

Implementación HW/SW de Arquitecturas de Clasificación de Paquetes en Lógica Reconfigurable

Luis Roberto Romano, Jairo Nicolás Trad

Universidad Nacional de Córdoba

Índice general

1. Introducción	1
2. Sistema	2
2.1. Sistemas Embebidos y Dispositivos Lógicos programables . .	2
2.1.1. Dispositivos Lógicos Programables	3
2.2. Redes de Computadoras	4
2.2.1. Modelo OSI	4
2.2.2. Ethernet	6
2.2.3. Búsqueda del prefijo más largo	6
2.3. FPGA	7
2.4. Sistemas embebidos	7
3. Arquitectura	9
3.1. Parte HW	9
3.1.1. Componentes del sistema	9
3.1.2. Módulo extractor de cabeceras	10
3.2. Parte SW	10
3.3. Parte SW	10
3.3.1. Búsqueda lineal	11
3.3.2. Búsqueda en Arbol unibit	11
3.3.3. Cache	12
4. Implementación	13
5. Resultados	14
5.1. Caso Algoritmos únicamente	14
5.2. Caso loopback	15
5.3. Implementación Completa	16
5.4. Cache	20

<i>ÍNDICE GENERAL</i>	II
6. Conclusiones	21
Bibliografía	22
Apendices	23
6.1. Configuración del Hardware	23

Índice de figuras

2.1. Diseño preliminar	3
2.2. Modelo OSI	5
2.3. PDUs en las distintas capas del modelo OSI	6
2.4. Sistema	8
2.5. Extractor de cabeceras	8
5.1. Retardo de Búsqueda LLU vs UTL	15
5.2. Caso Loopback para 1 y 15 palabras	16
5.3. Retardo mínimo LLU	17
5.4. Retardo promedio LLU	17
5.5. Retardo máximo LLU	18
5.6. Retardo mínimo UTL	18
5.7. Retardo promedio UTL	19
5.8. Retardo máximo UTL	19

Índice de cuadros

Capítulo 1

Introducción

Capítulo 2

Sistema

En este capítulo se introduce brevemente una serie de conceptos que fueron utilizados para la realización del presente proyecto integrador. No se pretende que el lector alcance una comprensión exhaustiva de los mismos, sino que tenga las herramientas necesarias para la correcta interpretación de los capítulos posteriores. De manera paralela se introducen los bloques básicos que formaran parte del sistema planteado.

2.1. Sistemas Embebidos y Dispositivos Lógicos programables

Un Sistema embebido es una combinación de hardware y software diseñado para realizar funciones dedicadas, generalmente en tiempo real. En algunos casos los sistemas embebidos no actúan de manera independiente y se encuentran integrados en un sistema o producto mayor. En la actualidad el 98 % de los microprocesadores fabricados tienen como destino algún sistema embebido mientras que solo el 2 % se destina a microprocesadores de propósito general.

El campo de aplicación para los sistemas embebidos es muy variado, desde dispositivos portátiles como reproductores MP3 o teléfonos celulares, hasta sistemas de control en centrales nucleares. No es posible caracterizar los componentes exactos de un sistema embebido ya que existen una gran cantidad de configuraciones posibles, pero dentro de los componentes fundamentales que se encuentran en la mayoría de los sistemas están:

- Microprocesador
- Memoria RAM

- Periféricos para Captura de datos y de Comunicación.

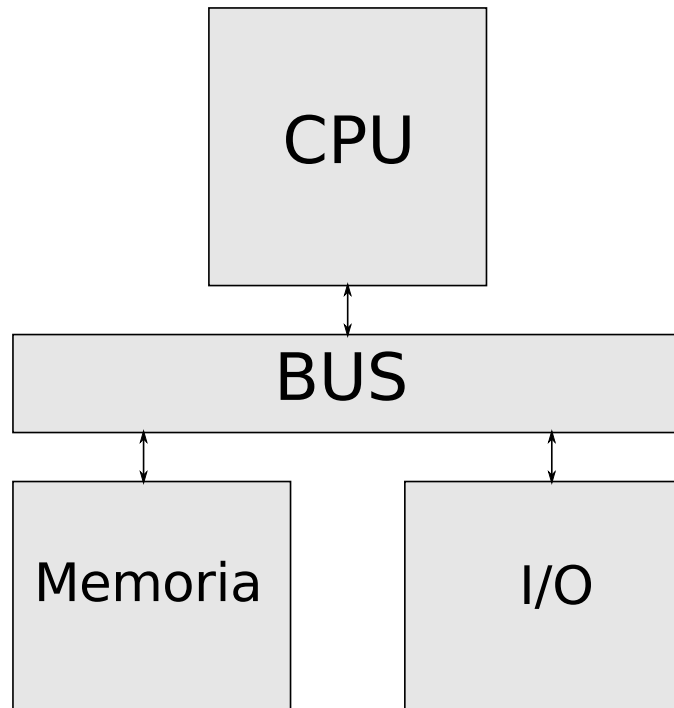


Figura 2.1: Diseño preliminar

El sistema embebido que será generado apartir de este proyecto no será la excepción y contará al menos con los componentes anteriormente nombrados. En la Figura 2.1 es posible ver el diseño preliminar del hardware de nuestro sistema.

