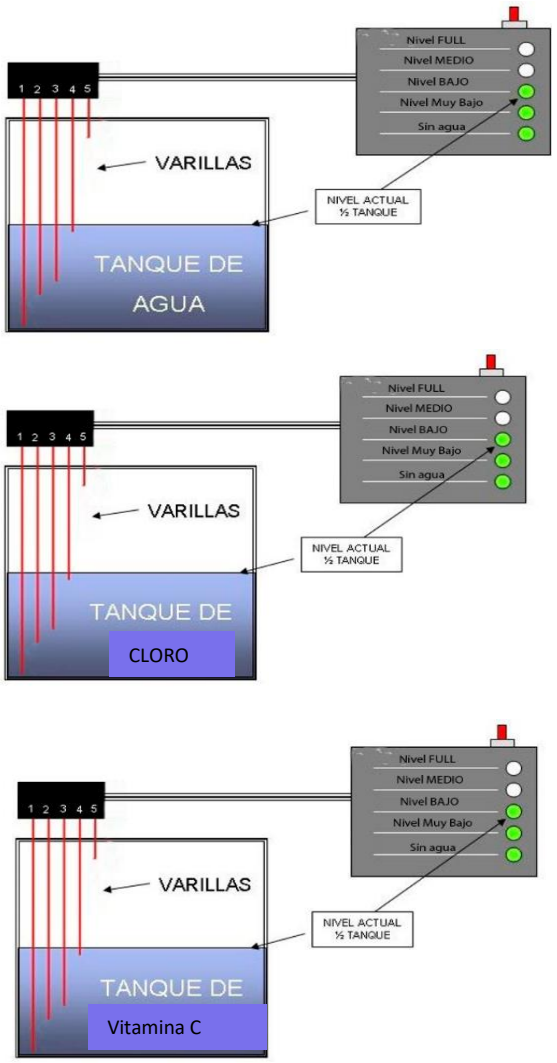

Figura 6. Tanques del ejercicio.

El proceso requerido para mezclar estos componentes es nivelar los tanques (ver figura 6) a igual escala seleccionando los niveles en las entradas al microprocesador, una vez bloqueado (ver figura 9) e iniciado el proceso no se deberían modificar los niveles de agua de cada uno de los tanques, cada nivel de tanque debe ser controlado por un FPGA.

Cuando los tres tanques se encuentren al mismo nivel se procederá con el proceso de mezclado normal (ver figura 7) dentro de otro tanque A donde quedaran los demás componentes, este proceso se logra haciendo girar un motor mediante la señal de un PIC, como señal de entrada al PIC debe ser cuando los tres tanques se encuentren al mismo nivel, mientras sean niveles diferentes en cada tanque no se procederá a mezclar. En caso de que se requiera niveles máximos en los tanques (nivel cuatro). Se procederá a ejecutar un mezclado especial en otro tanque B (ver figura 8) que requiere que el giro del motor sea a menos revoluciones por minuto durante el proceso.

