

« « « < HEAD ===== « « « < HEAD:pres/informe.out

===== » » » >

56386b87b7902864a65520f969e95c9b935cf371 » » » >

516da482f1c83befd271df24ec283a034a07d823:pres/informe.out

« « « < HEAD ===== « « « <

HEAD:pres/informe.out ===== » » » >

56386b87b7902864a65520f969e95c9b935cf371 » » » >

516da482f1c83befd271df24ec283a034a07d823:pres/informe.out

Implementación HW/SW de Arquitecturas de Clasificación de Paquetes Sobre Lógica Reconfigurable.

Jairo Trad y Luis R. Romano

Laboratorio de Comunicaciones Digitales
Universidad Nacional de Córdoba, Facultad Ciencias Exactas, Físicas y Naturales

January 6, 2012

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Módulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Líneas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos únicamente
- Caso loopback
- Implementación completa
- Comparativa inter-algoritmos

6

Conclusiones

Agenda

1

Motivación

● Requerimientos

- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Requerimientos de procesamiento en Redes

Características de Tráfico

- Las redes de datos crecen en **Complejidad**: nuevas aplicaciones, multimedia
- Las redes de datos crecen en **Velocidad**: $n \times 100Gbps$ (1Tbps@2015)
- **Consolidación** de múltiples servicios sobre redes Ethernet
- Redes *Locales*, *Metropolitanas* y *Extensas* utilizan **Conmutación de Paquetes**
- Adopción de tecnologías para *virtualización* en redes y servidores

Procesamiento de Paquetes

- Los *enlaces* ofrecen alta capacidad. El *procesamiento* de paquetes es **crítico** y debe optimizarse
- El procesamiento *a velocidad de línea*
- Paquete Ethernet mínimo = 64bytes \rightarrow 6nanosegundos/paquete

Requerimientos de procesamiento en Redes

Características de Tráfico

- Las redes de datos crecen en **Complejidad**: nuevas aplicaciones, multimedia
- Las redes de datos crecen en **Velocidad**: $n \times 100\text{Gbps}$ ($1\text{Tbps}@2015$)
- **Consolidación** de múltiples servicios sobre redes Ethernet
- Redes *Locales*, *Metropolitanas* y *Extensas* utilizan **Conmutación de Paquetes**
- Adopción de tecnologías para *virtualización* en redes y servidores

Procesamiento de Paquetes

- Los *enlaces* ofrecen alta capacidad. El *procesamiento* de paquetes es **crítico** y debe optimizarse
- El procesamiento *a velocidad de línea*
- Paquete Ethernet mínimo = $64\text{bytes} \rightarrow 6\text{nanosegundos/paquete}$

Agenda

1

Motivación

- Requerimientos

Soluciones

- Problema marco

- Objetivos

2

Sistema

- Solución Propuesta

- Descripción funcional de cada bloque

- Formato de la cabecera IP

- Algoritmos de Clasificación

- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura

- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II

- Herramientas / Recursos Utilizados

- Verificación

5

Resultados

- Introducción

- Caso algoritmos unicamente

- Caso loopback

- Implementacion completa

- Comparativa inter-algoritmos

6

Conclusiones

Granularidad

-
- The diagram illustrates the flow of packets through a network. At the top, three packets labeled PKT 3, PKT 2, and PKT 1 are shown entering a network. Below them, a large downward arrow indicates the flow. At the bottom, the packets are shown exiting the network as a single flow labeled 'FLOW 1'.

A set of small navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

Tecnologías Actuales

Requerimientos → **flexibilidad, performance**

Circuitos de Propósito Específico (ASICs)

- Cientos de bloques especializados trabajando en paralelo
- Alto desempeño. No programables, alto costo y tiempo de desarrollo.

Procesadores de Red (NPs)

- Múltiples elementos de procesamiento, buena performance para ciertas tareas. IXP(Intel), PowerNP (IBM)
- Difícil portabilidad, interfaces propietarias

Procesadores de Propósito General (GPPs)

- Arquitectura PC + Software especializado: *Click, Zebra/Xorp/Quagga*
- Alta flexibilidad, bajo costo. Limitación por transacciones con RAM y naturaleza secuencial

Tecnologías Actuales

Requerimientos → **flexibilidad, performance**

Circuitos de Propósito Específico (ASICs)

- Cientos de bloques especializados trabajando en paralelo
- Alto desempeño. No programables, alto costo y tiempo de desarrollo.

