Diseño y desarrollo de una arquitectura de procesamiento para el censado y tratamiento de variables meteorológicas en sistemas multiprocesamiento en bajo nivel con FPGAs en tiempo real.

Angel González Rodríguez

Universidad Autónoma del Estado de México (UAEMex)

Benemérita Universidad Autónoma de Puebla (BUAP)

México

angelgoro981118@gmail.com

**Resumen.** En este trabajo se presenta el diseño de una arquitectura de procesamiento para FPGAs en el lenguaje VHDL, que permite procesar paquetes de información meteorológica específicos en un determinado tiempo de operación. El diseño permite aproximar a una línea recta con base a un conjunto de puntos dispersos por medio del método analítico de mínimos cuadrados. Además de definir una solución para la combinación de parámetros meteorológicos, se desarrolló un procesamiento en paralelo, que permite almacenar y tratar los paquetes de información al mismo tiempo.

Abstract. In this work, the design of a processing architecture for FPGAs in the VHDL language is presented, which allows to process specific meteorological information packages in a certain operating time. The design allows to approximate a straight line based on a set of scattered points by means of the analytical method of least squares. In addition to defining a solution for the combination of meteorological parameters, parallel processing was developed, which allows the information packets to be stored and processed at the same time.

1 Introducción

Estudios recientes llevados a cabo por la NASA han revelado que la temperatura media global ya ha aumentado casi un grado desde el inicio de la revolución industrial en 1750. El planeta se encuentra dentro del rango permisible para evitar un cambio climático peligroso, es por lo que es urgente tomar acción efecto que permitan reducir emisiones y preparen, de una mejor manera a las comunidades, así como a la sociedad en general para un mundo más caluroso. A causa de los problemas meteorológicos actuales y aquellos que se pronostican futuros, es necesario tener un acceso inmediato, eficaz y viable para conocer las características, y, sobre todo, los principales cambios en una zona geográfica determinada.

* 1. **Servicio Meteorológico Nacional**

En México, el organismo público encargado de proporcionar información sobre el estado del tiempo, así como las variaciones y pronósticos en las variables meteorológicas es el Servicio Meteorológico Nacional (SMN). La sociedad, en general, sin exclusión tiene acceso a su sitio web.

El diseño e interacción con el usuario que ofrece el sitio web del Servicio Meteorológico Nacional es vasto de información, es confiable y seguro, pero, es poco interactivo, no es amigable y predictivo. Los usuarios toman más tiempo observando y deduciendo donde dar un “clic” para desplegar la información realmente útil para cada uno. Aunado a esta problemática, la información que brinda el sistema esta retrasada, no se presentan variaciones de parámetros en tiempo real, a causa de hacer uso de información tanto nacional como local. El retraso de tiempo en la información, en su mayoría se debe al desvió y el procesamiento que se realiza con ella.

El sistema del SMN se implementa de otros sistemas de sondeo y muestro basados en estaciones meteorológicas automáticas (EMA´s), ubicando puntualmente 189 distribuidas en todo México, encargadas de medir, principalmente velocidad del viento, dirección del viento, presión atmosférica, temperatura y humedad relativa y precipitación. La información de cada variable meteorológica tiene una actualización constante cada 15 minutos, dentro de cada estación meteorológica, sin embargo, él envió de información ocurre casi 2 horas después de la notificación, este lapso, si bien, no es mucho, dentro de la tolerancia para meteorología es considerablemente extenso, no es posible recibir alertas o información de una variación considerable de cualquier índole en este periodo de tiempo. El tiempo es crucial dentro de las decisiones y pronósticos meteorológicos, un segundo hace la diferencia, por esta razón, lo que se busca dentro de este proyecto es la comunicación y visualización en tiempo real, de cualquier variación sobre las variables meteorológicas antes mencionadas.

* 1. **Sistema de adquisición de datos**

La tarea fundamental de un sistema de adquisición de datos consiste en la medición o generación de señales físicas del mundo real. Antes de que una señal física pueda ser medida por un sistema de cómputo, se requiere un sensor o transductor que convierta la señal física en una señal eléctrica, tal como un voltaje o una corriente. Comúnmente, la tarjeta de adquisición de datos se considera como todo el sistema de adquisición, sin embargo, es únicamente uno de los componentes de este. A diferencia de los instrumentos autónomos, no pueden conectarse directamente a la tarjeta de adquisición de datos, algunas veces estas necesitan ser condicionadas para el proceso de adquisición y tratamiento de datos, obteniendo los datos en bruto, analizando la información, realizando operaciones lógicas o aritméticas y finalmente, presentando los resultados.

