

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. FPGA	2
2.2. Sistemas embebidos	2
3. Arquitectura	5
3.1. Parte HW	5
4. Implementación	8
5. Resultados	9
5.1. Configuración del Hardware	9
5.1.1. Caso loopback	11
5.1.2. Caso Algoritmos unicamente	12
5.1.3. Completa.	13
5.1.4. Cache.	16
6. Conclusiones	17
Bibliografía	18
Apendices	19

Índice de figuras

2.1. Sistema	3
2.2. Extractor de cabeceras	4
5.1. Caso Loopback para 1 y 15 palabras	11
5.2. Retardo de Búsqueda LLU vs UTL	12
5.3. Retardo Minimo LLU	13
5.4. Retardo promedio LLU	14
5.5. Retardo Minimo LLU	14
5.6. Retardo Minimo UTL	15
5.7. Retardo promedio UTL	15
5.8. Retardo Minimo UTL	16

Índice de cuadros

Capítulo 1

Introducción

Capítulo 2

Sistema

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)

2.1. 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).

2.2. 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 mo-

tivacion 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

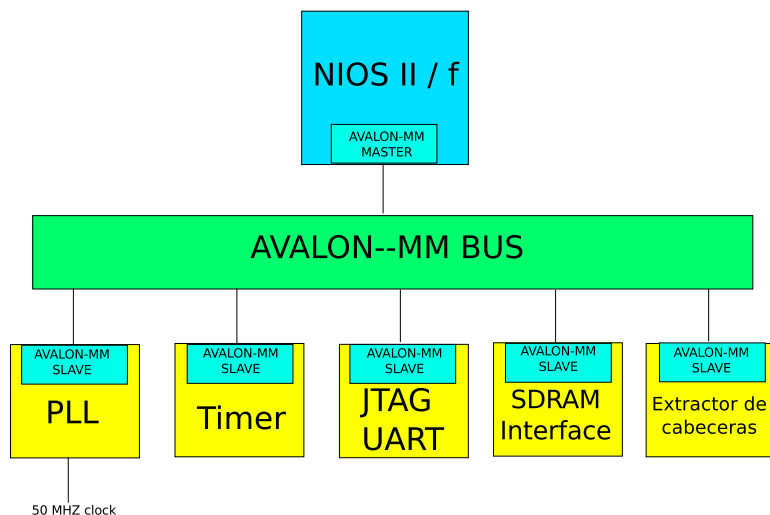


Figura 2.1: Sistema

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.

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

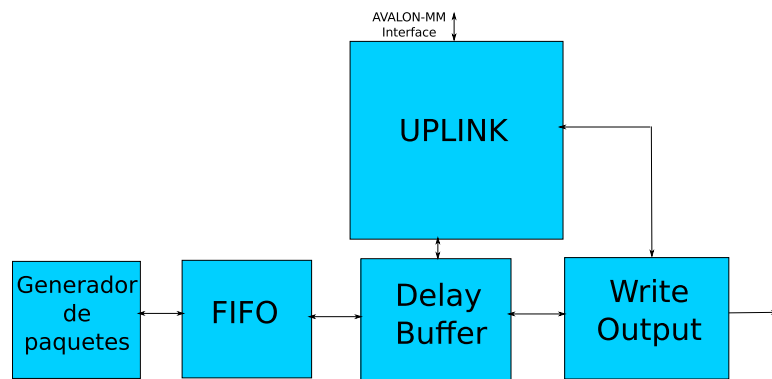


Figura 2.2: Extractor de cabeceras

Capítulo 3

Arquitectura

3.1. Parte HW

(Acá hay q poner una descripción detallada de los módulos mencionados en el capítulo anterior)

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.

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 (\lessdot) 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.

Busqueda 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 algun 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.

Capítulo 4

Implementación

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos de la ejecución del proyecto bajo ciertas condiciones representativas, con la intención de validar la funcionalidad y también de encontrar los puntos fuertes y las falencias del mismo.

