

Simulación de sistemas de modulación digital en banda base para ambientes de cómputo paralelo basado en lenguaje ZPL

Carlos Peña, Demetrio Rey Lago

Instituto de Matemática y Cálculo Aplicado (IMYCA), Facultad de Ingeniería

Universidad de Carabobo, Valencia, Venezuela

Email: penacj@uc.edu.ve, drey@uc.edu.ve

Resumen

En este trabajo se describe una forma de utilizar el lenguaje ZPL para simular sistemas de modulación digital en banda base en ambientes de arquitectura paralela, basados en cluster de computadoras. Para ello se presenta un análisis del potencial de paralelización de un sistema de modulación digital, lo que permitiría aprovechar la capacidad de cómputo matricial de ZPL. Simulaciones para el cómputo de tasa de error de bits en sistemas M-PSK y M-QAM son implementadas en los lenguajes ZPL, C y C++, los dos últimos como programas de comparación referencial. Los resultados muestran una reducción importante en el número de líneas de código así como en los tiempos de procesamiento al incorporar múltiples procesadores de la arquitectura paralela.

Palabras clave: Modulación digital, ZPL, cluster, paralelismo, simulación.

Simulation of baseband digital modulation systems for parallel computing environment based on ZPL language

Abstract

This work describes a way of using ZPL language to simulate digital modulation systems in parallel architectures, based on cluster of workstations. An analysis is presented to show the parallelism potential in digital modulation systems, taking advantage of the capacity for data-parallel array programming of ZPL. Simulations for bit error rate computation were implemented using ZPL, in M-PSK and M-QAM systems. In addition, software in C and C++ was developed as a reference. The results show an important reduction in the number of code lines as well as a reduction in processing time when multiple processors from the cluster are added to the computational process.

Keywords: Digital modulation, ZPL, cluster, parallelism, simulation.

1. INTRODUCCIÓN

La simulación por medio de herramientas computacionales es uno de los esquemas más utilizados en el campo de la investigación de los sistemas de comunicaciones digitales; esto constituye una forma de investigación de bajo costo y altamente flexible, que permite la predicción de los comportamientos de los elementos de objeto de estudio. Recientemente, ha habido un marcado incremento en el uso de recursos computacionales, tales como: memoria, capacidad de procesamiento, número de procesadores; debido a la alta complejidad en los sistemas de comunicaciones modernos. Una solución empleada para la simulación

de sistemas que demanden gran cantidad de recursos de cómputo, lo constituyen los computadores en arquitecturas paralelas [1].

Las arquitecturas de cómputo paralelo interpretan códigos escritos para tal fin. Estos programas ejecutan y distribuyen las cargas entre todos los procesadores de manera equitativa, con el propósito de utilizar de manera eficiente sus recursos y reducir los tiempos de cálculo y simulación. Una de las principales desventajas de estas arquitecturas es el hecho de requerir código específico para la realización de actividades de intercambio de información y asignación de recursos, no requerido en la arquitectura secuencial tradicional;

lo que aumenta la complejidad del programa, haciendo que en la práctica el cómputo paralelo se convierta en una herramienta poco atractiva para el investigador.

2. EL LENGUAJE ZPL

El lenguaje ZPL está concebido como cualquier otro lenguaje de arreglos, es decir este posee operadores elemento a elemento, grupos de datos de un arreglo o al arreglo completo, sin la necesidad de definir en forma explícita los índices correspondientes; por ejemplo $X := Y + W$. ZPL cuenta además con un conjunto de operaciones matemáticas tradicionales, así como operaciones de manipulación matricial [2][3]. Algunos de estos operadores son:

Tabla 1. Operadores del lenguaje ZPL.

Nombre del Comando	Símbolo	Propósito
Elemento a elemento	+ * - /	Operaciones aritméticas
Flood	>>	Replicación de data
Reduce	<<	Reducción de data
Elemento a elemento	sin, cos, ln	Funciones matemáticas

Una ventaja apreciable en ZPL, lo constituye el hecho que ZPL está definido en regiones [2]. Una región está definida por un conjunto de índices del tipo rectangular $[1...L, 1...M]$, sobre el cual se ejecutarán las operaciones de cómputo y se declararán las variables respectivas, es importante hacer notar que debe existir coincidencia de dimensionalidad cuando se realicen las diferentes operaciones de cómputo entre las variables. Los arreglos o variables rectangulares definidas en una región son la fuente de concurrencia del lenguaje, los datos asignados en un arreglo son almacenados en los procesadores del cluster y cada procesador realiza operaciones matriciales sobre éstos.