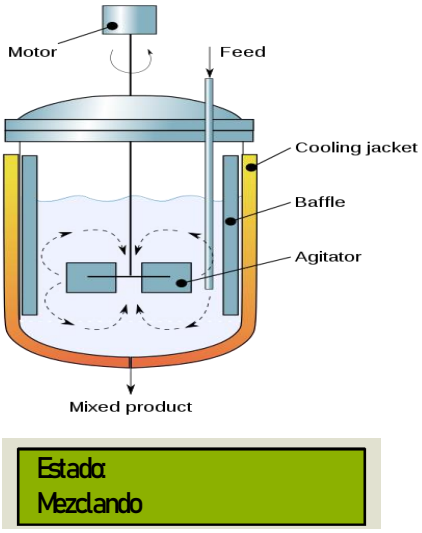

Figura 7. Mezcla normal tanque A

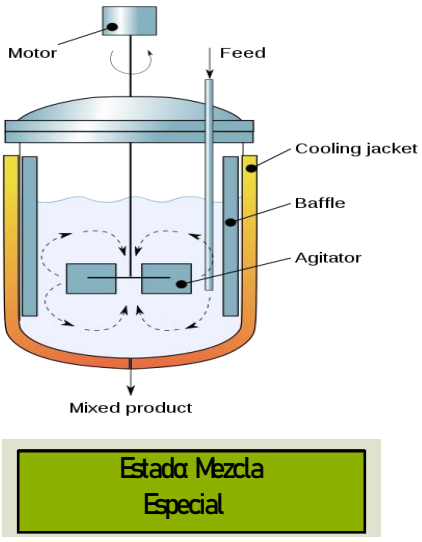

Figura 8. Mezcla especial tanque B

Se deberá diseñar el código VHDL del FPGA para cada uno de los tanques y montado con su respectivo archivo JED dentro del dispositivo en el simulador Proteus, para el control de llenado se requiere un circuito digital que obtenga como salidas los niveles que se escogieron en el momento de selección con el FPGA.

Figura 9. Niveles de los Tanques

2.9 Proteus

PROTEUS es una aplicación de simulación de circuitos y modelado de sistemas virtuales. Combina simulación de circuito SPICE de modo mixto, componentes animados y modelos de microprocesador para facilitar la co-simulación de diseños completos basados en microcontroladores. PROTEUS también tiene la capacidad de simular la interacción entre el software que se ejecuta en un microcontrolador y cualquier electrónica analógica o digital conectada.

Simula puertos de entrada/salida, interrupciones, temporizadores, USART y todos los demás periféricos presentes en cada procesador compatible.

El software que se utilizó no es el completo de PROTEUS, es una versión de demostración. Sin embargo, se puede escribir programas propios del software para que se ejecuten en el conjunto de diseños de muestra existente con fines de evaluación del proyecto que se requiere y contiene entre sus librerías el GAL22V10 con sus respectivas configuraciones para importar el archivo jed que se va a usar para lo mencionado.

Se tiene un FPGA para cada tanque, este a su vez será capaz de controlar como entradas las combinaciones manuales de niveles, como se puede observar en la figura 6, mientras el pin 5 (ver figura 10) se encuentre con el valor booleano uno lógico se pueden realizar las combinaciones, y cuando se encuentra en cero es para indicar que no se podrán hacer más modificaciones mientras se encuentre en el proceso de mezclado. Luego se procederá a iniciar el llenado de cada tanque.

Cada salida va a ser manipulada por el microprocesador GAL22V10 según sus condiciones de entrada

establecidas por el usuario. Cabe recalcar que habrá un pin habilitador para estas entradas el cual será el pin I5