Procesadores de Red (NPs)

- Múltiples elementos de procesamiento, buena performance para ciertas tareas. IXP(Intel), PowerNP (IBM)
- Difícil portabilidad, interfaces propietarias

Procesadores de Propósito General (GPPs)

- Arquitectura PC + Software especializado: *Click, Zebra/Xorp/Quagga*
- Alta flexibilidad, bajo costo. Limitación por transacciones con RAM y naturaleza secuencial

Tecnologías Actuales

Requerimientos → **flexibilidad, performance**

Circuitos de Propósito Específico (ASICs)

- Cientos de bloques especializados trabajando en paralelo
- Alto desempeño. No programables, alto costo y tiempo de desarrollo.

Procesadores de Red (NPs)

- Múltiples elementos de procesamiento, buena performance para ciertas tareas. IXP(Intel), PowerNP (IBM)
- Difícil portabilidad, interfaces propietarias

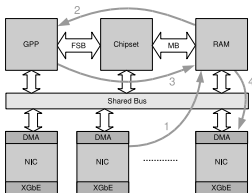
Procesadores de Propósito General (GPPs)

- Arquitectura PC + Software especializado: *Click, Zebra/Xorp/Quagga*
- Alta flexibilidad, bajo costo. Limitación por transacciones con RAM y naturaleza secuencial

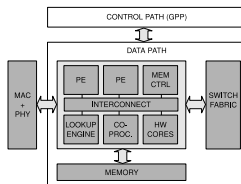
Nuevas tecnologías

Dispositivos Lógicos Programables (FPGAs)

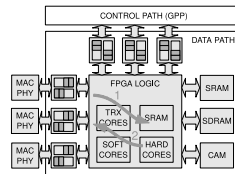
- Permiten *reconfiguración* y *reprogramación*, contando con librerías de *Open Hardware*.
- Su performance no es lejana a la de un ASIC. Fabricantes: Altera, Xilinx, Actel.
- Incorporación creciente de bloques *hardcore* especializados



Implementación con GPPs



Implementación con NPs



Implementación con FPGAs

Agenda

1

Motivación

- Requerimientos
- Soluciones
- **Problema marco**
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Clasificación de Paquetes

Clasificación

- La necesidad de procesar cada vez más paquetes de datos lleva a lo que se conoce como *clasificación*.
- Es el proceso de categorización de paquetes en distintos flujos.
- Efectuada en base a un número de campos de una cabecera.
- En general, para una clasificación basada en N campos, se dice que la misma es N-dimensional (o multidimensional)
- Un caso en particular de la clasificación unidimensional (N=1) utilizando el campo IP destino es lo que se conoce como *IP Lookup*.

IP LookUp

IP Lookup

- Se lleva a cabo en el dispositivo de enrutamiento.
- Un paquete llega por una interfaz de entrada. Éste porta una dirección IP determinada.
- El dispositivo consulta una tabla de forwardeo para determinar la interfaz de salida para el paquete en cuestión
- Dicha tabla contiene un conjunto de prefijos con sus correspondientes interfaces de salida.
- El paquete es correspondido con el prefijo más largo que esté contenido en la dirección de destino y luego es redirigido a la correspondiente interfaz de salida.

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- **Objetivos**

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Objetivos

Generales

- Estudiar las diversas arquitecturas de clasificación de paquetes para poder encontrar las limitaciones en la implementación de las mismas tanto en software como en hardware.
- Ganar conocimiento acerca las diversas posibilidades que ofrecen las FPGA para la implementación de este tipo de algoritmos y las opciones con las que se cuenta a la hora de implementar un sistema embebido en este tipo de dispositivos.

Específicos

- Implementar un sistema embebido que realice la clasificación unidimensional de paquetes mediante una arquitectura mixta, Hardware-Software, en lógica reprogramable y que permita contrastar algunos de los algoritmos de clasificación existentes.
- Implementar como mínimo dos algoritmos de clasificación.
- Mejorar los algoritmos anteriormente mencionados, poniendo el foco en optimizar el código.

Objetivos

Generales

- Estudiar las diversas arquitecturas de clasificación de paquetes para poder encontrar las limitaciones en la implementación de las mismas tanto en software como en hardware.
- Ganar conocimiento acerca de las diversas posibilidades que ofrecen las FPGA para la implementación de este tipo de algoritmos y las opciones con las que se cuenta a la hora de implementar un sistema embebido en este tipo de dispositivos.