* 1. **Arquitectura de procesamiento**

Muchos problemas de Ingeniería requieren de la medición y análisis de datos generados por uno o varios sensores específicos. Dos de los parámetros importantes en un sistema de adquisición de datos son el número de muestras requerido y el tiempo en el que se registra cada medición (tiempo de muestreo). La utilización del lenguaje VHDL [1] es aceptada desde hace mucho tiempo para la utilización en el diseño, simulación e implementación de hardware [2]. Sin embargo, sigue siendo un problema por resolver la optimización de distintos aspectos de los circuitos diseñados a la hora de su implementación real. En este contexto las herramientas de CAD (Computer Aided Design) como Alliance [3] [4] y Cadence [5] son de gran utilidad no solamente para el ciclo completo de desarrollo y verificación de hardware sino también para el análisis de alternativas de optimización.

Se presenta en el documento la posible implementación del sistema en una tarjeta de desarrollo DE0 NANO SOC (utilizada en algunas de las asignaturas del área de electrónica digital en la BUAP, siendo considerada como de las tarjetas de desarrollo más complejas dentro de la institución) como sistema de adquisición de datos.

La tarjeta cuenta con un FPGA (Arreglo de Compuertas Programable en Campo) en el cual se implementará una serie de circuitos digitales expresados por medio de un lenguaje descriptivo de Hardware (VHDL) a fin de implementar el sistema de adquisición de datos donde se leerán y tratarán paquetes de datos temporales provenientes de un sistema CANSAT, para posteriormente ser exportada la ecuación de la recta que mejor se forme con base a un conjunto de datos dispersos, a una hoja de cálculo para su análisis posterior.

2. DESCRIPCION DEL SISTEMA

Se requiere implementar una arquitectura de procesamiento especifica que permita almacenar y tratar paquetes de información meteorológica tales como temperatura, tiempo y posición. Se considera que los paquetes de información meteorológica vienen desde Mini-Laboratorios móviles aéreos con conectividad de RF por medio de drones, globos aerostáticos, por medio de un CANSAT se establece la transmisión de datos posterior a la FPGA donde se implementara la arquitectura.

Para la implementación de la arquitectura se requieren bajos recursos de la tarjeta, debido a la sencillez de estructura en la arquitectura, como es de esperarse, se requiere un sistema sencillo en su interior, pero se requiere complejo y lo suficientemente resistente para tratar con cientos o miles de paquetes de información en tiempo real. Para llevar a cabo el desarrollo de la arquitectura se opto por utilizar el software de Quartus II que brinda Altera, debido a su sencillez, eficacia y compatibilidad con las tarjetas con las que se cuenta, del mismo fabricante.

Al comenzar a describir el código en VHDL se consideraron las 3 posibles entradas de los paquetes de información, como lo son la posición, la temperatura y el tiempo, con la idea principal de obtener la ecuación de la recta que se ajuste a un conjunto de datos dispersos, considerando 3 posibles resultados, temperatura con base en la posición, temperatura con base en el tiempo, y la posición con base en el tiempo.

Se presentan 3 relaciones para obtener con base en la ecuación de la recta, descrita por la siguiente formula:

|  |  |
| --- | --- |
|  | (**1**) |

Como se observa, es necesario obtener los parámetros “m” y “b”, siendo los dos valores imprescindibles y únicos para generar la ecuación de la recta. Se opto por utilizar la técnica de mínimos cuadrados, dentro de la optimización matemática, en la que, dados un conjunto de pares ordenados, como lo es la variable independiente (x), y la variable dependiente (y), además de una familia de funciones, se intenta encontrar la función continua, dentro de dicha familia, que mejor se aproxime a los datos, con un mejor ajuste. Para llevar a cabo la técnica se modificó la ecuación de la recta con base en los parámetros y especificaciones que rige la técnica de mínimos cuadrados, considerando las variables “a” y “b” como el par ordenado de los datos, y “n” como el número de muestras requerido, se describen las siguientes fórmulas para la pendiente y el desplazamiento:

|  |  |
| --- | --- |
|  | (**2**) |
|  | (**3**) |

Generando así, la ecuación de la recta con las aproximaciones que se basan en la técnica de mínimos cuadrados en la siguiente ecuación:

|  |  |
| --- | --- |
|  | (**4**) |

A referencia de la ecuación (4) se realizo la mayor parte del código VHDL, realizando las operaciones básicas de suma, resta, multiplicación y división sobre los vectores definidos por “std\_logic\_vector (x downto 0)” en la arquitectura del lenguaje VHDL. En la descripción de la arquitectura, se presentan 3 procesos principales, cada uno de estos presenta la aproximación de pares ordenados para cada una de las 3 relaciones o líneas rectas necesarias. En cada proceso se utiliza la forma general para obtener la aproximación, se presentan algunas limitantes que son consideradas y resueltas en cada proceso. Como se observa en la ecuación (4) se requiere de sumas, restas, multiplicaciones y divisiones, así como de variables acumulativas que aumentan su valor en cada ciclo de reloj, hasta llegar al numero máximo de muestras requerido e inicializar su valor en cero nuevamente, para llevar a cabo esta tarea se realizó el siguiente proceso:

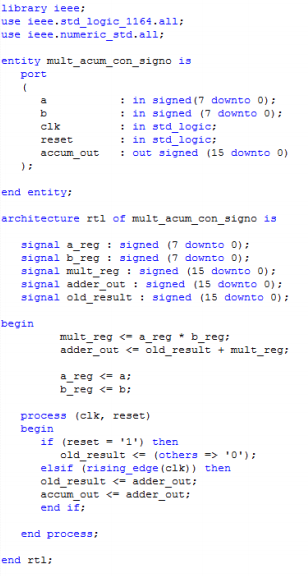


Figura 1: Código en VHDL de una multiplicacion acumulativa.

El código consiste en considerar 2 señales de entrada, de 8 bits cada una, realizar la multiplicacion de las señales considerando como respuesta un vector con el doble de bits, en este caso 16 bits, este resultado entra a un proceso de incremento acumulativo, es decir a este valor se suma el valor anterior (0) y se toma dicho resultado, para la siguiente interacción, se vuelve a realizar la multiplicacion y en este caso ya se presenta un resultado diferente de cero, el cual es sumado a la variable acumulativa, el proceso se realiza hasta que el número de muestras totales se alcanza, en ese momento se reinicializan los valores en 0, y comienza la cuenta.

Dentro del registro acumulativo se considera una entrada de reloj y un clear, el cual permite sincronizar los procesos, cuando se cambia de bajo a alto en el reloj se realiza el proceso para cada uno de los valores de entrada, pero mientras el clear se mantenga en alto, todos los valores se vuelven cero y no se registra la cuenta. Como se presenta en el código se debe de cumplir un cambio a alto en el reloj y un cambio a bajo del clear para procesar la información. La señal clear permite organizar los paquetes de datos, otorgando así el número de muestras y el tiempo de muestreo.

RESULTADOS

El método propuesto se implementó en la arquitectura, se consideraron los casos especiales en cada uno de los procesos, obteniendo así una respuesta y comportamiento apto para la adquisición y procesamiento de los datos meteorológicos.

