EL PROBLEMA SAT: ENFOQUE COMPARATIVO CON ADN Y FPGA

J. Sepúlveda, C. Camargo[+], A. Delgado[+]*

*Estudiante de Ingeniería Electrónica
Universidad Nacional de Colombia, Bogotá

[+]Profesores Ingeniería Electrónica
Universidad Nacional de Colombia, Bogotá

marthase@presidency.com

ABSTRACT

The problem of Boolean Satisfiability or SAT, has been considered as benchmark by the molecular computing community. In this paper we prove that the electronics solution, using a FPGA (Field Programmable Gate Array), is faster and easier to implement than the present chemical solution, using electrophoresis and PCR, that could take several hours to complete.

RESUMEN

El problema de satisfacción Booleana o SAT, ha sido considerado un modelo por la comunidad del cómputo molecular. En éste artículo probamos que la solución electrónica, usando un FPGA (arreglo de compuertas programables por campo), es más rápida y fácil de implementar que la solución química actual, que emplea electroforesis y PCR, que puede tardar horas en completarse.

EL PROBLEMA SAT: ENFOQUE COMPARATIVO CON ADN Y FPGA

J. Sepúlveda, C. Camargo[+], A. Delgado[+]*

*Estudiante de Ingeniería Electrónica
Universidad Nacional de Colombia, Bogotá

[+]Profesores Ingeniería Electrónica
Universidad Nacional de Colombia, Bogotá

RESUMEN

El problema de satisfacción Booleana o SAT, ha sido considerado un modelo por la comunidad del cómputo molecular. En éste artículo probamos que la solución electrónica, usando un FPGA (arreglo de compuertas programables por campo), es más rápida y fácil de implementar que la solución química actual, que emplea electroforesis y PCR, que puede tardar horas en completarse.

1. INTRODUCCIÓN

El problema de satisfacción Booleana consta de un conjunto finito de variables, donde se debe encontrar el valor de éstas para que la salida de una función Booleana, expresada como producto de cláusulas (suma de variables booleanas), sea igual a uno [1]. El dominio de este problema es finito, lo cual lo convierte en un problema de búsqueda intensiva muy interesante que se ha trabajado desde antes de los años sesenta. Para esto se han propuesto diversas soluciones desde varias áreas del saber: la estadística, la matemática hasta las áreas que tienen que ver con la computación y la electrónica.

Recientemente se ha propuesto un computador molecular para evaluar de manera iterativa el problema SAT [2]. En esta aplicación la evaluación de cada cláusula es una iteración e involucra dos procesos químicos que se emplean para manipular el ADN, electroforesis y PCR, que en la actualidad pueden tener una duración aproximada de 4 horas[2].

La computación molecular con ADN soluciona el problema SAT pero el proceso es lento y complejo, lo cual no lo hace apto para una aplicación real, ya que existen mejores herramientas como los FPGAs, que también cuentan con un alto grado de paralelismo y una mayor y más rápida forma de Interpretación de resultados.

Se empleó el lenguaje de descripción de Hardware VHDL, para solucionar el mismo problema resuelto por medio de computación molecular por Sakamoto et al [3]; aunque el modelo desarrollado es flexible y se puede usar para un mayor número de variables.

El artículo se divide en tres secciones. La primera sección describe el problema SAT. En la segunda sección se hace un breve resumen sobre los principios de la computación molecular con ADN y las soluciones que con ésta técnica se han planteado para resolver el problema SAT. La tercera sección se muestra nuestra solución implementada en un FPGA. Por último se generan las conclusiones.

2. PROBLEMA SAT

El problema de satisfacción booleana es un problema de búsqueda intensiva, que consiste en encontrar el valor de unas variables que hacen que el valor de una función booleana sea igual a uno.