Específicos

- Implementar un sistema embebido que realice la clasificación unidimensional de paquetes mediante una arquitectura mixta, Hardware-Software, en lógica reprogramable y que permita contrastar algunos de los algoritmos de clasificación existentes.
- Implementar como mínimo dos algoritmos de clasificación.
- Mejorar los algoritmos anteriormente mencionados, poniendo el foco en optimizar el código.

Agenda

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

Solución Propuesta

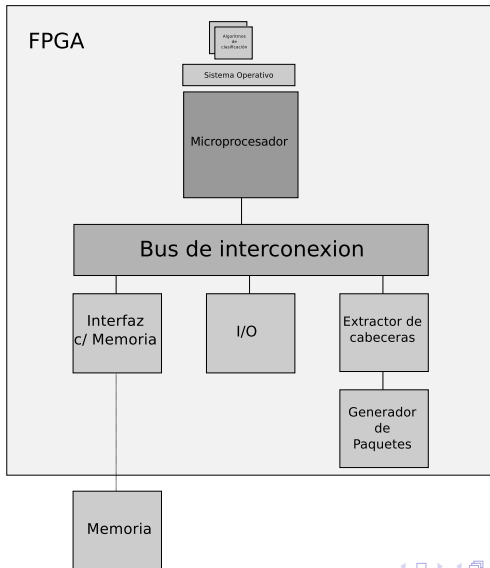
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Módulo extractor de cabeceras

- Arquitectura
- Diagrama en bloques: Líneas E/S

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

Solución



Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- **Descripción funcional de cada bloque**
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Descripción funcional

Microprocesador

- Recibe una cabecera y la procesa mediante la ejecución de un software de clasificación de paquetes.

Bus de interconexión

- Interconecta los componentes del sistema.

Memoria

- Almacena el software de clasificación de paquetes.

Modulo extractor de cabeceras

- Extrae una cabecera a partir de cada paquete recibido.
- La envía al software de clasificación
- Recibe la información generada por dicho software

Descripción funcional

Microprocesador

- Recibe una cabecera y la procesa mediante la ejecución de un software de clasificación de paquetes.

Bus de interconexión

- Interconecta los componentes del sistema.

Memoria

- Almacena el software de clasificación de paquetes.

Modulo extractor de cabeceras

- Extrae una cabecera a partir de cada paquete recibido.
- La envía al software de clasificación
- Recibe la información generada por dicho software

Descripción funcional

Microprocesador

- Recibe una cabecera y la procesa mediante la ejecución de un software de clasificación de paquetes.

Bus de interconexión

- Interconecta los componentes del sistema.

Memoria

- Almacena el software de clasificación de paquetes.

Modulo extractor de cabeceras

- Extrae una cabecera a partir de cada paquete recibido.
- La envía al software de clasificación
- Recibe la información generada por dicho software

Descripción funcional

Microprocesador

- Recibe una cabecera y la procesa mediante la ejecución de un software de clasificación de paquetes.

Bus de interconexión

- Interconecta los componentes del sistema.

Memoria

- Almacena el software de clasificación de paquetes.

Modulo extractor de cabeceras

- Extrae una cabecera a partir de cada paquete recibido.
- La envía al software de clasificación
- Recibe la información generada por dicho software

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- **Formato de la cabecera IP**
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

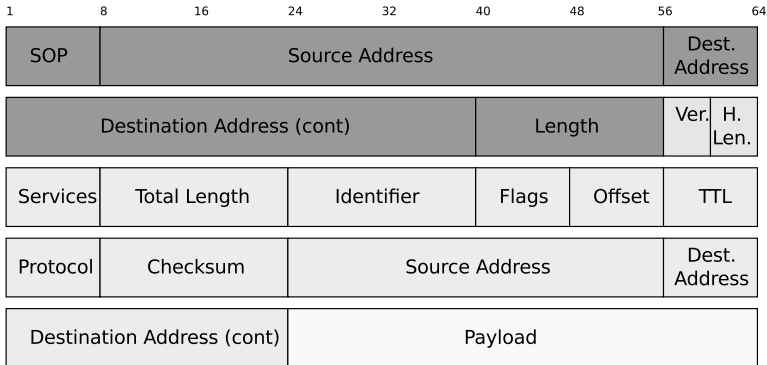
Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos


6

Conclusiones

Formato de la cabecera



 Campos de encabezado Ethernet

 Campos de encabezado IP

Algoritmos de Clasificación

Linear Lookup (LLU)

- Prefijos almacenados en una lista enlazada
- Se toma como entrada una determinada dirección de destino y se va comparando nodo a nodo.
- Prefijos ordenados por longitud
- La primer coincidencia es la mejor