2.1.1. Dispositivos Lógicos Programables

Los dispositivos lógicos pueden ser clasificados en dos categorías, fijos y programables. Los fijos son permanentes, están preparados para realizar una o una variedad específica de funciones y una vez manufacturados no pueden ser modificados. En cambio, los dispositivos lógicos programables, de ahora en adelante referenciados como PLDs, ofrecen como su nombre lo indica la posibilidad de ser reprogramados en cualquier momento con cualquier función lógica que el usuario requiere y que sea soportado por el dispositivo en cuestión.

Existen dos grandes tipos de PLDs, FPGAs y CPLDs. Los CPLDs ofrecen una capacidad lógica pequeña, pero con unas características de temporización predecibles lo que los hace dispositivos ideales para aplicaciones de control y cualquier otra tarea de tiempo real. Además su consumo de potencia reducido y su bajo costo los posicionan muy bien en el campo de los dispositivos portátiles. Por otro lado las FPGAs ofrecen una mayor densidad lógica y una mejor prerrománica. Actualmente es posible encontrar hasta 2 Millones de celdas lógicas en una FPGA comercial. Además estos dispositivos ofrece módulos embebidos como tranceptores, memoria o microprocesadores. Todo esto le permite a las FPGAs ser usadas en una amplia variedad de aplicaciones como procesamiento de datos, instrumentación, telecomunicaciones y procesamiento de señales.

2.1.2. Sistemas embebidos en Logica Reprogramable

§

2.2. Redes de Computadoras

2.2.1. Modelo OSI

Es el modelo de red descriptivo creado por la Organización Internacional para la Estandarización en el año 1984. Sirve como marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

Se basa en el concepto de *capa*: Un conjunto de rutinas (y en algunos casos, hardware) que cumplen una serie de funciones determinadas. Cada capa ofrece servicios a las q se ubican por encima y hace uso de servicios ofrecidos por las de abajo.

El modelo está conformado por las siguientes capas:

- Aplicación: Representa el punto de acceso de las aplicaciones que hacen uso del esquema de transmisión de datos.
- Presentación: Relacionada con la representación de los datos.
- Sesión: se encarga de mantener y controlar el enlace establecido entre dos computadores que están transmitiendo datos.
- Transporte: encargada de efectuar el transporte de los datos (que se encuentran dentro del paquete) de la máquina origen a la de destino, independizándolo del tipo de red física que se esté utilizando.

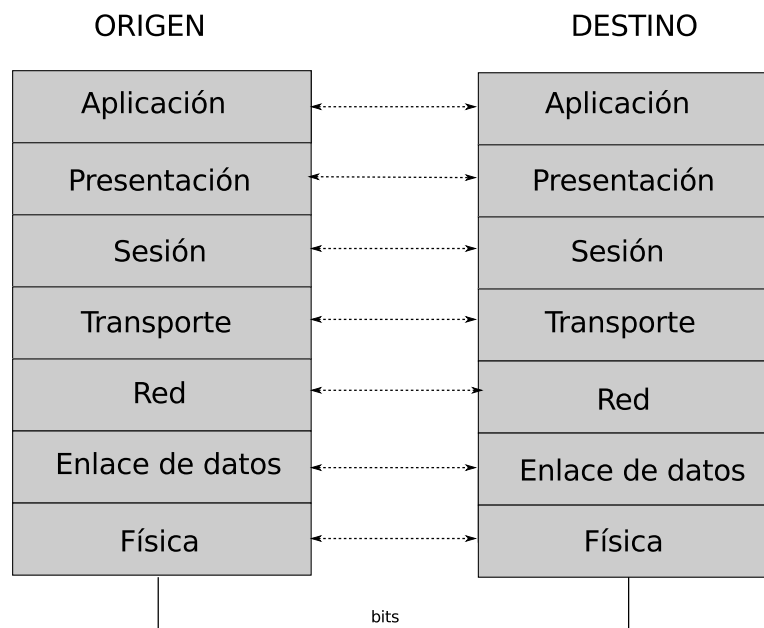


Figura 2.2: Modelo OSI

- **Red:** Se encarga de identificar el enrutamiento existente entre una o más redes.
- **Enlace de datos:** se ocupa del direccionamiento físico, de la topología de la red, del acceso al medio, de la detección de errores, de la distribución ordenada de tramas y del control del flujo.
- **Física:** se encarga de las conexiones físicas de la computadora hacia la red, tanto en lo que se refiere al medio físico como a la forma en la que se transmite la información.