Considere tres variables o letras booleanas X_1 , X_2 , X_3, las cuales se emplean para describir una función F en forma de producto de cláusulas, es decir como producto de sumas de letras y sus expresiones negadas:

$F(X_1 , X_2 , X_3)= (X_1 + \overline{X}_2).(\overline{X}_1 + X_3).(X_2 + \overline{X}_3)$

Donde F esta expresada como producto de tres cláusulas. Para resolver esta ecuación, el valor de las cláusulas debe ser uno, es así como se deduce que $X_1=X_2=X_3=1$.

La solución de esta función es fácil, pero cuando el número de variables aumenta, el número de cláusulas también lo hace, haciendo que el problema se vuelva complejo.

El problema SAT que resolveremos por medio de un FPGA consta de seis variables y un total de diez cláusulas.

F(a,b,c,d,e,f,) = **(**a OR b OR (NOT c)**)** AND **(**a OR c OR d**)** AND **(**a OR (NOT c)OR (NOT d)**)** AND **(**(NOT a) OR (NOT c) OR d)**)** AND **(**a OR (NOT c) OR e**)** AND **(**a OR d OR (NOT f)**)** AND **(**(NOT a) OR c OR d**)** AND **(**a OR c OR (NOT d)**)** AND **(**(NOT a) OR (NOT c) OR (NOT d)**)** AND **(**(NOT a) OR c OR (NOT d)**).**

Donde se encontró una única solución (a,b,c,d,e,f)=(0,1,1,0,1,0).

Este problema fue resuelto por Sakamoto et al [3] empleando Computación Molecular con ADN.

3. SOLUCIÓN DEL SAT POR COMPUTACION MOLECULAR CON ADN

En las últimas décadas se han venido desarrollando diferentes herramientas que tratan de resolver o hacer más fácil encontrar la respuesta a algunos problemas de gran complejidad. Es así como se le abre paso a una nueva ciencia llamada "Computación Molecular con ADN ". El término apareció por primera vez en 1994 cuando Leonard Adleman reportó la construcción de un "Computador de ADN" [4], en el cual pudo resolver el problema del agente viajero usando técnicas de recombinación del ADN.

3.1. Principios de la computación molecular

El ADN o ácido desoxirribonucléico [5] es el código básico de todos los seres vivos, se encuentra en el núcleo de la célula y determina su forma y función. Esta macromolécula tiene una estructura de doble hélice similar a una escalera retorcida, donde sus lados son cadenas de segmentos de fosfatos y un azúcar de cinco carbonos denominada desoxirribosa, los travesaños llevan la información genética y están formados por cuatro bases: Adenina (A), Tiamina (T), Guanina (G) y Citosina (C). Cada travesaño esta conformado por dos bases: Adenina - Tiamina o Guanina - Citosina.

Se denomina nucleótido a una subunidad de la molécula de ADN conformado por un fosfato, un azúcar y una de las cuatro bases. La secuencia de estos nucleótidos, determina el código genético único de cada individuo.
El ADN puede replicarse, este proceso se lleva a cabo antes de la división celular. Cuando esto sucede, la escalera se divide entre sus bases (es decir los travesaños de la escalera retorcida se dividen en dos), lo cual hace que las bases puedan atraer nucleótidos libres, adhiriédolos a la escalera. Este proceso genera la formación de dos cadenas de ADN idénticas.

Además si las cadenas son complementarias, se pueden crear enlaces a través de los fosfatos o los azúcares, formando cadenas de ADN mucho más largas. En el laboratorio, se pueden crear cadenas 1000 a 5000 letras de longitud, donde cada letra es la unión entre dos bases, lo cual genera 4^L posibilidades de combinación [5] (donde L es el número de letras empleadas). Entre menos letras conformen una cadena, menor probabilidad de error habrá.

3.1.1 Formación de las cadenas de ADN

Existen varios procesos realizados en el laboratorio, para llevar a cabo la formación de una doble hélice de ADN, de esta forma se permite manipular y codificar los problemas que se van a resolver.

Proceso de Hibridación: Se seleccionan dos cadenas sencillas, que posean bases compatibles para la unión, y se garantiza una temperatura, la cual debe permanecer constante en todo el proceso. En la etapa de diseño, se busca que el número de uniones indeseadas sea mínima.