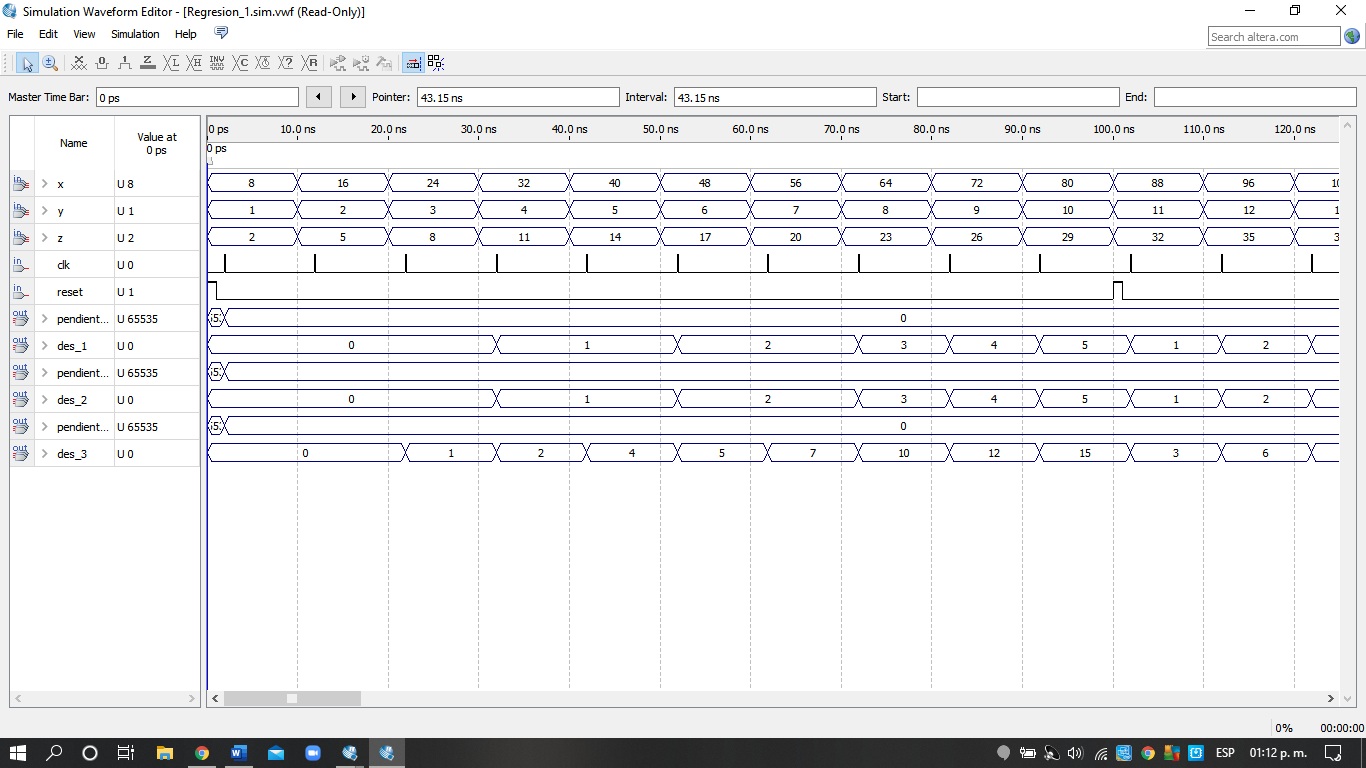


Figura 2: Univeristy Program VWF de la arquitectura.

Se presentan en la Figura 2, las entradas y salidas de la arquitectura en general, considerando las entradas de datos, “x” como temperatura, “y” como tiempo, “z” como posición, las entradas que permiten el control del sistema, un reloj que permite obtener datos cada cambio a alto, es decir, cada 10ns, mientras que la señal de borrado permite y limita la toma de muestras necesarias, en este caso 10 para cada proceso. Considerando que las salidas que se muestran son representadas por los dos componentes necesarios para definir la línea recta, la pendiente y el desplazamiento, para cada uno de los 3 casos de procesos. Se opto por mostrar los valores y resultados como decimal sin signo, esto para facilitar la visualización de los resultados.

CONCLUSIONES

Este trabajo fue diseñado y desarrollado con la finalidad de aprovechar algunas de las características más importantes e influyentes del lenguaje VHDL. Gracias a la potencia y reconocimiento del lenguaje fue posible detallar aspectos importantes para mejorar el funcionamiento del sistema.

Es posible el uso e implementación de la arquitectura a una tarjeta de desarrollo física, operando como un sistema de adquisición de datos a partir de describir bloques digitales simples en lenguaje descriptivo VHDL. Se representa una alternativa con una arquitectura rígida, al poderse modificar conforme a las necesidades del usuario en el caso múltiples sensores, número de muestras y tiempos de muestreo, considerando siempre las limitaciones del sistema y de la tarjeta de desarrollo físico.

Durante el diseño y desarrollo de la arquitectura se presentaron bastantes problemas relacionadas al tratamiento y manejo del lenguaje VHDL, con ayuda de la bibliografía utilizada y mencionada, se dio solución a cada problema.