Unidad de datos del protocolo

Es un bloque de datos que cada capa utiliza para el intercambio de información con su par en el sistema destino. Suele denominarse PDU, por las siglas en inglés de *Protocol Data Unit*.

Cuando llega la información por parte del usuario, la capa de aplicación le agrega información de control y se conforma la PDU de dicha capa, la cual es transferida a la siguiente, que va a hacer lo mismo. De esta manera, cada capa agrega información a la PDU recibida de la anterior y transfiere

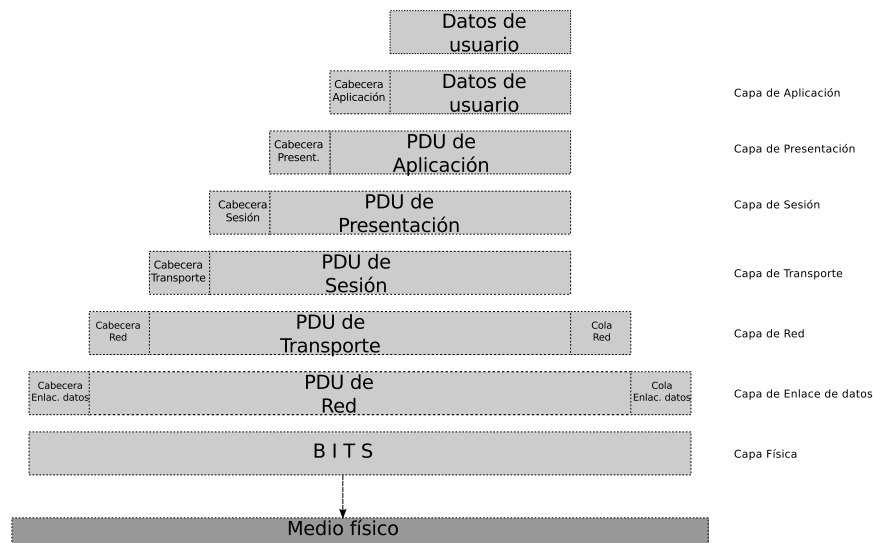


Figura 2.3: PDUs en las distintas capas del modelo OSI

la PDU resultante hacia la de abajo. Esto se repite hasta llegar al nivel físico, en el cual la información se transmite desde el sistema origen hacia el sistema destino en forma de bits.

Una vez que los datos han llegado a destino, comienza el proceso inverso. Los bits en capa física son transferidos a la capa de enlace de datos, la cual los delimita conformando tramas (PDUs de dicha capa). En base a la información de control, cada capa va transfiriendo su PDU hacia arriba, hasta que los datos llegan a la aplicación de destino. Se dice que la comunicación entre capas pares es transparente, ya que cada una de ellas no es consciente de los detalles de la implementación de las capas inferiores, tanto en el sistema origen como en el sistema destino. La figura 2.2 muestra este concepto.

2.2.2. Ethernet

Ethernet es un estándar de transmisión de datos para redes de área local. El mismo define las características del cableado y señalización en capa física, así como también el formato de trama en capa de enlace de datos.

2.2.3. Búsqueda del prefijo más largo

(... pendiente ...)

_-* Esto queda como reserva de datos que despues vmaos a usar****

Uno de los mayores cuellos de botella en los routers lo constituye el cómputo de del prefijo más largo para cada paquete entrante.

Implementar ciertos esquemas de clasificacion en hardware se ve limitado principalmente debido a 2 factores: La cantidad de memoria requerida y lla complejidad creciente de los mismos.

En tanto crece el trafico en las redes se ve la necesidad de implementar esquemas mas complejos de clasificacion.

Dichos esquemas tienen una implementación más sencilla en software. Dichas implementaciones estan ampliamente difundidas en la web. Esto hace innecesario tener que “adaptarlas” a un HDL. El paso es trivial (corregir esta redaccion)

2.3. FPGA

Son dispositivos lógicos programables cuya lógica interna puede ser re-configurada. Esta característica permite implementar un diseño propio, con la posibilidad de efectuar la cantidad necesaria de pruebas hasta llegar a los resultados deseados. Se puede instanciar componentes usando la lógica interna (ej microprocesadores, PLL, Memorias,etc).

2.4. Sistemas embebidos

Aunque se pone mucho foco en el diseño de los procesadores de proposito general en la realidad estos solo representan solo una pequeña proporcion de los procesadores efectivamente producidos cada año. Existe una especial motivacion en la industria por los denominados “Procesadores Heterogeneos” que integran sistemas dedicados con procesadores de proposito general.