Unibit trie lookup (UTL)

- Prefijos almacenados en un arbol
- Cada nodo representa un bit del prefijo
- Se toma como entrada una dirección de destino y se va recorriendo el arbol en base a los bits

Algoritmos de Clasificación

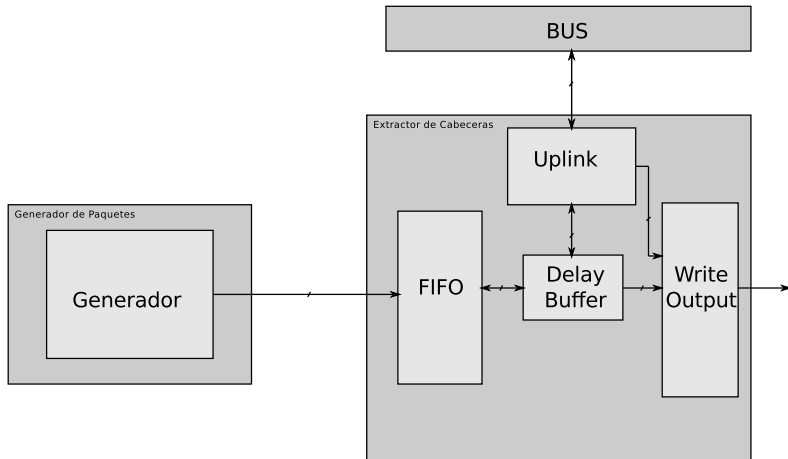
Linear Lookup (LLU)

- Prefijos almacenados en una lista enlazada
- Se toma como entrada una determinada dirección de destino y se va comparando nodo a nodo.
- Prefijos ordenados por longitud
- La primer coincidencia es la mejor

Unibit trie lookup (UTL)

- Prefijos almacenados en un arbol
- Cada nodo representa un bit del prefijo
- Se toma como entrada una dirección de destino y se va recorriendo el arbol en base a los bits

Modulo extractor de cabeceras



Descripción funcional

Generador de paquetes

- Genera paquetes de red Ethernet.
- La distancia entre paquetes y el tamaño de los mismos es configurable.

FIFO

- Recibe los paquetes generados por el módulo anterior y los almacena en una memoria interna.
- De tamaño configurable.

Delay Buffer

- Va tomando los datos desde la FIFO.
- Detecta inicio y finalización de cada paquete.
- Envía las primeras 5 palabras del paquete, suficientes para cubrir la cabecera IP, a Uplink.
- Mantiene almacenado el paquete mientras el software toma una decisión.

Descripción funcional (cont)

Uplink

- Entiende las señales que maneja el Bus y puede interrumpir al Procesador.
- Genera interrupciones al procesador cuando los datos están disponibles.
- Cuando el procesador responde con el resultado de la clasificación, el mismo es almacenado y enviado al módulo Write Output

Write Output

- Toma la salida de Delay Buffer.
- Escribe el resultado que le envía Uplink en la etiqueta que se encuentra anexa en cada una de las palabras del paquete.

Descripción funcional (cont)

Uplink

- Entiende las señales que maneja el Bus y puede interrumpir al Procesador.
- Genera interrupciones al procesador cuando los datos están disponibles.
- Cuando el procesador responde con el resultado de la clasificación, el mismo es almacenado y enviado al módulo Write Output

Write Output

- Toma la salida de Delay Buffer.
- Escribe el resultado que le envía Uplink en la etiqueta que se encuentra anexa en cada una de las palabras del paquete.

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- **Arquitectura**
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Arquitectura

Avalon

- Es una familia de interfaces provista por Altera
- Diferentes tipos, en base a las necesidades (ST, MM, Clock, Interrupt, Conduit, Reset)

En este diseño

- Avalon MM → Los dispositivos están mapeados en memoria
- Avalon Clock Interface → Provee señal de clock al sistema
- Avalon Interrupt Interface → Permite que el módulo extractor de cabeceras interrumpa al procesador
- Avalon Conduit Interface → Permite visualizar resultados fuera de la FPGA

Arquitectura

Avalon

- Es una familia de interfaces provista por Altera
- Diferentes tipos, en base a las necesidades (ST, MM, Clock, Interrupt, Conduit, Reset)

En este diseño

- Avalon MM → Los dispositivos están mapeados en memoria
- Avalon Clock Interface → Provee señal de clock al sistema
- Avalon Interrupt Interface → Permite que el módulo extractor de cabeceras interrumpa al procesador
- Avalon Conduit Interface → Permite visualizar resultados fuera de la FPGA

Señales utilizadas

Avalon MM

- address: Direcciona los datos enviados a o recibidos desde el procesador.
- chipselect: Se usa en combinacion con read o write.
- read: Indica una transferencia de lectura.
- readdata: Bus por el cual se transfieren los datos desde el periférico hacia el procesador.
- write: Indica una transferencia de escritura.
- writedata: Bus por el cual se transfieren los datos desde el procesador hacia el periférico.