5.1. 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

5.1.1. Caso loopback

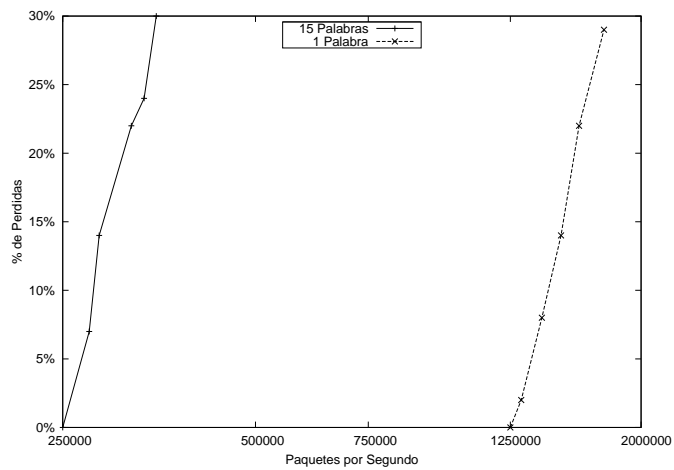


Figura 5.1: Caso Loopback para 1 y 15 palabras

En primera instancia nos ocupamos del caso loopback a los fines de

encontrar los límites superiores de nuestro diseño. En este caso, 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 posible ver la cantidad de paquetes por segundo, el origen corresponde a la mayor velocidad a la que es posible transmitir sin pérdidas. En las Ordenadas se puede observar la cantidad de paquetes perdidos en valores porcentuales, para obtener esta métrica se procesa una cantidad constante de paquetes, 9000, y luego se contrasta este valor con un contador global que el Generador estampa en la última palabra de cada paquete. Así se calcula la cantidad de paquetes perdidos, sobre la cantidad total de paquetes generados. Este mismo sistema es el usado en todos los gráficos posteriores.

5.1.2. Caso Algoritmos únicamente

Se estudiará la performance de los algoritmos aplicados, midiendo el retardo de lookup en función de la posición en la tabla de ruteo, se realizarán las pruebas de manera independiente al módulo extracto de cabeceras, para así obtener una medida más exacta del rendimiento.

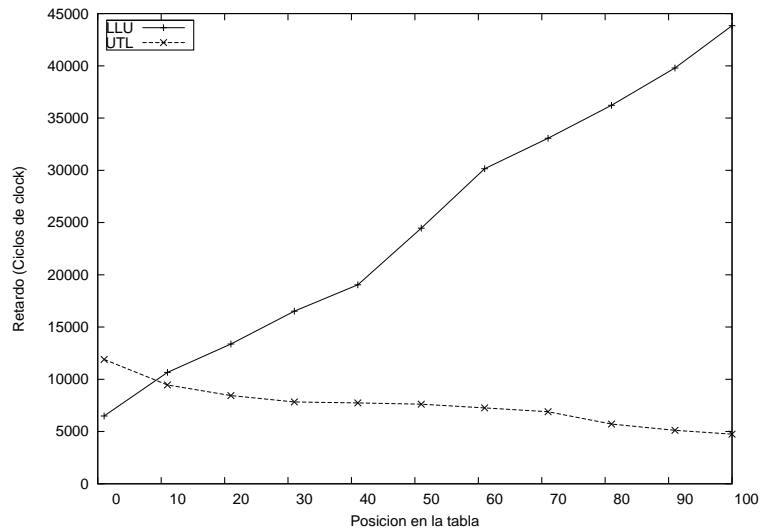


Figura 5.2: Retardo de Búsqueda LLU vs UTL

5.1.3. Completa.

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

Linear Lookup

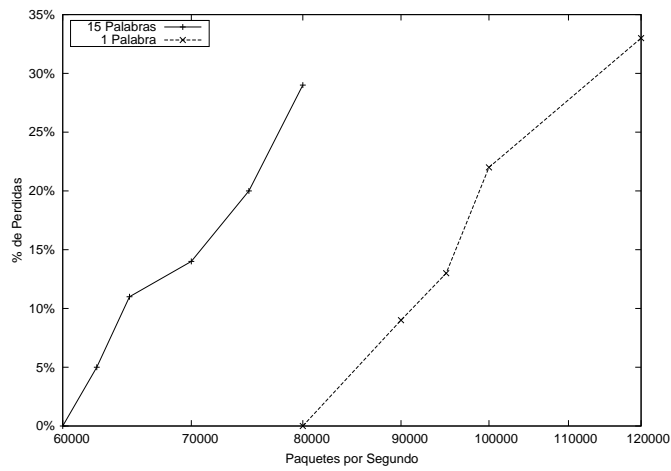


Figura 5.3: Retardo Mínimo LLU

Se puede observar que la diferencia entre la cantidad de paquetes que pueden ser transmitidos sin errores en el mejor caso(Mínimo) entre el modulo que envía una palabra al procesador y el que envía toda la cabecera, es considerable, mientras que a medida de que se avanza en el recorrido de la tabla estos valores convergen, lo que era esperable ya que para accesos muy lentos a la tabla, el retardo introducido por el Hardware se vuelve despreciable.