Método de purificación biotín-avidín: Se fija una molécula biotín-avidín, la cual se une a la cadena deseada, atrayendo a otras moléculas que generan que esta gran cadena de nucleótidos se ancle en el fondo del tubo de ensayo. Se realiza un lavado con agua para deshacerse de las cadenas indeseadas para luego calentar la mezcla por encima de la temperatura de fusión,

separando así la cadena deseada. Para obtener una secuencia de 20 letras, se requiere de 800 minutos.

Amplificación por PCR: Es un proceso cíclico, el cual usa la reacción en cadena de la polimerasa para realizar copias de una cadena de ADN. En n pasos se crea 2^n copias de una cadena doble. Consiste en la separación de cadenas por medio del empleo de la temperatura de fusión, la cual varía según la longitud de la cadena. Este proceso también es llamado desnaturalización

3.2 Ventajas y desventajas de la computación molecular por ADN

La ventaja más importante de éste tipo de computación se encuentra en su alto grado de paralelismo, ya que los procedimientos químicos, como el de hibridación, se realizan al mismo tiempo. Por otra parte el proceso de replicación cuenta con un porcentaje bajo de error, lo cual permite obtener cientos de cadenas idénticas de un mismo patrón.

Otro aspecto importante radica en las dimensiones de las cadenas de ADN, unas cuantas micras de largo por algunos nanómetros de ancho. En tres gramos de agua se pueden encontrar 10^{22} oligos, es decir cadenas cortas de ADN (20 a 40 letras)[5]

Como desventaja se encuentra el hecho de que la formación de cadenas de ADN para la codificación de un problema, es un proceso lento que depende del número de letras que se deseen generar, por ejemplo el proceso de hibridación, aunque paralelo tarda más de cuatro horas en completarse. Además éste tipo de computación no posee un protocolo para la codificación, es decir depende plenamente del autor.

3.3. El Problema SAT y La Computación Molecular

Un primer acercamiento a la solución de éste tipo de problemas por medio de la computación molecular por ADN lo propuso Lipton [6]. Consideremos la función a resolver $F(X_1,X_2)$. Para tratar el problema, se procedió a codificar las variables en secuencias de ADN de 20 nucleótidos de largo, luego se mezclaban estas cadenas en tubos de ensayo con las secuencias de ADN complementarias a las codificadas. Las cadenas dobles resultantes contenían las variables, X_1 y X_2, codificadas con algún valor, es decir representaban una cadena de unos y ceros que conformaban los valores asignados a las variables de la función del problema SAT.

Después se iban separando las cadenas resultantes que poseían un mismo código o valor para una misma variable, es decir todas las cadenas que asignaban un uno a la variable X_1 se colocaban en otro tubo de ensayo y se dejaban aquellas que asignaban cero a esta variable.
Luego se mezclaban las soluciones según las cláusulas que conforman la función que se estaba evaluando. Si las soluciones no eran satisfactorias se desechaban, hasta encontrar la respuesta.

Otra solución para este problema, empleando computación molecular fue propuesta por Sakamoto et al, donde se enfocan más en la formación de hélices de ADN por las moléculas que representan la función. Este algoritmo disminuye el número de pasos requeridos en la computación, lo cual a su vez reduce el error por la introducción de cambios indeseados al proceso. El procedimiento seguido para la resolución del problema fue: Primero se generan las cadenas de literales que representan la función (los cuales surgen por la combinación entre cláusulas), es decir se codifica el problema, luego se permite la combinación del ADN que representa las cadenas de literales para formar hélices por el proceso de hibridación. Este paso se controla por medio de la regulación de la temperatura, haciendo que las enzimas que eran necesarias en la implementación de [6] ya no se utilicen. Por último se remueven las hélices de ADN, dejando las soluciones del problema SAT; siendo este paso el de mayor complejidad [3].

La función implementada por Sakamoto et al [3] fue la misma propuesta en la sección 2 del artículo, la cual constaba de seis variables y diez cláusulas.