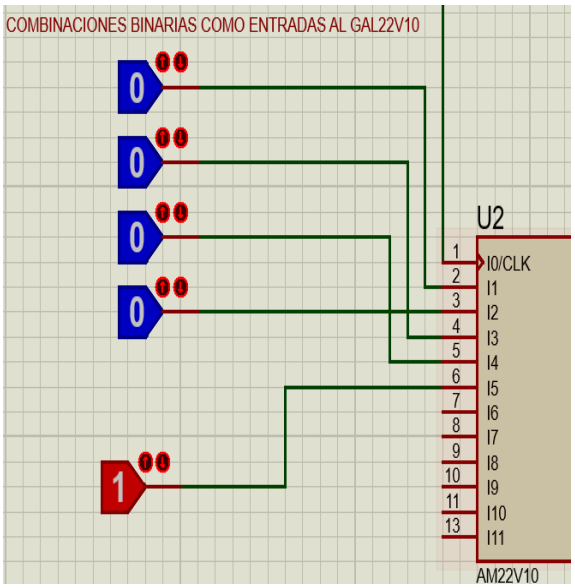

Figura 10: Entradas al GAL22v10

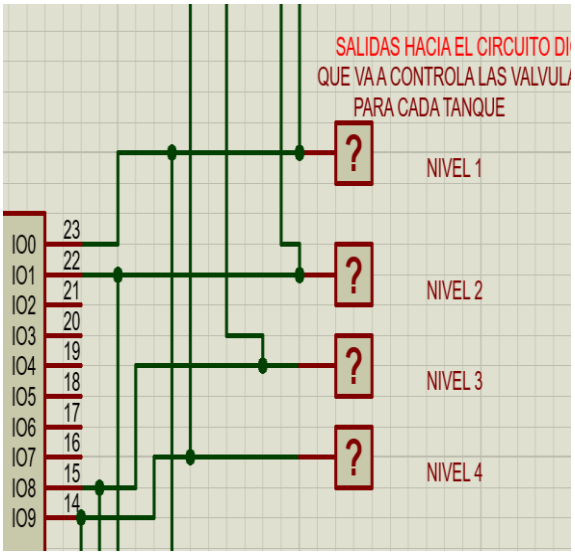

Figura 11. Salidas del FPGA

Después de procesar las entradas el FPGA dará como salidas la señal de acuerdo al nivel de agua (ver figura 11).

3. Resultados

3.1 Simulación

El microprocesador GAL22V10 con su código responde a la siguiente tabla junto con sus pines de entradas y salidas, para mejor comprensión se expresa como ALTO el número 1 y como BAJO el número 0. Se puede establecer durante la simulación el nivel con el que se requiere trabajar siempre y cuando el pin de entrada I5 este en ALTO.

Esta es la manera más fácil de poder utilizar este microprocesador ya que el tiempo de respuesta es inmediato y su programación no es tan complicada en lenguaje VHDL.

ENTRADAS					SALIDAS			
I1	I2	I3	I4	I5	I00	I01	I08	I09
0	0	0	1	1	1	0	0	0
0	0	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1

La entrada I5 está en ALTO (ver figura 12) lo que significa que cualquier cambio puede darse entre las entradas I1 a I4, las cuales son las entradas del nivel de tanque, para este caso tenemos a nivel 3

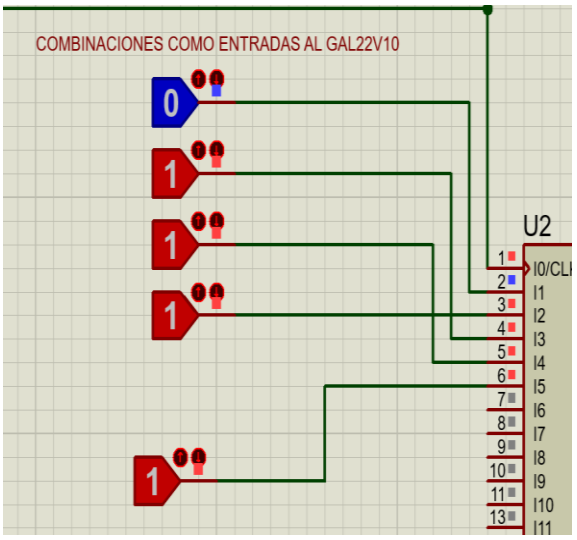

Figura 12. GAL22V10 en ejecución.

Una vez seleccionado el nivel de tanque de deberá poner la entrada I5 en BAJO para que ningún valor de las entradas al FPGA pueda modificarse en el transcurso del proceso que utiliza un FPGA para el control de llenado de cada tanque. Se usó el PIC16f887 de tal manera que la entrada al pin RA2 analógico se detecte que el proceso de mezcla de inicio y se mostrará por un LCD el proceso en curso. Durante la ejecución del proceso no se presentó ningún error por parte del microprocesador GAlV22V10. Esto indica que el microprocesador está cumpliendo con las condiciones indicadas por el ejercicio propuesto.