Avalon Clock Interface

- clk: Provee clock para la sincronización de señales.

Señales utilizadas

Avalon MM

- address: Direcciona los datos enviados a o recibidos desde el procesador.
- chipselect: Se usa en combinacion con read o write.
- read: Indica una transferencia de lectura.
- readdata: Bus por el cual se transfieren los datos desde el periférico hacia el procesador.
- write: Indica una transferencia de escritura.
- writedata: Bus por el cual se transfieren los datos desde el procesador hacia el periférico.

Avalon Clock Interface

- clk: Provee clock para la sincronización de señales.

Señales utilizadas (cont)

Avalon Interrupt Interface

- irq: Permite al modulo enviar una señal de interrupción al procesador cuando hay una cabecera lista para ser procesada.

Avalon Conduit Interface

- export: Permite exportar señales "fuera" de la FPGA. En este caso, se la utiliza para visualizar ciertos resultados de la clasificación por medio del panel de LEDs de la placa de desarrollo.

Señales utilizadas (cont)

Avalon Interrupt Interface

- irq: Permite al modulo enviar una señal de interrupción al procesador cuando hay una cabecera lista para ser procesada.

Avalon Conduit Interface

- export: Permite exportar señales "fuera" de la FPGA. En este caso, se la utiliza para visualizar ciertos resultados de la clasificación por medio del panel de LEDs de la placa de desarrollo.

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

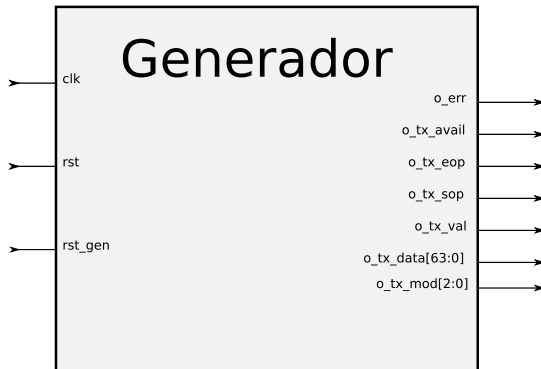
Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