Para resolver el problema, se implementó un total de 30 literales, lo cual implica un largo tiempo empleado en la generación de las cadenas de ADN, sin mencionar los procesos de control de temperatura llevados a cabo para la ligación de cadenas, ya que en el método de [3], no se emplearon enzimas. En general la velocidad de la computación molecular depende de dos factores: El grado de paralelismo y el número de pasos que se deben implementar [6], pero además se debe tener en cuenta el tiempo para producir cada literal y el de reacción.

Este algoritmo presenta una gran ineficiencia, no solo en tiempo sino con respecto a la cantidad de ADN requerido pues en la propuesta de [6] se necesitan 2^n moléculas para codificar las variables, pero en el algoritmo de [3] se deben generar 3^m literales para m cláusulas, donde m puede llegar a ser cuatro veces n, debido a la representación especial que se emplea en esta solución.

Al final, se debe visualizar la respuesta, utilizando un método basado en rayos X para determinar su valor.

Se decía que este tipo de soluciones eran mucho mejores que las encontradas por medio de una computación normal secuencial, debido a que el proceso llevado a cabo por el método del ADN era paralelo en su totalidad, pero se han venido desarrollando otras técnicas de resolución de problemas de búsqueda intensiva como las propuestas en sistemas configurables como el FPGA, donde se tiene la posibilidad de procesar en paralelo o en forma secuencial, según convenga, lo cual lo hace muy veloz para diferentes aplicaciones.

Otra característica importante de los FPGAs es su alta versatilidad, que permite reprogramarlos muy rápidamente empleando un lenguaje sencillo como lo es VHDL. Además el resultado arrojado es de muy fácil interpretación.

4. SOLUCIÓN POR MEDIO DE FPGA

Los FPGAs (Field Programmable Gate Arrays) son dispositivos que consisten en un arreglo de compuertas lógicas configurables. Su programación es sencilla y se realiza mediante un lenguaje de alto nivel (VHDL, Verilog).
Este dispositivo puede operar en forma secuencial, paralela o una combinación de las dos.

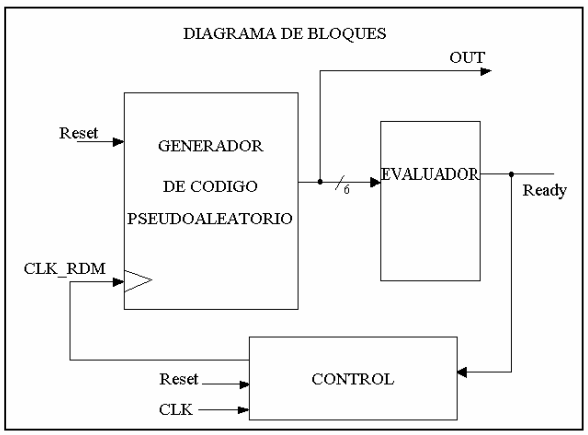


Figura 1. Diagrama de bloques de la solución para resolver el problema SAT

Los FPGAs tienen tres ventajas muy importantes: 1) La disponibilidad del hardware durante el proceso de diseño, 2) La versatilidad del componente debido a sus formas de operación y número de compuertas lógicas con las que se cuenta y 3) La visibilidad, ya que el resultado se puede apreciar con gran facilidad [7].

Debido a estas características, los FPGAs se convierten en una herramienta muy útil para resolver problemas como los de búsqueda intensiva, tipo SAT.

Se han publicado diversos artículos que implementan algoritmos para la solución del problema SAT en FPGAs. Zhong et al [8] implementaron el algoritmo de Davis-Putnam [1] en el cual se evalúa la consistencia de la función, es decir si existe un conjunto de valores que satisfacen la función.