Se pone en ejecución el proceso para la obtención de salidas al Circuito digital de control cuyo propósito es la activación del motor de la mezcladora siempre y cuando los tres tanques estén al mismo nivel (ver figura 13), cada tanque es controlado por un FPGA

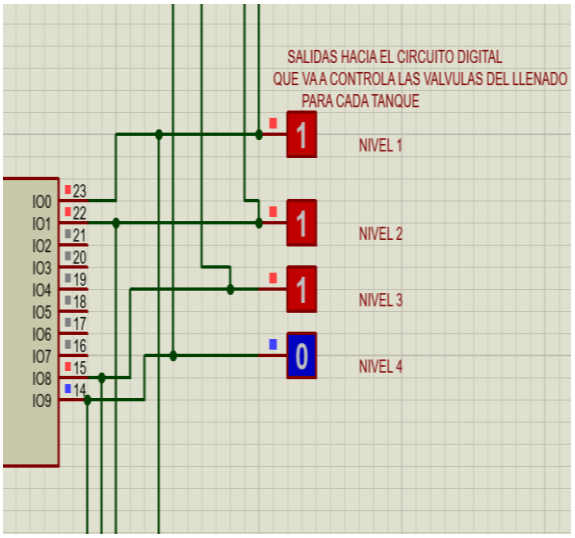

Figura 13. Salidas del FPGA en ejecución.

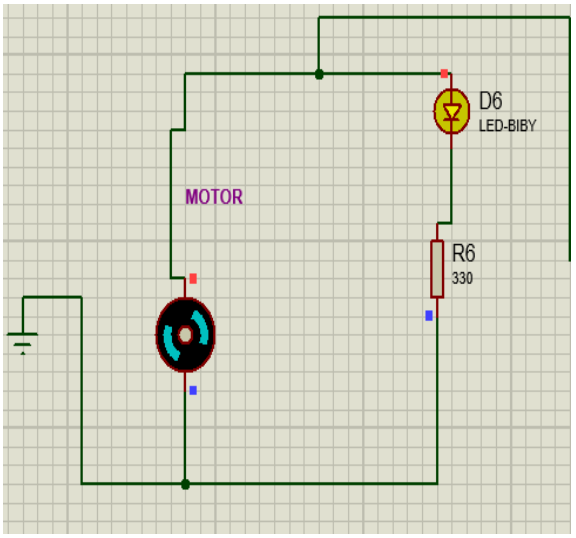

Figura 14: Estado del motor encendido

Se tiene un circuito digital de control cuyo propósito es la activación del motor de la mezcladora siempre y cuando los tres tanques estén al mismo nivel, cada tanque es controlado por un FPGA, el tipo de mezcla se visualizará en un LCD.

El motor de mezcla de componentes comenzará a girar de acuerdo con los niveles del tanque, si los niveles de los tanques se encuentran hasta el tercer nivel se procederá a realizar una mezcla normal con el giro de motor a una velocidad estándar. Y si los tres tanques se encuentran en su máximo nivel, es decir el nivel cuatro se procederá a realizar una mezcla especial haciendo que el motor gire (ver figura 14) a una velocidad con menos revoluciones por minuto que la estándar. Se configura el reloj del FPGA a una frecuencia de 60HZ (ver figura 15).

Se comprobó cada uno de los casos para cada nivel de tanque y los resultados obtenidos fueron satisfactorios y correspondientes a la tabla de entradas y salidas del microprocesador GAL22V10, de esta forma se cumplió con el objetivo de este proyecto el cual consistía en virtualizar este tipo de circuito integrado en el PROTEUS. Se lo pudo simular lo más cercano a la realidad posible utilizando una frecuencia en un rango de 60 HZ a 100 HZ.

Con los resultados obtenidos se puede mencionar que PROTEUS trabaja muy bien con este tipo de diseño ya que se requirió programas alternos como Cypress Warp Galaxy para poder codificar el GAL22V10.

Se usa mucho este tipo de simulaciones para luego realizar implementaciones físicas esto con la finalidad de comprobar errores que puede haber al momento de la ejecución de los distintos procesos para los que fueron diseñados, en este caso no hubo ningún error al momento de ejecución ni con el microprocesador GAL22V10 ni con los demás elementos vinculados dentro del circuito.