Dependiendo del tipo de operación invocada, los procesadores pueden requerir o no comunicación. Esta comunicación no requiere de definición explícita ya que es manejada directamente por el compilador, que está optimizado para esta función. Otorgándole una gran portabilidad al programa, ya que un código de cómputo puede ser escrito sin distinguir en el número de procesadores que conforme la red del cluster.

3. EJEMPLO DE UN SISTEMA DE MODULACIÓN DIGITAL EN ZPL

La modulación digital es el proceso mediante el cual se transforman los símbolos digitales en formas de onda adecuadas para la transmisión a través de un canal de comunicación [4]. Estos símbolos son mapeados a una señal con forma de onda típicamente sinusoidal (1), donde su amplitud y fase varían con respecto al tiempo.

$$s(t) = A(t) \cos [\omega_o t + \theta(t)] \quad (1)$$

La simulación de este tipo de sistemas es particularmente costosa en términos de tiempo de procesamiento y volumen de información, ya que requiere del muestreo de la señal a una tasa al menos igual o superior a la frecuencia de Nyquist. Otro método más eficiente, lo constituye la simulación por medio de la aplicación del Teorema de la Envolvente Compleja [4], que permite el procesamiento de señal en su banda base (3), donde $i(t)$ y $q(t)$, son las componentes en fase y en cuadratura, respectivamente; lo que permite procesar información con requerimientos inferiores a los de banda pasante.

$$s(t) = \text{Re} \{ g(t) e^{j\omega_o t} \} \quad (2)$$

$$g(t) = |i(t) + jq(t)| e^{j\theta(t)} \quad (3)$$

$$\theta(t) = \tan^{-1} \left[\frac{q(t)}{i(t)} \right] \quad (4)$$

En general la simulación de un sistema de modulación digital requiere del procesamiento de la señal mensaje por parte de tres elementos fundamentales: el modulador, el canal de comunicaciones y el demodulador [4].

La arquitectura de un sistema de modulación digital otorga una alta posibilidad a la paralelización de su proceso (Figura 1), ya que el mismo posee las siguientes características:

- La información es procesada en bloques de datos de longitud fija, conocidos como trama.
- Las tramas son generadas estadísticamente independientes entre sí.

- Las tramas de un modulador digital y de su correspondiente receptor son de igual longitud.
- La señal de ruido en un canal de comunicaciones es estadísticamente independiente a la señal proveniente del modulador digital.
- La señal de ruido puede ser procesada en bloques de igual longitud a la de una trama de comunicaciones, afectando a una trama por vez.
- La señal modulada digitalmente s y la señal de ruido n poseen componentes en fase (s_i, n_i) y en cuadratura (s_q, n_q).
- Las componentes en fase y en cuadratura son ortogonales entre sí.

Un ejemplo de una sección de un programa para un sistema como el presentado en la Figura 1, se escribiría según indica la Figura 2. La definición de las regiones R y L establecen los espacios de memoria de operación de los datos, la primera es una matriz

delimitada por dos subíndices p y N que indican el número de niveles de energía y el número de símbolos por trama, respectivamente; la región L es un vector también definido por p . Las operaciones aritméticas $+$, $-$, $*$, $/$ son del tipo elemento a elemento; eliminando la costosa necesidad de algoritmos específicos para estas operaciones. Adicionalmente funciones matemáticas tales como $\sin$, $\cos$, atan2 o relaciones $=$, operan elemento a elemento en la matriz [3].

ZPL carece de generadores de números aleatorios por lo que se desarrollaron rutinas especializadas para generar estos valores y posteriormente variables aleatorias con distribuciones uniforme, gaussiana y rayleigh utilizando lenguaje C.

El operador de reducción $+\ll$, permite la realización de la suma acumulada de los errores producidos en una trama por cada nivel de energía en el vector `Error_Simb`. El código mostrado anteriormente puede ser modificada para manejar múltiples tramas agregando un comando de control de flujo como el `while` o `for`.