4.1 Planteamiento de la Solución

El algoritmo que presentamos (Figura.1), se implemento en el XC2S200 de Xilinx, el cual soluciona el problema planteado por Sakamoto et al [3] y que se resolvió empleando computación molecular por ADN. El objetivo es poder comparar la eficiencia y desempeño de los FPGAs frente a éste tipo de computación para un problema de búsqueda intensiva. Nuestra solución emplea un generador controlado de números pseudoaleatorios de seis bits, los cuales constituyen los posibles valores de las variables a, b, c, d, e y f que hacen que la función F(a, b, c, d, e, f) sea igual a uno. De esta forma se evalúa la función paralelamente y se entrega un resultado. Si el valor obtenido es satisfactorio, es decir es igual a uno, se habilitan seis salidas del FPGA para que se conozca el valor correcto de las variables. En seguida se explicará el algoritmo detalladamente.

4.1.1 Bloque generador pseudoaleatorio

Este bloque consiste en un generador pseudoaleatorio, el cual posee 2^6 posibles combinaciones de las variables a, b, c, d, e, f. Posee dos señales de control: CLK_rdm. Cada vez que se produce un flanco de subida en esta señal se generará un nuevo valor, el cual será evaluado para probar si corresponde a la solución de la función y RESET.

El generador aleatorio esta basado en el algoritmo de [9], el cual propone un registro de desplazamiento conectado en forma anillo.

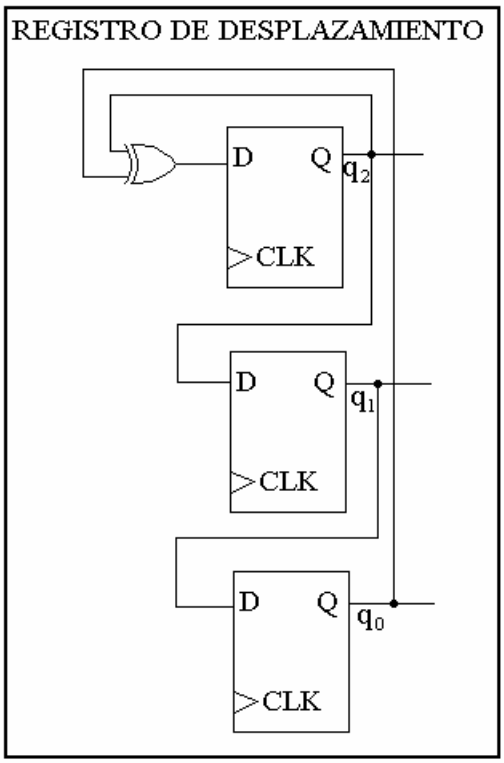
La entrada de estos registros esta dada por la ecuación de recurrencia:

$$X_n=(A_1 \textbf{ AND } X_{n-1}) \textbf{ XOR } (A_2 \textbf{ AND } X_{n-2}) \ldots \textbf{ XOR } (A_m \textbf{ AND } X_{n-m})$$

Donde la combinación de los productos de las salidas de otros registros con los coeficientes A_n, generan el nivel de entrada de un registro como se observa en la Figura 2. En esta ecuación se emplea la función XOR para incrementar la probabilidad de aparición de unos lógicos en la secuencia de números pseudoaleatorios.

En forma de vectores se puede expresar como

$$q (i+1) = A. q(i)$$



Fiugura 2. Registro de desplazamiento conectado en anillo

q representa las salidas de los diferentes registros que están conectados en la configuración anillo y A es la matriz de transferencia de un registro de desplazamiento de configuración anillo.
Es decir que para nuestro generador de seis posiciones tenemos:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

El bloque generador de números pseudoaleatorios, consta de seis registros de desplazamiento para representar las variables a, b, c, d, e y f. La matriz A de 6x6 se modifica para obtener la matriz de coeficientes, los cuales determinan el número de saltos que se realizarán en el generador pseudoaleatorio. Esta nueva matriz se obtiene elevando a alguna potencia la matriz A. En nuestro ejemplo la elevamos al cuadrado, generando la nueva matriz A de coeficientes.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

En un generador pseudoaleatorio, se repite un patrón, después de un número de ciclos, conocido como periodo del generador.