Uno de los problemas que mayor auge tuvieron a lo largo del desarrollo de la arquitectura fue la carencia de los recursos físicos necesarios, así como la ausencia presencial durante el desarrollo de cada etapa del proyecto debido a las precauciones sanitarias que actualmente se llevan a cabo. Sin embargo, sin considerar esta limitante fuera de nuestras manos, la arquitectura se trató de diseñar lo mejor posible, con las limitantes que se presentaban día a día, y considerando el diseño hasta nivel simulación con ayuda de la herramienta “University Program VWF” que el software Quartus II de Altera, presenta como documento. La visualización del diagrama de tiempo generado por la herramienta fue de vital importancia para observar el comportamiento físico que presentaría el sistema y la arquitectura en un ambiente controlado.

Durante el desarrollo del estado del arte para llevar a cabo el proyecto se presentaron interrogantes y limitaciones en el conjunto de adquisición y procesamiento de los datos, derivado de esto, para la arquitectura en general se consideraron paquetes de información de tipo entero, es decir omitiendo la parte fraccionaria de la información, así como el signo de cada uno de los datos, esto debido a la complejidad de las operaciones realizadas en cada uno de los procesos, así como errores frecuentes ocasionados por los signos de los datos y su compatibilidad con el resto de librerías para la programación de VHDL.

REFERENCIAS

1. IEEE Standard VHDL Language Reference Manual, Std. 1076-1993, IEEE, NY, 1993.
2. J. Bhasker, (1995). A VHDL Primer, Prentice Hall PTR, Upper Saddle River NJ 07458.
3. Kai-shing Lam, Frederic Ak, (2004). “Alliance Tutorial, Part 1 VHDL Modeling and Simulation”, Pierre & Marie Curie University, France, pp. 2-16. Alliance distribution: <http://www-asim.lip6.fr/pub/alliance/distribution/latest>.
4. Greiner A., F. Pêcheux. (1992). “ALLIANCE . A Complete Set of CAD Tools for Teaching VLSI Design”, Proceedings of the Third Eurochip Workshop on VLSI Design Training, pp. 230-37, Grenoble, France, Sept.
5. Virginia Polytechnic Institute and State University. (2003) “Cadence tutorial for VLSI Design”. http://www.ee.vt.edu/~ha/cadtools/cadence/cadence.html
6. Machado, F., Borromeo, S. (2010). Diseño de circuitos digitales con VHDL. Departamento de Tecnología Electrónica. Universidad Rey Juan Carlos, 1.
7. Haskel, R. E. & Hanna D. M. (2009) Learning by Example Using VHDL. Advanced Digital Design. Rochester, MI LBE Books.
8. T. L. Floyd. (2000). Fundamentos de Sistemas Digitales, Prentice-Hall, Madrid.
9. G. Maxinez David, Alcala Jessica. (2002). VHDL: El Arte de Programar Sistemas Digitales, CECSA (Compañía Editorial CONTINEN), 352.
10. Torre Albarsanz, M. (2015). Análisis de temperatura en FPGAs (Bachelor's thesis).
11. Reyes, O., Cadena, O., & Correa, R. (2007). Design of a prototype for measuring heat flux using direct calorimetry by temperature variation measurements. Ingeniería y Desarrollo, (22), 85-106.
12. López Velasco, W. E., & Moyón Amaguaya, N. F. (2011). Desarrollo de un Sistema Scada para la Medición de Temperatura con Sistemas Embebidos para el Laboratorio de Mecatrónica de la Facultad de Mecánica (Bachelor's thesis).
13. Argüello, D., Pérez, S., Molina, S., & Morales, E. Diseño, desarrollo y análisis de prestaciones de un procesador académico modular avanzado usando lenguajes de descripción de hardware HDL y FPGA.
14. Cirstea, M., Aounis, A., McCormick, M. y Urwin, P. (2001). Diseño y análisis de sistemas de control vectorial mediante VHDL [para motores de inducción]. IEEE 32ª Conferencia Anual de Especialistas en Electrónica de Potencia (IEEE Cat. No. 01CH37230) (1), 81-84.
15. Brown, Stephen D. (1992). Field-Programmable Gate Arrays. Boston: The Kluwer International series in engineering and computer science, SECS 180.
16. Chang, Kou-Chuan. (1999). Digital Systems Design with VHDL and Synthesis: An Integrated Approach. Los Alamitos: IEEE Computer Society.
17. Harris, David Money, y Sarah L. Harris. (2007). Digital design and computer architecture. Amsterdam: Morgan Kaufmann Publishers.
18. J. I. Artigas, L. Á. Barragán, C. Orrite e I. Urriza. (2002). Electrónica digital. Aplicaciones y problemas con VHDL, Prentice-Hall, Madrid.
19. K. Skahill. (1996). VHDL for Programmable Logic, Addison-Wesley.