(acá se podría poner algunas generalidades sobre Ethernet e IP)

(tambien se podria agregar algo del problema del prefijo más largo)

El sistema implementado en el presente trabajo consta de un micro-procesador NIOS2/f interconectado mediante un bus Avalon-MM a 5 componentes:

- PLL
- JTAG UART
- Iterfaz con SDRAM
- Timer

- Módulo extractor de cabeceras

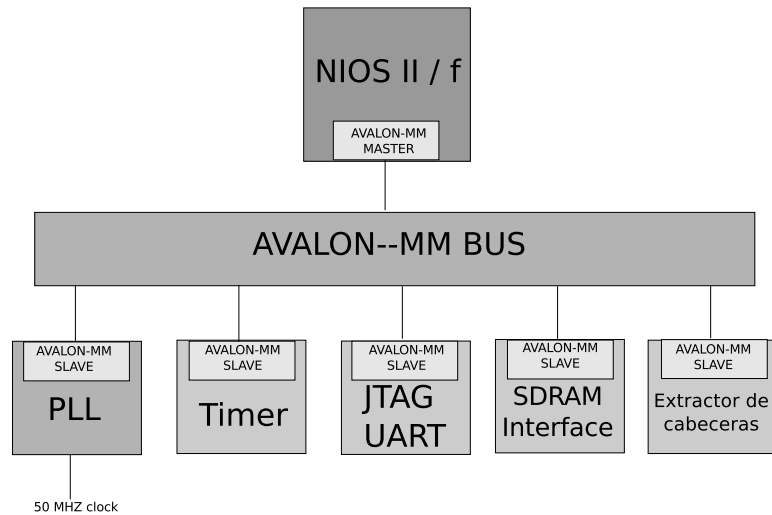


Figura 2.4: Sistema

A su vez, éste último está conformado por un generador de paquetes Ethernet conectado a una FIFO. Esta está a su vez conectada a un modulo denominado delay buffer, que está conectado al modulo uplink y al write output.. Uplink a su vez está conectado también a write output.

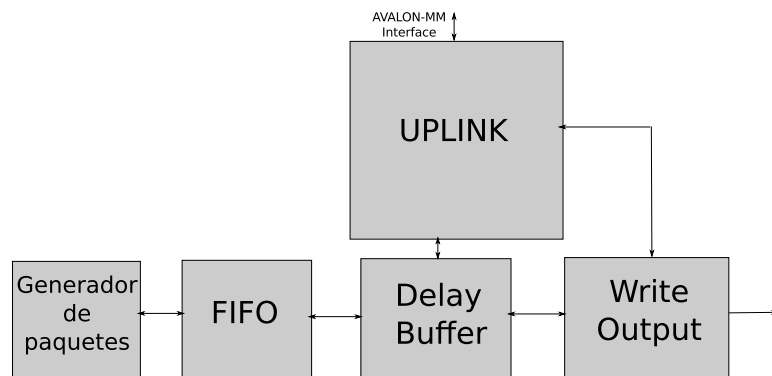


Figura 2.5: Extractor de cabeceras

Sobre el hardware descrito anteriormente se ejecuta un software de clasificación de paquetes, que se encuentra almacenado en una memoria RAM.

Capítulo 3

Arquitectura

3.1. Parte HW

3.1.1. Componentes del sistema

NIOS II

Es un microprocesador softcore. Esto significa que el mismo es instanciado usando la lógica propia de la FPGA. En este diseño, ejecuta un software de clasificación de paquetes que se almacena en una memoria SDRAM en la placa de desarrollo.

PLL

Este módulo toma como entrada una señal de clock de 50 MHz de frecuencia y la bifurca en 2: Una de ellas alimentará al módulo que oficia de interfaz con la memoria SDRAM y la otra hará lo propio con el resto de los componentes del sistema. Estas señales están defasadas entre sí 60° con el fin de evitar el skew producido por la diferencia entre la llegada del clock a la memoria y al resto del sistema.

Timer

Módulo utilizado para llevar estadísticas de retardo dentro del software.

JTAG UART

Este módulo permite interactuar con el sistema vía USB. Esto implica tanto la configuración de la FPGA, como también la posibilidad de ver la

ejecución del software en una consola.

Interfaz con SDRAM

Tiene la función de interconectar al sistema con la memoria SDRAM de la placa de desarrollo.

Extractor de cabeceras

Este módulo extrae cabeceras de paquetes Ethernet y las envía al software, donde son procesadas y devueltas. Su funcionamiento se detallará a continuación.

3.1.2. Módulo extractor de cabeceras

(..esta parte completala vos...)