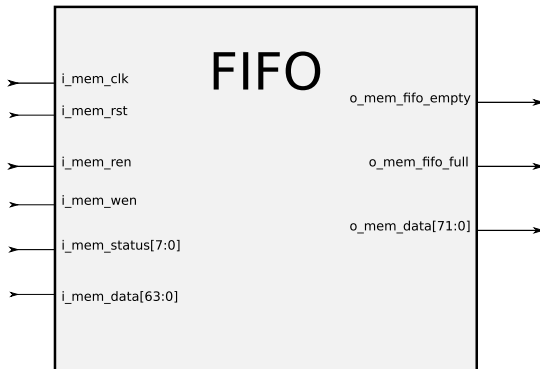
6

Conclusiones

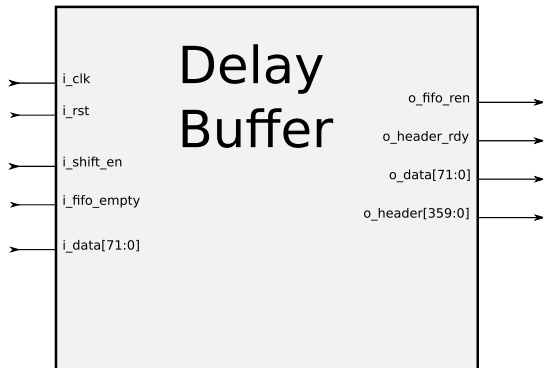
Generador



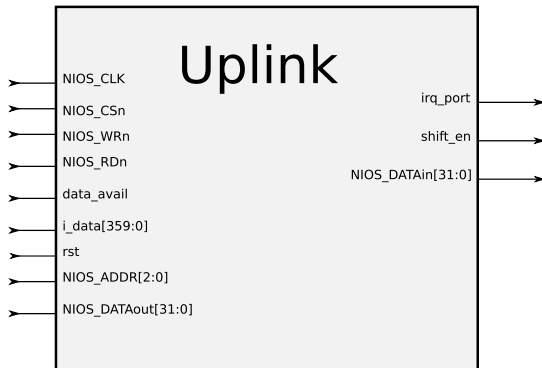
FIFO



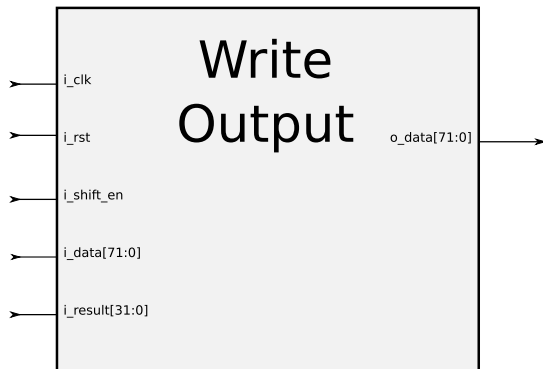
Delay Buffer



Uplink



Write Output



Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- **NIOS II**
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Microprocesador NIOS II

NIOS II

- Procesador RISC de 32 bits de proposito genera diseñado específicamente para la familia de FPGAs de Altera.
- Set de instrucciones, bus de datos y espacio de direcciones de 32 bits.
- Soporte de hasta 32 interrupciones.
- Entorno de desarrollo de software basado en GNU C/C++ integrado a Eclipse.
- Integración con SignalTap® II, el analizador de lógica embebida de Altera permitiendo el análisis de todas las señales presentes en la FPGA.
- Set de instrucciones compatible entre todas las versiones del procesador Nios II.
- Multiplicación y división en una sola instrucción de 32 x 32 produciendo un resultado de 32-bits.

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- **Herramientas / Recursos Utilizados**
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Herramientas / Recursos Utilizados

Quartus

- IDE de Altera
- Incluye editor de textos y herramientas para síntesis
- Lenguaje HDL utilizado: Verilog HDL

Eclipse IDE for NIOS

- Version de la IDE Eclipse adaptada para trabajar con el microprocesador NIOS II
- Lenguajes utilizados: C,C++

Herramientas / Recursos Utilizados

Quartus

- IDE de Altera
- Incluye editor de textos y herramientas para síntesis
- Lenguaje HDL utilizado: Verilog HDL

Eclipse IDE for NIOS

- Version de la IDE Eclipse adaptada para trabajar con el microprocesador NIOS II
- Lenguajes utilizados: C,C++

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- **Verificación**

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Verificación

Etapas

- 1º Paso: Módulo más simple (simpleRW)
- 2º Paso: Implementación del modulo extractor. Debugging de señales.
- 3º Paso: Integración extractor-software. LLU y UTL.

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- **Introducción**
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Introducción

Presentación de los resultados



Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- **Caso algoritmos unicamente**
- Caso loopback
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Caso algoritmos unicamente

.. Grafico correspondiente ...

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- **Caso loopback**
- Implementacion completa
- Comparativa inter-algoritmos

6

Conclusiones

Caso loopback

.. Grafico correspondiente ...

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- **Implementacion completa**
- Comparativa inter-algoritmos

6

Conclusiones

Implementación completa: LLU

.. Graficos retardo mínimo ...

Implementación completa: LLU

.. Graficos retardo promedio ...

Implementación completa: LLU

.. Graficos retardo máximo ...

Implementación completa: UTL

.. Graficos retardo mínimo ...

Implementación completa: UTL

.. Graficos retardo promedio ...

Implementación completa: UTL

.. Graficos retardo máximo ...

Agenda

1

Motivación

- Requerimientos
- Soluciones
- Problema marco
- Objetivos

2

Sistema

- Solución Propuesta
- Descripción funcional de cada bloque
- Formato de la cabecera IP
- Algoritmos de Clasificación
- Modulo extractor de cabeceras

3

Arquitectura

- Arquitectura
- Diagrama en bloques: Lineas E/S

4

Implementación

- NIOS II
- Herramientas / Recursos Utilizados
- Verificación

5

Resultados

- Introducción
- Caso algoritmos unicamente
- Caso loopback
- Implementacion completa
- **Comparativa inter-algoritmos**

6

Conclusiones

Comparativa inter-algoritmos

.. Grafico comparativo ...

Conclusiones

Conclusiones