A continuación se muestra un segmento del código en VHDL que implementa la matriz A en el FPGA. El vector Rnd_Out constituye la salida de los registros que conforman la conexión anillo.

```
Process ( Clk, reset )
 Begin
  If Reset = '1' then
    Rnd_out <= "000001";
    elsif Clk'event and clk = '1' then

        Rnd_Out(0) <= Rnd_Out(2);
        Rnd_Out(1) <= Rnd_Out(3);
        Rnd_Out(2) <= Rnd_Out(4);
        Rnd_Out(3) <= Rnd_Out(5);
        Rnd_Out(4) <= Rnd_Out(0) xor Rnd_Out(5);
        Rnd_Out(5) <= Rnd_Out(0) xor Rnd_Out(1) xor Rnd_out(5);

     end if;
 end process;
```

Las variables que conforman el vector de salida Rnd_Out cambian durante cada ciclo de reloj y se observó en la implementación que el índice de aparición de números repetidos hasta que se encuentra la respuesta es muy bajo.

4.1.2 Bloque de Control

Es el encargado de controlar al generador de números pseudoaleatorios; cada vez que se presenta un flanco en la señal CLK_RDM se genera un número.

El control a su vez también depende del estado de tres señales externas: CLK (oscilador de cristal), RESET (pulsador) y ENABLE, como se observa en la Figura 3, el cual debe tener un valor lógico ALTO para que el sistema comience su operación.

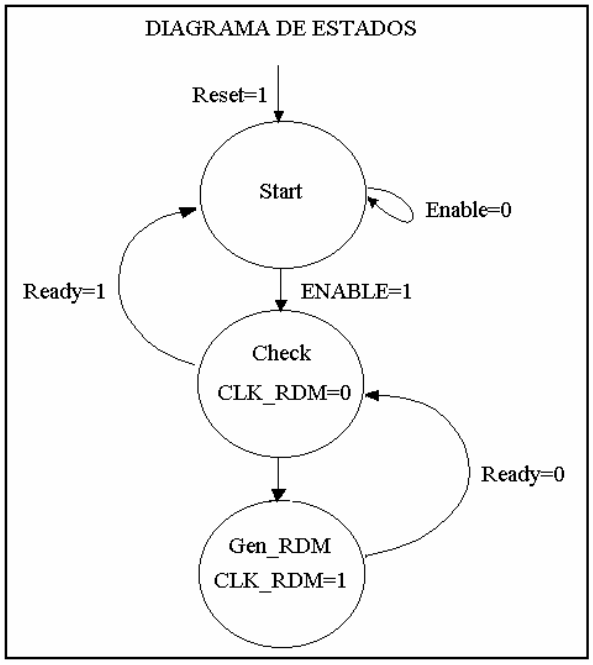


Figura. 3 Máquina de estado empleada para el bloque Control.

El control, esta habilitado a su vez por una señal interna proveniente del bloque que evalúa la función (Ready). Una vez que se encuentre el resultado, es decir que F sea igual a uno, esta señal se encarga de deshabilitar el control, de tal forma que no se sigan produciendo números pseudoaleatorios.

Para la programación de éste bloque se empleo una máquina de estado (Figura 3), la cual varía con cada flanco de subida del reloj u oscilador.

4.1.3 Bloque evaluador

Este Bloque contiene la función que se va a evaluar en forma de producto de cláusulas, explicadas anteriormente.

El evaluador tiene como entradas los seis bits previamente generados (vector E) donde C son las cláusulas programadas y F es la función que debe ser evaluada.

```
Process (E )
 Begin

      C0 <= ((E(0) OR E(1) OR (NOT E(2))) ;
      C1 <= ((E(0) OR E(2) OR E(3));
      C2 <= ((E(0) OR (NOT E(2)) OR (NOT E(3)));
      C3 <= (((NOT E(0)) OR (NOT (E2)) OR E(3)));
      C4 <= ((E(0)  OR (NOT E(2)) OR E(4));
      C5 <= ((E(0) OR E(3) OR (NOT (E5)));
      C6 <= (((NOT E(0)) OR E(2) OR E(3));
      C7 <= ((E(0) OR E(2) OR (NOT E(3)));
      C8 <= (((NOT E(0)) OR (NOT E(2)) OR (NOT E(3)));
      C9 <= (((NOT E(0)) OR E(2) OR (NOT E(3)));

F = C0 AND C1 AND C2 AND C3  AND C4 AND C5 AND C6
AND C7 AND C8 AND C9;
end process;
```