3.2. Parte SW

3.3. Parte SW

Se implementaron 2 algoritmos de clasificacion: Búsqueda lineal y Búsqueda en árbol unibit.

El software utilizado para realizar las pruebas consistió en 2 proyectos por separado. Uno para cada tipo de búsqueda en la tabla.

El mismo fue desarrollado en lenguaje c++, por presentar éste ciertas facilidades para las implementaciones llevadas a cabo. Puntualmente se sacó ventaja de un STL container (list) para implementar la búsqueda lineal. Esta plantilla cuenta con, entre otras cosas, la posibilidad de ordenar la lista con sólo una llamada a función.

Para efectuar el intercambio de datos con el hardware se hizo uso de las macros IOWR e IORD, las cuales escriben y leen respectivamente los datos hacia/desde un componente conectado al bus Avalon MM. La razón de haber usado dichas macros yace en el hecho de que las mismas no son cacheadas". Esta característica se torna indispensable en este diseño, ya que en el mismo no se puede leer un dato sin saber si está verdaderamente disponible en el bus.

3.3.1. Búsqueda lineal

Se implementó en una lista enlazada, creada a partir del template list de c++. Los nodos de la lista contienen 3 campos:

- Dirección de red (entero de 32 bit sin signo)
- Máscara de red (entero de 32 bit sin signo)
- Identificador de decisión (entero de 32 bit con signo)

Como se le dió prioridad a los prefijos de red más largos, se debió sobrecargar el operador de comparación (`<`) para que la función sort pudiese ordenar en base a la longitud de máscara. De esa manera, los nodos que contenían valores de máscara más grandes quedaban en las primeras posiciones de la lista.

Cuando la función encargada del lookup recibe una dirección IP de destino, realiza los siguientes pasos:

- Coloca un iterador al comienzo de la lista.
- Realiza un AND con el valor de máscara del nodo que está siendo apuntado. Si el resultado de la operación es igual al valor de dirección de red de dicho nodo, entonces se retorna con el valor identificador de decisión. En otro caso, continúa la búsqueda en el siguiente nodo.

3.3.2. Búsqueda en Arbol unibit

Se implementó una clase en la cual se definieron las características de los nodos del arbol, como así tambien las operaciones de inserción y búsqueda. Cada nodo cuenta con los siguientes campos:

- gw: es un identificador de la decisión a tomar. En los nodos no asociados a una decision, tiene el valor estipulado en la macro NONE.
- zero / one: Son punteros a nodo, asociados a los bits 0/1 del prefijo que se esté leyendo.

En este contexto, pueden existir 2 tipos de nodo:

- Común: Está asociado a la macro NONE. La misma lo diferencia del nodo decisión.

- Decisión: Contiene en el campo gw un valor que identifica a la decisión a tomar.

El algoritmo de búsqueda toma como entrada la dirección IP de destino del paquete a clasificar. Luego de ello, va haciendo un testeo bit a bit de la misma, partiendo con un puntero de recorrido desde el nodo raíz. Si el bit de la dirección es 0 y el puntero zero está apuntando hacia algún nodo, el puntero de recorrido se mueve al nodo apuntado por el puntero zero. En caso contrario, se mueve al nodo apuntado por one (En caso de que exista). Esto se repite nodo a nodo, hasta que:

- El puntero de recorrido queda varado en un nodo decision, con lo cual se retorna el valor de gw. Ó
- El puntero de recorrido queda varado en un nodo común.

Contemplando esta última posibilidad, el algoritmo hace que en cada nodo se chequee si se trata de un nodo decisión. En dicho caso, se almacena el campo gw en una variable y se continua el recorrido. Si se da un caso en el cual el nodo de recorrido queda apuntando a un nodo comun y luego de testear un bit se determina que el mismo no tiene un nodo asociado (es decir, que alguno de los punteros zero / one esté en NULL) la funcion retorna la variable anteriormente mencionada.

3.3.3. Cache

Se implementó una cache directa. La misma consta de una tabla hash de 16 entradas. Las colisiones se resuelven por reemplazo directo. La misma fue testada con ambos algoritmos mencionados anteriormente. Para ello, se agregó una lógica adicional que consistió en:

- Al tomar una direccion IP, chequear primero si el valor de decisión se encuentra en caché.
- Si está, retornar dicho valor.
- En otro caso, efectuar el lookup y almacenar el valor de decisión en caché.

Capítulo 4

Implementación

Capítulo 5

Resultados

En este capítulo se presentan los datos obtenidos de la ejecución del proyecto bajo ciertas condiciones representativas, con la intención de validar la funcionalidad y también de encontrar los alcances y límites del mismo. Primero se estudiara el tiempo de respuesta de los algoritmos por separado, más tarde el rendimiento en la configuración mas simple, luego el sistema completo bajo condiciones de configuración varias y por ultimo el estudio de la mejora introducida por el uso de la cache.