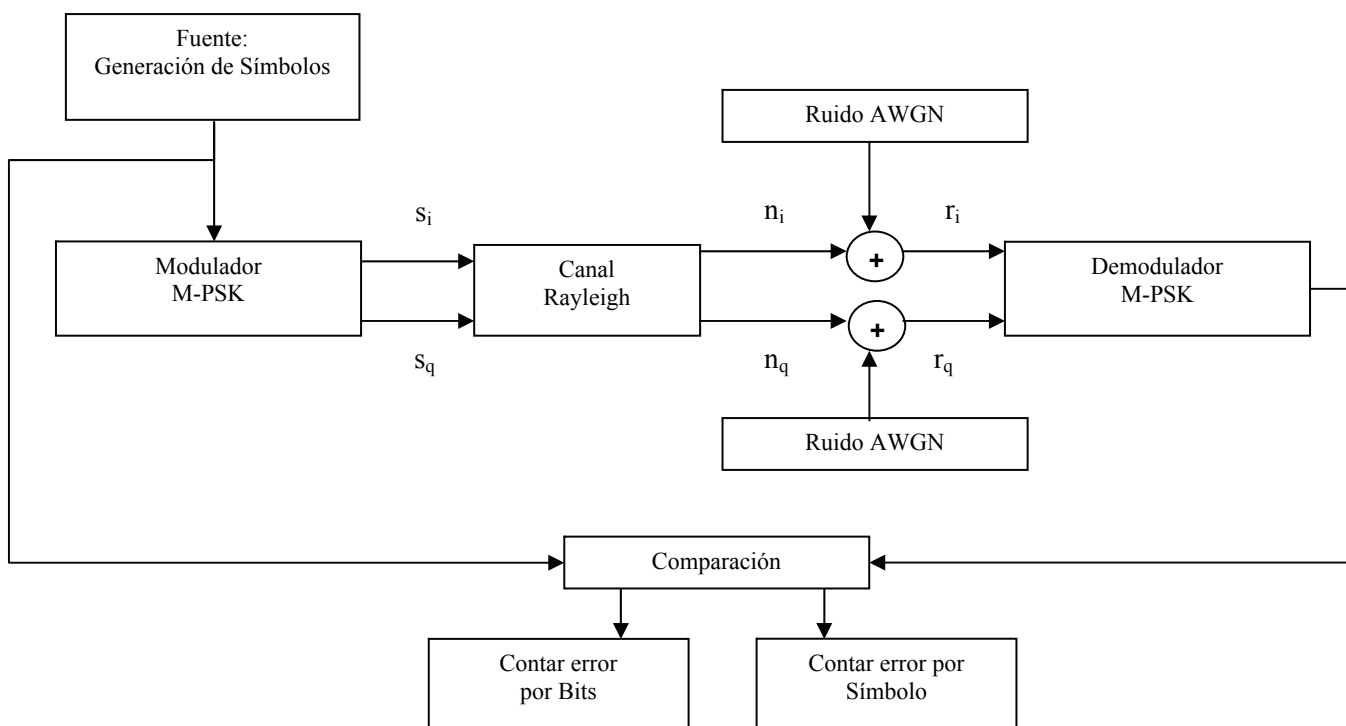


Figura 1. Diagrama de bloques de un simulador Montecarlo de tasa de error de bit y símbolos, para un modulador/demodulador digital M-PSK.

```

region R = [1..p, 1..N]; -- Región de N símbolos a transmitir y
                        -- recibir por p niveles de energía
    L = [1..p, 1]; -- Región de vector de salida por niveles
                        -- de SNR.

procedure psk_zpl();
var    Simb_Tx, Simb_Rx      : [R] double; --Variables de Tx y Rx
        N_I, N_Q, I, Q       : [R] double; --Variables en fase (I)
                                         --, Cuadratura (Q) y
                                         -- ruido gaussiano N_I , N_Q

        Phi_Tx, Phi_Rx       : [R] double; --Fases de Tx y Rx
        Aleatoria_uni        : [R] double; -- Variable Aleatoria
                                         -- Uniforme

        Dist_Ray              : [R] double; --Variable distribución
                                         -- Rayleigh

        Semilla1              : [R] integer; --Semilla para generadores
                                         -- de Variables
                                         -- Aleatorias

        Temp1, Temp2          : [R] double; -- Variables Temporales
        SNR                   : [L] double; -- Niveles de energía en dB
        Error_Simb            : [L] double; --Contador de errores
        M                     : integer;   --Número de símbolos

begin
[L] Error_Simb := 0;          --Iniciando vector contador de Errores
.
.
[R] Aleatoria_uni := gen_uniforme(Semilla1);
    N_I := gen_gaussiano_I(Semilla1,SNR);
    N_Q := gen_gaussiano_Q(Semilla1,SNR);
    Dist_Ray :=gen_rayleigh(Semilla1);
    Simb_Tx := floor(M*Aleatoria_uni);
    Phi_Tx :=2*pi*(Simb_Tx)/M;
    I:=Dist_Ray*cos(Phi_Tx)+N_I
                                         Q:=Dist_Ray*sin(Phi_Tx)+N_Q

    Phi_Rx := atan2(Q,I);
    if Phi_Rx < 0 then
        Phi_Rx := Phi_Rx + 2*m_pi;
    end;
    Temp1 := floor(Phi_Rx*M/(2*m_pi));
    Simb_Rx := Phi_Rx*M/(2*m_pi);
    Temp1 := Simb_Rx - Temp1;
    if Temp1 > 0.5
        Simb_Rx := Simb_Rx + 1;
    if Simb_Tx = Simb_Rx then          -- comparación entre Tx y Rx