De forma paralela se procesa y se encuentra la información, de tal manera que si los valores supuestos son satisfactorios, el evaluador hace posible que se éstos se conozcan, direccionándolas a unos pines de salida establecidos. Si esto no ocurre, el bloque no realiza ningún cambio y el control se encarga de que el generador continúe enviando nuevos datos.

4.2 Resultados

El algoritmo fue implementado en el XC2S200 de Xilinx, con el cual se obtuvo que F(a, b, c, d, e, f) es igual a uno cuando (a,b,c,d,e,f)=(0,1,1,0,1,0), encontrándose esta solución en un tiempo de 1.568 milisegundos con un reloj cuyo periodo es de 32 microsegundos, para un total de 49 ciclos de reloj.

En la Figura. 4 se muestra el resultado de la prueba del circuito. La señal RESULT equivale al valor obtenido a la salida del bloque del generador de números pseudoaleatorios, ENABLE, señal activa en ALTO, RESET, activo en ALTO y el CLK o señal de reloj.

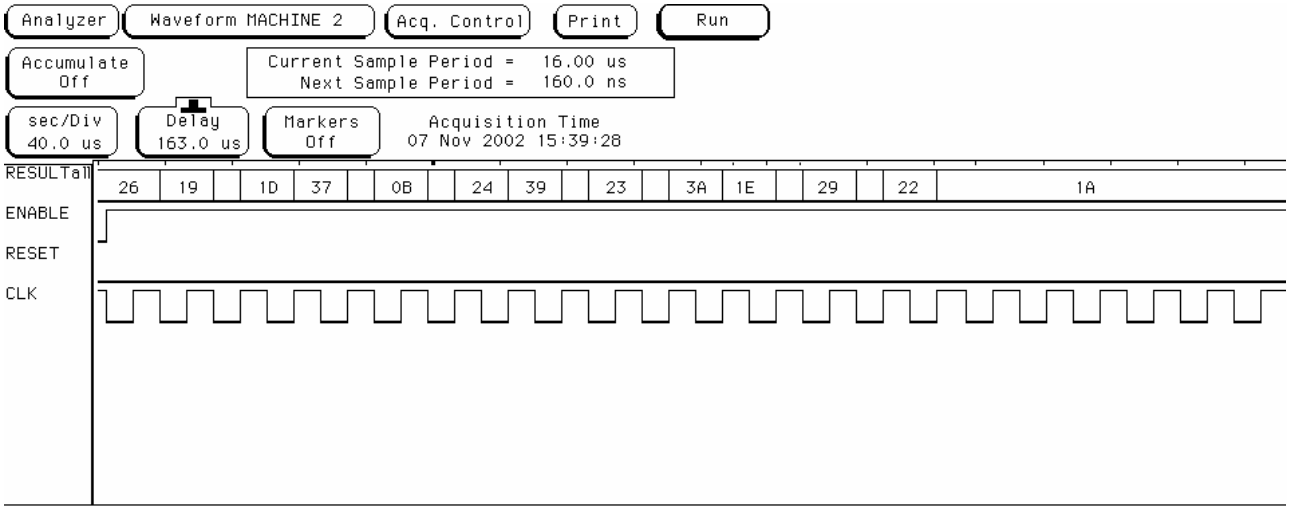


Figura. 4 Formas de onda de las señales implementadas. Se encuentra la respuesta correcta en 1.568 mseg con un clock de 500 hertz.

5. CONCLUSIONES

El problema de satisfacción Booleana (SAT), es un problema combinacional que consiste en encontrar el valor de las entradas de una función booleana de tal forma que la salida sea igual a uno.