APRENDICE 1. CÓDIGO EN VHDL GENERAL DEL DISEÑO DE LA ARQUITECTURA

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

entity Regresion\_1 is

generic ( m : natural := 8;

l : natural := 16;

o : natural := 32;

n : natural := 10

);

port ( clk : in std\_logic;

reset : in std\_logic;

x : in std\_logic\_vector (m-1 downto 0); --Temperatura

y : in std\_logic\_vector (m-1 downto 0); --Tiempo

z : in std\_logic\_vector (m-1 downto 0); --Posicion

pendiente\_1 : buffer std\_logic\_vector (l-1 downto 0);

pendiente\_2 : buffer std\_logic\_vector (l-1 downto 0);

pendiente\_3 : buffer std\_logic\_vector (l-1 downto 0);

des\_1 : buffer std\_logic\_vector (o-1 downto 0);

des\_2 : buffer std\_logic\_vector (o-1 downto 0);

des\_3 : buffer std\_logic\_vector (o-1 downto 0)

);

end entity;

architecture arch\_alu of Regresion\_1 is

begin

--Temperatura vs Tiempo

process (clk,reset,x,y)

variable result1: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable result2: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable result3: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable result4: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable x\_y: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable x\_2: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable dos: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable cuatro: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable num: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable den: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable uno: std\_logic\_vector(o-1 DOWNTO 0) := (others => '0');

variable dos\_1: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable dos\_2: std\_logic\_vector(o-1 DOWNTO 0) := (others => '0');

variable mult\_reg1: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable adder\_out1: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable old\_result1: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable mult\_reg2: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable adder\_out2: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable old\_result2: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable mult\_reg3: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable adder\_out3: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable old\_result3: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable mult\_reg4: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable adder\_out4: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable old\_result4: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

begin

--1) Suma total de (x\*y)

mult\_reg1:=std\_logic\_vector(unsigned(x)\*unsigned(y)); adder\_out1:=std\_logic\_vector(unsigned(old\_result1)+unsigned(mult\_reg1));

--2) Suma total de x

mult\_reg2:=x; adder\_out2:=std\_logic\_vector(unsigned(old\_result2)+unsigned(mult\_reg2));

--3) Suma total de y

mult\_reg3:=y; adder\_out3:=std\_logic\_vector(unsigned(old\_result3)+unsigned(mult\_reg3));

--4) Suma total de (x^2)

mult\_reg4:=std\_logic\_vector(unsigned(x) \* unsigned(x));

adder\_out4:=std\_logic\_vector(unsigned(old\_result4)+unsigned(mult\_reg4));

IF (reset = '1') THEN

old\_result1 := (others => '0');

old\_result2 := (others => '0');

old\_result3 := (others => '0');

old\_result4 := (others => '0');

ELSIF (rising\_edge(clk)) THEN

old\_result1 := adder\_out1;

result1 := adder\_out1;

old\_result2 := adder\_out2;

result2 := adder\_out2;

old\_result3 := adder\_out3;

result3 := adder\_out3;

old\_result4 := adder\_out4;

result4 := adder\_out4;

END IF;

--PENDIENTE

x\_y := std\_logic\_vector(unsigned(result2) \* unsigned(result3));

dos := std\_logic\_vector(unsigned(x\_y) / n);

num := std\_logic\_vector(unsigned(result1) - unsigned(dos));

x\_2 := std\_logic\_vector(unsigned(result2) \* unsigned(result2));

cuatro := std\_logic\_vector(unsigned(x\_2) / n);

den := std\_logic\_vector(unsigned(result4) - unsigned(cuatro));

pendiente\_1 <= std\_logic\_vector(unsigned(num) / unsigned(den));