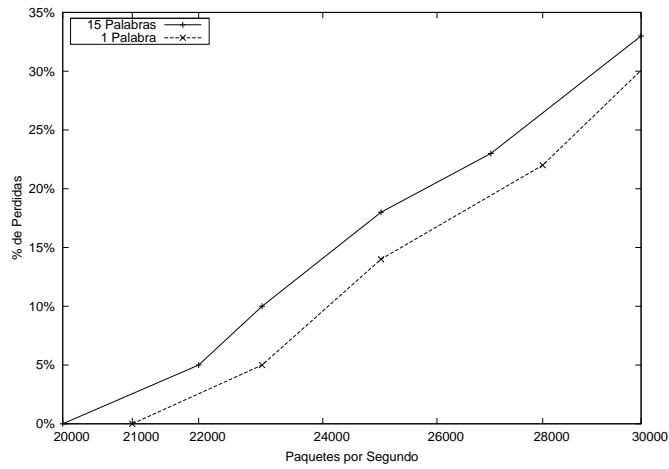


Figura 5.4: Retardo promedio LLU

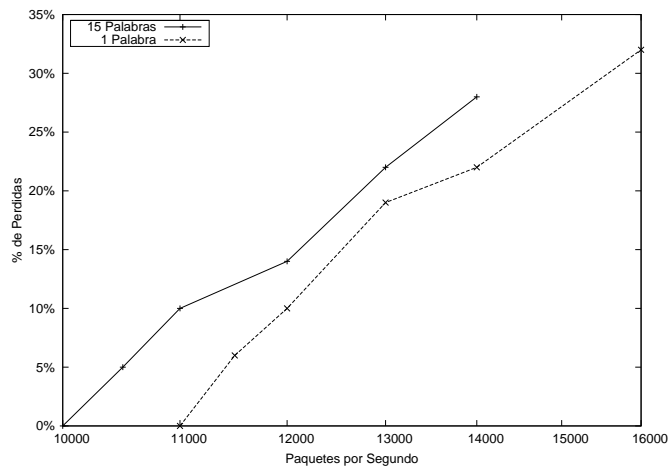


Figura 5.5: Retardo Mínimo LLU

Unibit Trie Lookup

En los gráficos que corresponden al Unibit trie LookUp se puede observar que existe una menor diferencia entre la máxima cantidad de paquetes que es posible transmitir 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 vuelve mas imperceptible, lo que da la pauta de que a medida de que el tiempo de acceso se vuelve mas uniforme la mejora introducida en el

Hardware se vuelve despreciable.

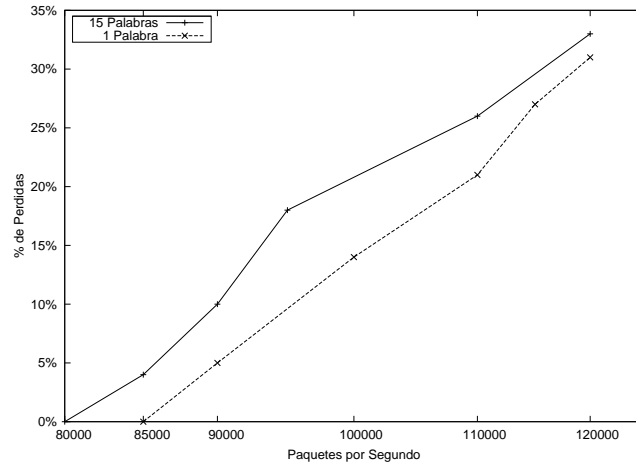


Figura 5.6: Retardo Mínimo UTL

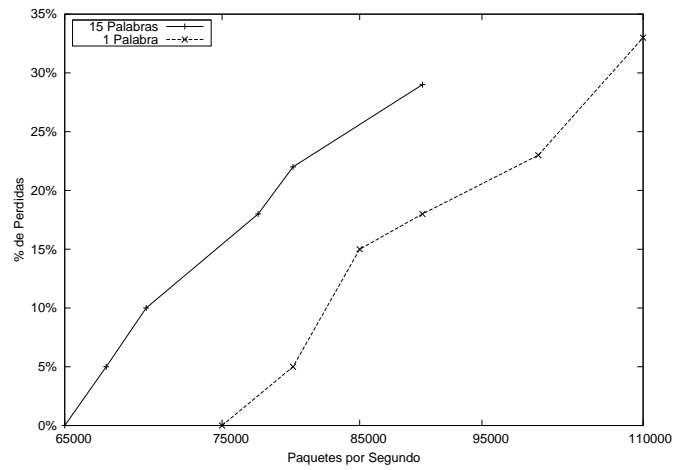


Figura 5.7: Retardo promedio UTL

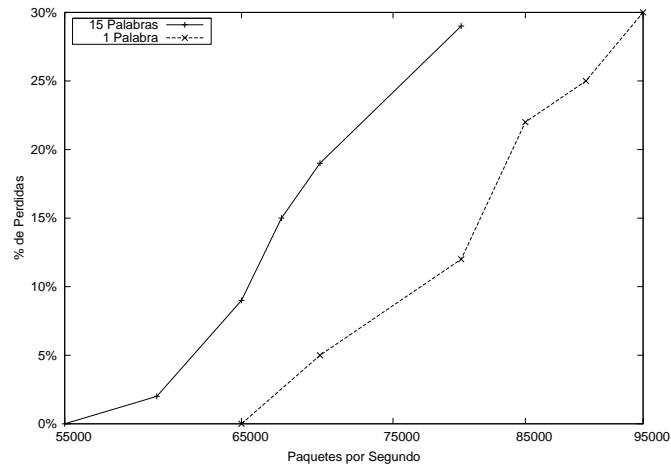


Figura 5.8: Retardo Mínimo UTL

5.1.4. Cache

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 miss, 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

Apendices