5.1. Caso Algoritmos únicamente

Con la intención de obtener un gráfico que represente el rendimiento de los dos algoritmos implementados, se medirá el retardo de búsqueda en función de la posición en la tabla de enrutamiento, se realizaran las pruebas de manera independiente al modulo extracto de cabeceras. En el eje de las abscisas se expresa la ubicación en una tabla de 100 elementos y en las ordenadas se puede observar el tiempo de búsqueda en ciclos de reloj.

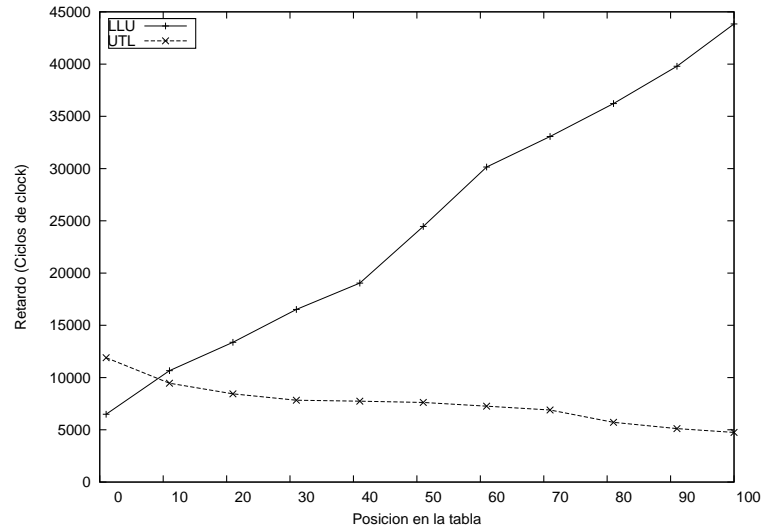


Figura 5.1: Retardo de Búsqueda LLU vs UTL

5.2. Caso loopback

Se estudia del caso loopback a los fines de encontrar los limites superiores de nuestro diseño. En este, el software solo se limita a recibir los datos e inmediatamente después confirma el procesamiento y envía los resultados de regreso al hardware. Se realizan las pruebas correspondientes para las dos versiones de Uplink. En el eje de las abscisas es posibles ver la cantidad de paquetes por segundo, el origen corresponde a la mayor velocidad a la que es posible transmitir sin perdidas. En las Ordenadas se puede observar la cantidad de paquetes perdidos en valores porcentuales, para obtener esta métrica se proceso una cantidad constante de paquetes, 9000, y luego se contrasto este valor con un contador global que el Generador estampa en la ultima palabra de cada paquete. Así se calculo la cantidad paquetes perdidos, sobre la cantidad total de paquetes generados. Este mismo sistema es el usado en todos los gráficos posteriores.

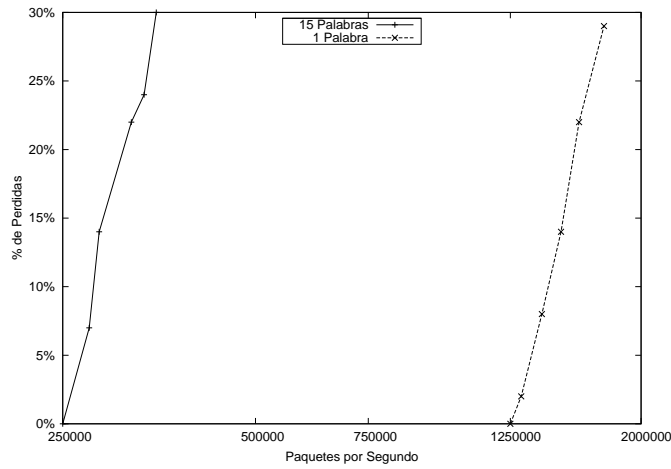


Figura 5.2: Caso Loopback para 1 y 15 palabras

5.3. Implementación Completa

Se verificara el rendimiento del sistema implementado de manera completa. Se consideran seleccionaran tres puntos en las curvas que indican los tiempos de accesos del algoritmos, un punto mínimo que corresponde al menor tiempo de acceso, un punto promedio que ejercita 10 entradas equidistantes a lo largo de la tabla y un punto máximo que indica el peor tiempo de acceso posible para un Algoritmo dado.

Linear Lookup

En la figura 5.3 se puede observar que la diferencia en la cantidad de paquetes que pueden ser transmitidos sin errores en el mejor caso entre el modulo que envía una palabra al procesador y el que envía toda la cabecera, es considerable, mientras que a medida que aumenta la profundidad de búsqueda en la tabla estos valores convergen, como es posible ver en la figuras 5.4 y 5.5, lo que era esperable ya que para accesos muy lentos a la tabla el retardo introducido por el Hardware se vuelve despreciable.