        Temp1 :=0;                    -- No error
    else
        Temp1 :=1;                    -- Error
[L] Error_Simb := +<<[R] Temp1 + Error_Simb; -- Contador de Errores
.
end;

```

Figura 2. Ejemplo de código en ZPL para la simulación Montecarlo de rata de error de bit y símbolos, para un modulador/demodulador digital M-PSK.

4. SIMULACIONES Y RESULTADOS

4.1 El Cluster Nimbus

Para este estudio fue utilizado el cluster NIMBUS, el cual está ubicado en el IMYCA (Instituto de Matemática y Cálculo Aplicado) y consta de cinco procesadores Intel Pentium IV de 2.4 GHz de velocidad conectados por una red de alta velocidad del tipo Gigabit Ethernet. El nodo principal cuenta con 512 Mbytes de memoria RAM y los nodos secundarios con 256 Mbytes de RAM.

4.2 Las simulaciones

Los programas simulados fueron sistemas basados en sistemas de modulación M-PSK ($M=2, 4, 8$ y 16) y modulación de amplitud en cuadratura (*Quadrature Amplitud Modulation*) M-QAM ($M=4, 16$ y 64) [4-8]. Se realizó el cálculo de tasa de error de bits (*Bit Error Rate*), BER hasta una probabilidad de 10^{-4} , para señales transmitidas en canales con desvanecimiento lento y plano tipo Rayleigh, contaminados con ruido gaussiano blanco (*Additive White Gaussian Noise*), AWGN. Para cada caso se generaron al menos 21 puntos de relación señal a ruido (*Signal to Noise Ratio*) SNR, contando al menos 100 errores símbolos por punto de energía. Ambos simuladores fueron programados en ZPL, en lenguaje C (M-PSK) y en C++ (M-QAM).

4.3 Los resultados

Uno de los resultados más importantes consistió en una reducción notable del número de líneas requeridas por cada tipo de programa, en el cual los programas codificados en ZPL obtuvieron resultados notablemente menores (Figuras 3 y 4), cuando son comparados con los lenguajes C y C++.

Los tiempos de procesamiento observados (Figuras 5, 6, 7 y 8) para cada caso (M-PSK y M-QAM) muestran el mismo patrón de decaimiento, a medida que aumentan el número de procesadores para el caso de los programas basados ZPL, mejorando los tiempos de cómputo observados en los programas basados en C y C++ para un solo procesador.