--DESPLAZAMIENTO

uno := std\_logic\_vector('0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & unsigned(result3) / n);

dos\_1 := std\_logic\_vector('0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & unsigned(result2) / n);

dos\_2 := std\_logic\_vector(unsigned(pendiente\_1) \* unsigned(dos\_1));

des\_1 <= std\_logic\_vector(unsigned(uno) - unsigned(dos\_2));

END PROCESS;

--Posicion vs Tiempo

process (clk,reset,z,y)

variable result1\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable result2\_10: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable result3\_10: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable result4\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable z\_y: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable z\_2: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable dos\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable cuatro\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable num\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable den\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable uno\_10: std\_logic\_vector(o-1 DOWNTO 0) := (others => '0');

variable dos\_100: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable dos\_200: std\_logic\_vector(o-1 DOWNTO 0) := (others => '0');

variable mult\_reg1\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable adder\_out1\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable old\_result1\_10:std\_logic\_vector(l-1 DOWNTO 0):= (others => '0');

variable mult\_reg2\_10: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable adder\_out2\_10: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable old\_result2\_10:std\_logic\_vector(m-1 DOWNTO 0):= (others => '0');

variable mult\_reg3\_10: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable adder\_out3\_10: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable old\_result3\_10:std\_logic\_vector(m-1 DOWNTO 0):= (others => '0');

variable mult\_reg4\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable adder\_out4\_10: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable old\_result4\_10:std\_logic\_vector(l-1 DOWNTO 0):= (others => '0');

begin

--1) Suma total de (z\*y)

mult\_reg1\_10:=std\_logic\_vector(unsigned(z)\*unsigned(y)); adder\_out1\_10:=std\_logic\_vector(unsigned(old\_result1\_10)+unsigned(mult\_reg1\_10));

--2) Suma total de z

mult\_reg2\_10 := z;

adder\_out2\_10:=std\_logic\_vector(unsigned(old\_result2\_10)+unsigned(mult\_reg2\_10));

--3) Suma total de y

mult\_reg3\_10 := y;

adder\_out3\_10:=std\_logic\_vector(unsigned(old\_result3\_10)+unsigned(mult\_reg3\_10));

--4) Suma total de (z^2)

mult\_reg4\_10 := std\_logic\_vector(unsigned(z) \* unsigned(z));

adder\_out4\_10:=std\_logic\_vector(unsigned(old\_result4\_10)+unsigned(mult\_reg4\_10));

IF (reset = '1') THEN

old\_result1\_10 := (others => '0');

old\_result2\_10 := (others => '0');

old\_result3\_10 := (others => '0');

old\_result4\_10 := (others => '0');

ELSIF (rising\_edge(clk)) THEN

old\_result1\_10 := adder\_out1\_10;

result1\_10 := adder\_out1\_10;

old\_result2\_10 := adder\_out2\_10;

result2\_10 := adder\_out2\_10;

old\_result3\_10 := adder\_out3\_10;

result3\_10 := adder\_out3\_10;

old\_result4\_10 := adder\_out4\_10;

result4\_10 := adder\_out4\_10;

END IF;

--PENDIENTE

z\_y := std\_logic\_vector(unsigned(result2\_10) \* unsigned(result3\_10));

dos\_10 := std\_logic\_vector(unsigned(z\_y) / n);

num\_10 := std\_logic\_vector(unsigned(result1\_10) - unsigned(dos\_10));

z\_2 := std\_logic\_vector(unsigned(result2\_10) \* unsigned(result2\_10));

cuatro\_10 := std\_logic\_vector(unsigned(z\_2) / n);

den\_10 := std\_logic\_vector(unsigned(result4\_10) - unsigned(cuatro\_10));

pendiente\_2 <= std\_logic\_vector(unsigned(num\_10) / unsigned(den\_10));

-- DESPLAZAMIENTO

uno\_10 := std\_logic\_vector('0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & unsigned(result3\_10) / n);

dos\_100 := std\_logic\_vector('0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & unsigned(result2\_10) / n);

dos\_200 := std\_logic\_vector(unsigned(pendiente\_2) \* unsigned(dos\_100));

des\_2 <= std\_logic\_vector(unsigned(uno\_10) - unsigned(dos\_200));