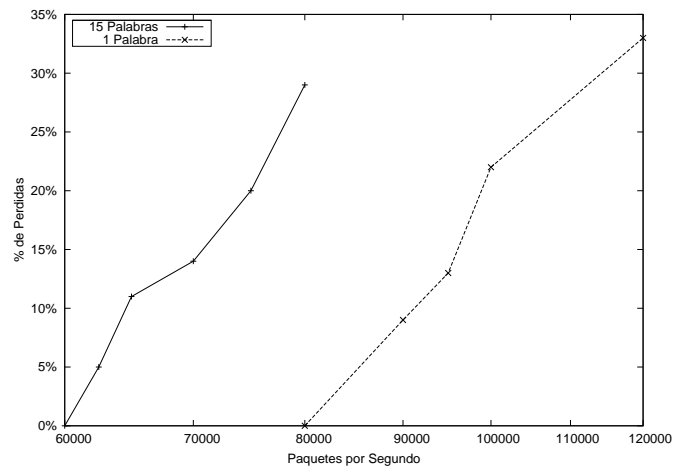


Figura 5.3: Retardo mínimo LLU

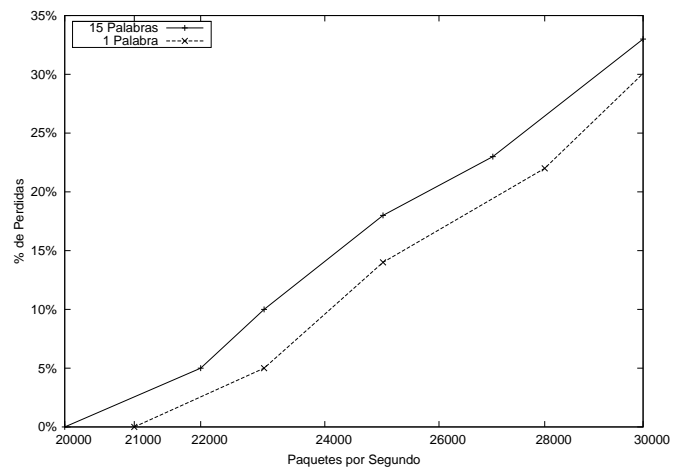


Figura 5.4: Retardo promedio LLU

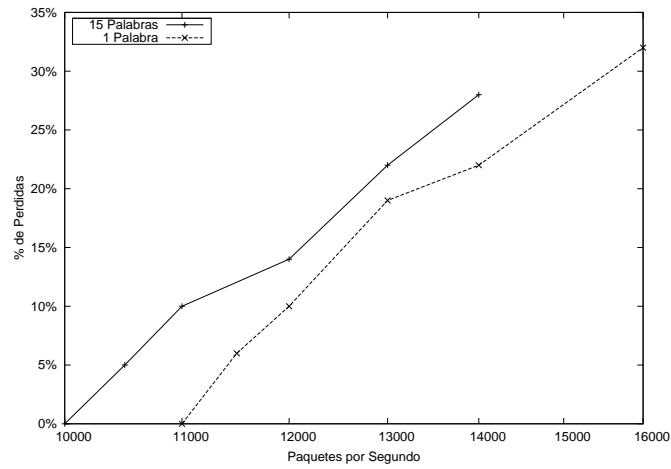


Figura 5.5: Retardo máximo LLU

Unibit Trie Lookup

En los gráficos que corresponden al Unibit Trie Lookup es posible observar que existe una menor diferencia entre la máxima cantidad de paquetes que pueden ser transmitidos sin error en cada uno de los 3 puntos elegidos. Así como también la diferencia entre enviar el paquete entero y solo la IP destino se reduce, lo que da la pauta de que cuando el tiempo de acceso es uniforme el impacto de la mejoras del Hardware tiende a ser menor.