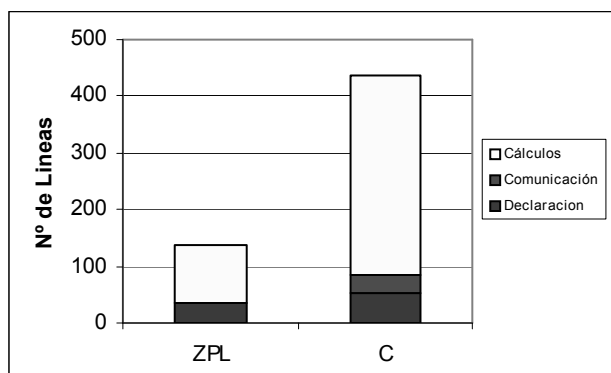


Figura 3. Número de líneas de código para programa M-PSK.

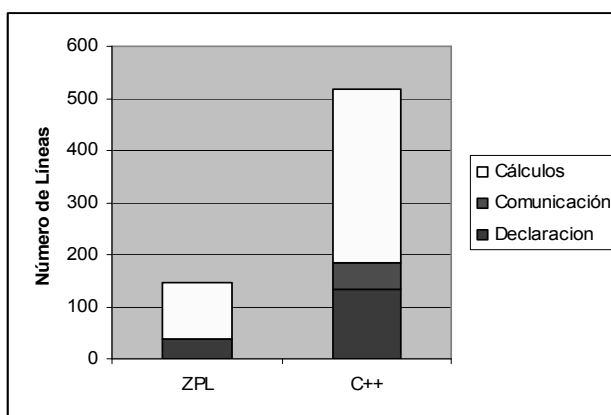


Figura 4. Número de líneas de código para programa M-QAM.

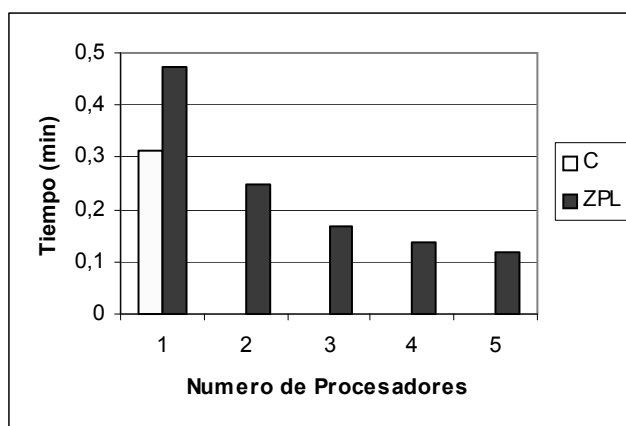


Figura 5. Tiempos de ejecución para simulador BPSK (2-PSK) programados en ZPL y C.

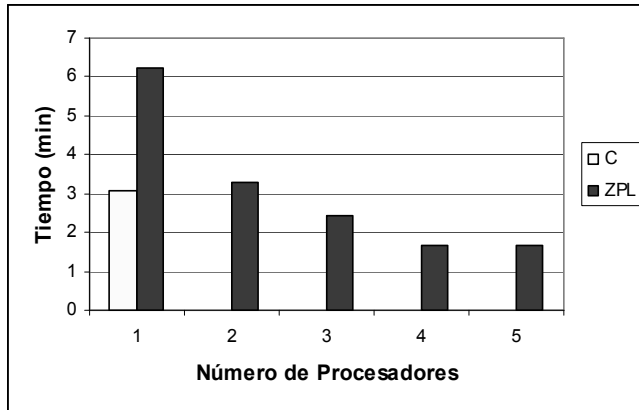


Figura 6. Tiempos de ejecución para simulador 16-PSK, programados en ZPL y C.

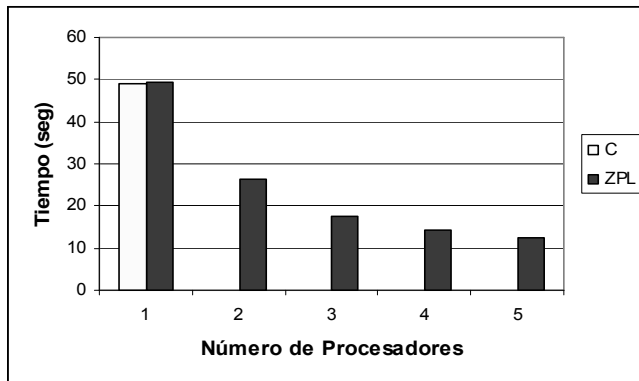


Figura 7. Tiempos de ejecución para simulador 4-QAM, programados en ZPL y C++.