END PROCESS;

--Temperatura vs Posicion

process (clk,reset,x,z)

variable result1\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable result2\_20: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable result3\_20: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable result4\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable x\_z: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable x\_2: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable dos\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable cuatro\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable num\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable den\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable uno\_20: std\_logic\_vector(o-1 DOWNTO 0) := (others => '0');

variable dos\_1000: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable dos\_2000: std\_logic\_vector(o-1 DOWNTO 0) := (others => '0');

variable mult\_reg1\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable adder\_out1\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable old\_result1\_20:std\_logic\_vector(l-1 DOWNTO 0):= (others => '0');

variable mult\_reg2\_20: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable adder\_out2\_20: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable old\_result2\_20:std\_logic\_vector(m-1 DOWNTO 0):= (others => '0');

variable mult\_reg3\_20: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable adder\_out3\_20: std\_logic\_vector(m-1 DOWNTO 0) := (others => '0');

variable old\_result3\_20:std\_logic\_vector(m-1 DOWNTO 0):= (others => '0');

variable mult\_reg4\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable adder\_out4\_20: std\_logic\_vector(l-1 DOWNTO 0) := (others => '0');

variable old\_result4\_20:std\_logic\_vector(l-1 DOWNTO 0):= (others => '0');

begin

--1) Suma total de (x\*z)

mult\_reg1\_20 := std\_logic\_vector(unsigned(x) \* unsigned(z));

adder\_out1\_20:=std\_logic\_vector(unsigned(old\_result1\_20)+unsigned(mult\_reg1\_20));

--2) Suma total de x

mult\_reg2\_20 := x;

adder\_out2\_20:=std\_logic\_vector(unsigned(old\_result2\_20)+unsigned(mult\_reg2\_20));

--3) Suma total de z

mult\_reg3\_20 := z;

adder\_out3\_20:=std\_logic\_vector(unsigned(old\_result3\_20)+unsigned(mult\_reg3\_20));

--4) Suma total de (x^2)

mult\_reg4\_20 := std\_logic\_vector(unsigned(x) \* unsigned(x));

adder\_out4\_20:=std\_logic\_vector(unsigned(old\_result4\_20)+unsigned(mult\_reg4\_20));

IF (reset = '1') THEN

old\_result1\_20 := (others => '0');

old\_result2\_20 := (others => '0');

old\_result3\_20 := (others => '0');

old\_result4\_20 := (others => '0');

ELSIF (rising\_edge(clk)) THEN

old\_result1\_20 := adder\_out1\_20;

result1\_20 := adder\_out1\_20;

old\_result2\_20 := adder\_out2\_20;

result2\_20 := adder\_out2\_20;

old\_result3\_20 := adder\_out3\_20;

result3\_20 := adder\_out3\_20;

old\_result4\_20 := adder\_out4\_20;

result4\_20 := adder\_out4\_20;

END IF;

--PENDIENTE

x\_z := std\_logic\_vector(unsigned(result2\_20) \* unsigned(result3\_20));

dos\_20 := std\_logic\_vector(unsigned(x\_z) / n);

num\_20 := std\_logic\_vector(unsigned(result1\_20) - unsigned(dos\_20));

x\_2 := std\_logic\_vector(unsigned(result2\_20) \* unsigned(result2\_20));

cuatro\_20 := std\_logic\_vector(unsigned(x\_2) / n);

den\_20 := std\_logic\_vector(unsigned(result4\_20) - unsigned(cuatro\_20));

pendiente\_3 <= std\_logic\_vector(unsigned(num\_20) / unsigned(den\_20));

-- DESPLAZAMIENTO

uno\_20 := std\_logic\_vector('0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & unsigned(result3\_20) / n);

dos\_1000 := std\_logic\_vector('0' & '0' & '0' & '0' & '0' & '0' & '0' & '0' & unsigned(result2\_20) / n);

dos\_2000 := std\_logic\_vector(unsigned(pendiente\_3) \* unsigned(dos\_1000));

des\_3 <= std\_logic\_vector(unsigned(uno\_20) - unsigned(dos\_2000));

END PROCESS;

end architecture;