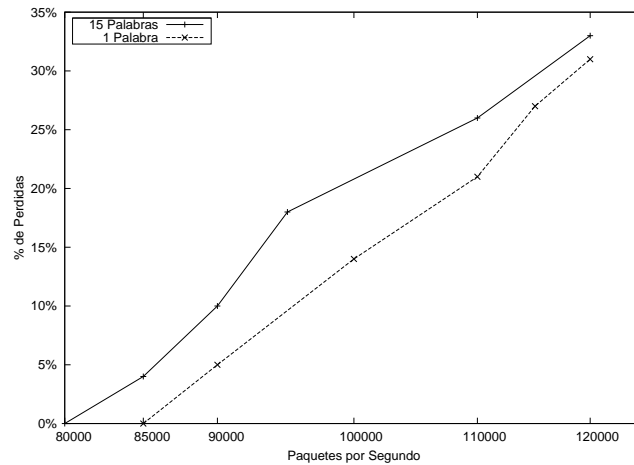


Figura 5.6: Retardo mínimo UTL

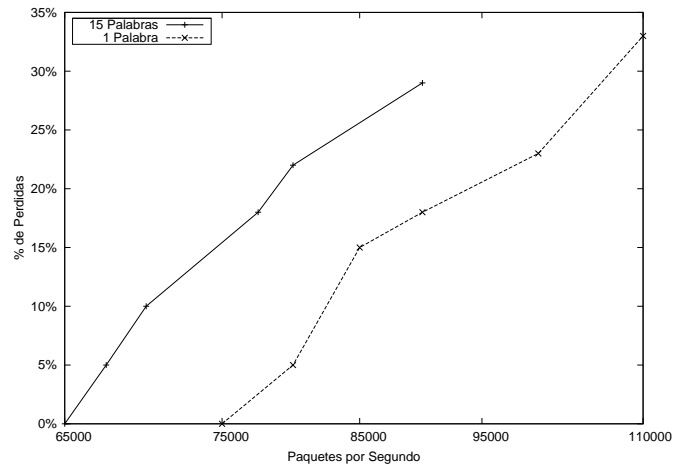


Figura 5.7: Retardo promedio UTL

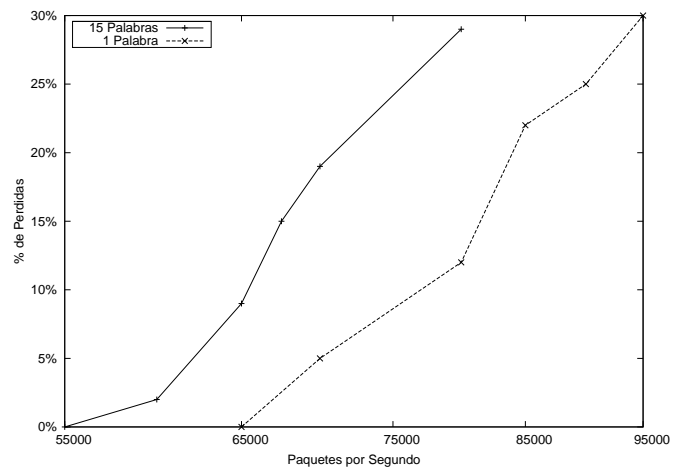


Figura 5.8: Retardo máximo UTL

(Falta la comparativa inter-algoritmos)

5.4. Cache

(Falta Mejorar y agregar el grafico) Se probara el funcionamiento del sistema incluyendo la cache implementada en software. Para ello, se efectuara una toma de datos en la cual se compararan 2 sucesiones: la primera de ella resultara en fallos al no estar presente ninguno de los valores.

Puede notarse que, en el caso de que la búsqueda termine en un fallo, se introduce un retardo adicional (si se compara con el escenario en el cual no existe la cache). Ello se debe a la búsqueda que se efectúa en la cache antes de hacerlo propiamente en la tabla de ruteo, tanto en la implementación linked-list como en la implementación unibit-trie.

Capítulo 6

Conclusiones

Bibliografía

Appendices

6.1. Configuración del Hardware

Feature	Description
FPGA	<ul style="list-style-type: none">■ Cyclone II EP2C35F672C6 with EPCS16 16-Mbit serial configuration device.
I/O Interfaces	<ul style="list-style-type: none">■ Built-in USB-Blaster for FPGA configuration■ Line In/Out, Microphone In (24-bit Audio CODEC)■ Video Out (VGA 10-bit DAC)■ Video In (NTSC/PAL/Multi-format)■ RS232■ Infrared port■ PS/2 mouse or keyboard port■ 10/100 Ethernet■ USB 2.0 (type A and type B)■ Expansion headers (two 40-pin headers)

Memory	<ul style="list-style-type: none"> ■ 8 MB SDRAM, 512 KB SRAM, 4 MB Flash ■ SD memory card slot
Displays	<ul style="list-style-type: none"> ■ Eight 7-segment displays ■ 16 x 2 LCD display
Switches and LEDs	<ul style="list-style-type: none"> ■ 18 toggle switches ■ 18 red LEDs ■ 9 green LEDs ■ Four debounced pushbutton switches
Clocks	<ul style="list-style-type: none"> ■ 50 MHz clock ■ 27 MHz clock ■ External SMA clock input

La FPGA incluida en la placa es una Cyclone II EP2C35 cuyas especificaciones son:

Feature	Description
LEs	33216
Total RAM bits	483840
Embedded multipliers	35
PLLs	4
Maximum user I/O pins	475