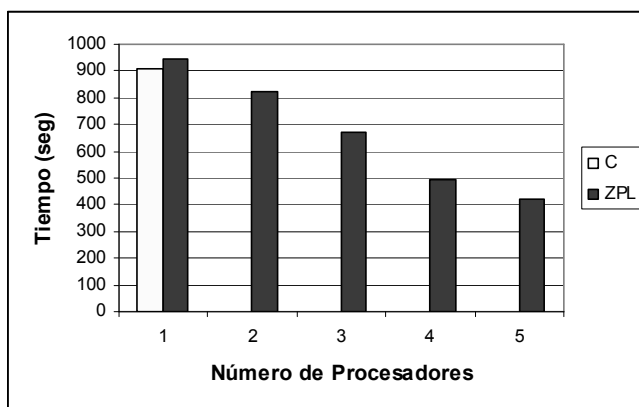


Figura 8. Tiempos de ejecución para simulador 16-QAM, programados en ZPL y C++.

Los resultados obtenidos hacen notar la potencialidad de ZPL como lenguaje de programación de simulaciones para sistemas de modulación digital, ya que evidencian una reducción de un 67% en el número de líneas de programación comparado con el programa escrito en lenguaje C y de un 71% versus el programa en C++. Adicionalmente, los tiempos de cómputo mejoran a partir de la incorporación de dos o más procesadores; superando los programas escritos en lenguajes secuenciales (C / C++). Este tipo de tendencia ya había sido reportada en [9], para otro tipo de aplicaciones, lo que confirma la versatilidad del lenguaje en el manejo de arreglos de datos en paralelo y su correspondiente distribución entre los múltiples procesadores que conforman el cluster.

5. CONCLUSIONES

El lenguaje ZPL puede constituirse como una poderosa herramienta de simulación en el campo de las telecomunicaciones, ya que permite realizar en forma exitosa la implementación de este tipo de programas. El programador en este lenguaje no requiere de un conocimiento importante acerca de paralelismo computacional, ya que el lenguaje no necesita en forma explícita la definición y asignación de recursos de cómputo y procesamiento en un cluster de computadoras.

El lenguaje ZPL ha demostrado ser una herramienta efectiva en la utilización de arquitecturas tipo cluster de computadoras, ya que es capaz de reducir los tiempos de cómputo a medida que se incorporen más procesadores. Los sistemas de modulación digital poseen un alto potencial de paralelización, lo que los convierten en candidatos ideales para la realización de programas de simulación basados en lenguajes de cómputo matricial. Se hace necesario la realización de futuros estudios orientados a verificar las propiedades de precisión de cálculo del lenguaje ZPL, así como el análisis de paralelización de otros sistemas de comunicación.

6. AGRADECIMIENTOS

Los agradecimientos son dirigidos al Instituto de Matemática y Cálculo Aplicado de la Universidad de Carabobo, por apoyar esta investigación.

7. REFERENCIAS BIBLIOGRÁFICAS

- [1] Parhami, B. "Introduction to parallel programming". Kluwer Academics. New Cork, 2002
- [2] ZPL Project at University of Washington. <http://www.cs.washington.edu/research/zpl>.
- [3] Snyder L. "A Programmers Guide to ZPL". Department of Computer Science and Engineering. University of Washington, 1999.
- [4] Proakis J. "Digital Communications 2nd Edition". McGraw-Hill. New Cork, 1998
- [5] Proakis J., Salehi M. "Communications Systems Engineering". Prentice hall. New Jersey, 1994
- [6] Proakis J., Salehi M. "Comtemporary Communications Systems using Matlab". PWS. Massachussets, 1998.
- [7] Peebles, P. "Digital Communication Systems". Prentice hall. New Jersey, 1987 .
- [8] Stark H., Woods J., "Probability, Random, Processes and Estimation Theory for Engineers 2nd Edition". Prentice Hall, 1994.
- [9] Canning J., Rey D., Stubblefield J. "Porting the Parallel Array Program Language to an Embedded Multicomputing System". Proccedings APL Madrid, 2002