Este artículo muestra las ventajas que conlleva la resolución de Problemas SAT por medio de dispositivos lógicos programables como son los FPGA (Field Programmable Gate Array), debido a sus características paralelas y su enorme versatilidad, que los hacen más eficientes, logrando la solución en unos pocos ciclos de reloj. En contraste se describe el proceso, lento y complejo, llevado a cabo para la solución del mismo problema por medio de la computación molecular por ADN.

En el algoritmo presentado, se realizó un procedimiento similar al seguido por la computación molecular por ADN. Se mostró que existen dispositivos, como los FPGAs, que también trabajan en forma paralela, optimizando el tiempo empleado para manipular datos.

Entre las ventajas de resolver por medio de los FPGAs el problema de búsqueda intensiva se encuentran:

1) La forma para crear los diferentes literales en los FPGAs, se basa en una programación con un lenguaje de alto nivel como el VHDL, haciendo este proceso más sencillo que la formación de literales por ADN, ya que este proceso requiere de mayor tiempo (más de 1000 minutos) y de controles químicos y de temperatura.

2) El proceso de diseño y de modificación del problema es muy complejo en la computación por ADN, ya que se debe tener en cuenta que el número de uniones indeseadas entre cadenas sencillas durante el proceso de hibridación debe ser mínima. Por otra parte, en los FPGAs estos cambios se realizan en el programa por medio de un lenguaje de alto nivel.

3) La forma de visualizar el resultado en los FPGAs es muy sencilla, pues existen diversas formas de detectar un nivel lógico alto o bajo. Para la computación por ADN, es necesario un proceso de rayos X para visualizar la solución del problema.

Aún cuando se presenten todos estos inconvenientes con la computación molecular, se debe tener en cuenta que es una ciencia que lleva muy pocos años de investigación y así como se nombran algunas debilidades que presenta frente a la solución obtenida por medio de los FPGAs, es bueno anotar que una gran ventaja de la computación molecular por ADN es su gran capacidad, pues en una gota de agua, se podrían contener miles de literales. Por lo tanto la computación molecular por ADN se debe estudiar y desarrollar hacia campos donde sus propiedades sean mejor aprovechadas.

6. REFERENCIAS

[1] M. Davis, H. Putnam "*A Computing Procedure for Quantification Theory*," *Journal of the ACM*, pp. 201-215, Septiembre 1959.

[2] R.S Braich, N. Chelyapov, C. Johnson, P.W.K Rothemund, L. Adleman: "*Solution of a 20-variable 3-SAT problem on a DNA computer*". Science,Vol. 296, pp. 499 – 502, 2002.

[3] K.Sakamoto,H. Gouzu,K. Komiya,D. Kiga,S.Yokoyama, T. Yokomori, M. Hagiya "*Molecular Computation by ADN Hairpin Formation*", Science Vol 288, 1223- 1226, Mayo 2000

[4] M. Adleman, "*Molecular Computation of Solutions to Combinational Problems"* Science Vol 266, pp. 1021 - 1024 , 1994.

[5] L.Rojas, R. García: "*Simulación de ADN de circuitos booleanos combinatorios*". Ingeniería Eléctrica. Universidad Nacional., pp 32 – 54, 2002

[6] R. Lipton, "*ADN solution of Hard Computational Problems*", Science, pp. 268-542 , Abril 1995.

[7] B,Hutchings, B, nelson "*Uniying simulation and execution in a design environment for FPGA systems*" IEEE transactions on very large scale integration VLSI systems,Vol 9, pp 201-202 Febrero 2001.

[8] P.Zhong, M. Martonosi, P. Ashar,S.Malik. "*Accelerating Boolean Satisfiability with Configurable Hardware*", in IEEE,pp. 186-195.1998.

[9] P. Chu, R. Jones, "Design techniques of FPGA based random number generator". Online : *academic.csuohio.edu:8080/chup/chu_web_stuff/research_stuff/random*