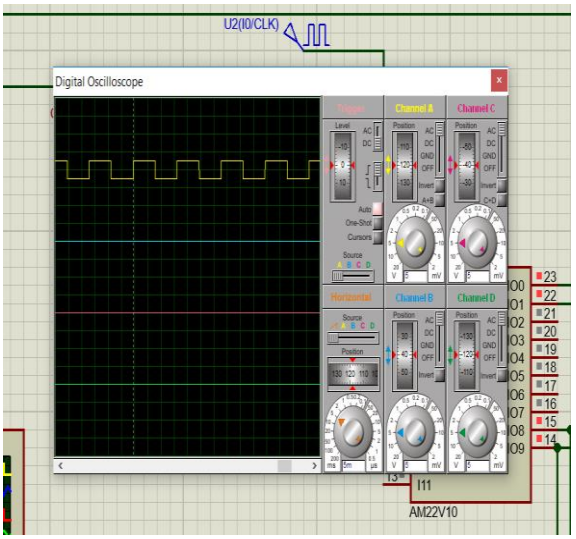

Figura 15. Reloj del FPGA

4. Conclusiones

Se logró virtualizar el microprocesador GAL22V10 para controlar el nivel de cada uno de los 3 tanques. El FPGA trabajó sin problemas con un reloj a una frecuencia de 100 Hz. El tiempo de respuesta de los pines salidas del FPGA fue instantáneo, logrando un rendimiento óptimo en el proceso deseado. El circuito principal fue diseñado para mostrar el funcionamiento del GAL22V10 en la herramienta de software Proteus con circuitos digitales básicos.

El dispositivo GAL22V10 sirve como ejemplo de lo que pueden hacer los FPGA en la industria. Se comprobó el comportamiento del dispositivo trabajando a 60 HZ.

5. Referencias bibliográficas

[1] ACM. (junio de 2019). ACM DIGITAL LIBRARY. Recuperado el 30 de marzo de 2021, de ACM DIGITAL LIBRARY.

[2] Agrawal, A., Akhilesh, J., Chankyu, L., & kaushik, R. (Diciembre de 2018). https://ieeexplore.ieee.org. Recuperado el 30 de marzo de 2021, de https://ieeexplore.ieee.org.

[3] Barrios Alfaro, Y. (2017). AccedaCRIS.ulpgc. Recuperado el 30 de marzo de 2021, de AccedaCRIS.ulpgc.

[4] Chavez, E., & Vasquez , A. (2017). http://congresos.cio.mx/. Obtenido de http://congresos.cio.mx/.

[5] Garcia , N., & Valdes, P. (2018). https://www.researchgate.net/. Obtenido de https://www.researchgate.net/.

[6] Granado, E., Marín, W., & Pérez, O. (marzo de 2010). Scielo. Recuperado el 30 de marzo de 2021, de Scielo.

[7] Mogollón Baca, F. (25 de junio de 2019). Repositorio Dspace. Recuperado el 30 de marzo de 2021, de Repositorio Dspace.

[8] Montoya Vásquez, D., & Portilla, J. (2017). http://profesores.elo.utfsm.cl. Recuperado el 30 de marzo de 2021, de http://profesores.elo.utfsm.cl.

[9] Panchi, G., Vinueza, M., & Manzaba, E. (mayo de 2020). https://search.proquest.com. Recuperado el 30 de marzo de 2021, de https://search.proquest.com.

[10] Quiles, F. J., Ortiz, M. A., Moreno, C., & Benavides Benítez, J. I. (s.f.). researchgate.net. Recuperado el 30 de marzo de 2021, de researchgate.net.

[11] Risco, M. (2017). https://www.researchgate.net. Obtenido de https://www.researchgate.net.

[12] Roman, C., Calviz, M., Amoedo, P., & Romero, M. (2018). https://www.aadeca.org. Obtenido de https://www.aadeca.org.

[13] Sosa , J., Garcia, H., Salinas, E., Ortega, R., & Hernandez, R. (2018). http://www.scielo.org.mx/. Recuperado el 30 de marzo de 2021, de http://www.scielo.org.mx/.

[14] Torres, R., Perez, F., Jose, B., Zambrano, L., & Sepúlveda, R. (2017). https://www.researchgate.net. Obtenido de https://www.researchgate.net.

[15] Vargas, L., Cogua, D., & Rincón, B. (2017). https://revistas.itc.edu.co. Obtenido de https://revistas.itc.edu.co.