FPGAs Hardware Reconfigurable en Linux

Tropea S.E., Borgna J.D.

Introducción

Hardware

Sistemas de cálculo automático

Reseña histórica de la evolución de circuitos integrados reusables

- 1946: Primera computadora digital: ENIAC
- Programa fijo (cableado) y Harware fijo

- Cálculos balísticos.
- 1440 veces más rápido que los humanos.
- Costo de casi U$S 500.000
- 0,05 MIPS y 200 bytes de memoria
- 17.468 válvulas, 7.200 diodos, 1.500 relés, 70.000 resistores y 10.000 capacitores.
- 27 toneladas

Reseña histórica de la evolución de circuitos integrados reusables

'20s

1947

1953

- 1947: Primer transistor. Laboratorios Bell (Shockley)
- 1951: Primer aplicación comercial del transistor.
- Fotos: válvulas de principios de los '20, primer transistor y 2N43 de 1953.

Reseña histórica de la evolución de circuitos integrados reusables

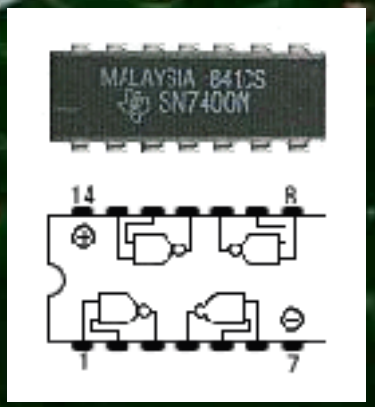
- 1951: IBM introduce la primer computadora que es un éxito comercial (701)
- 1956: Primeras computadoras basadas en transistores.

- Primeros lenguajes de programación. (COBOL y FORTRAN)
- Programas configurables y Hardware fijo.

Reseña histórica de la evolución de circuitos integrados reusables

Primer C.I.

RTL F/F

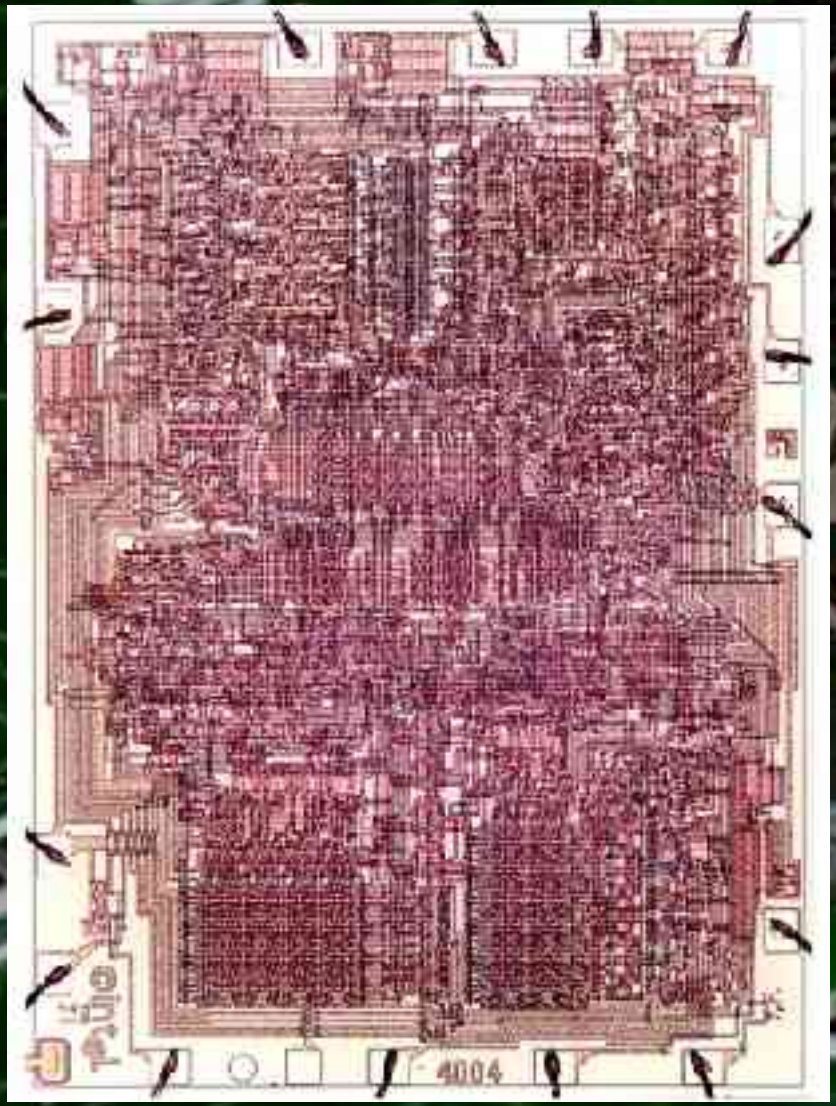


- 1958: Primer circuito integrado. Texas Instruments (Jack Kilby)
- 1961: Primer circuito integrado comercial. Fairchild Semiconductors (Robert Noyce) (RTL)
- RTL -> DTL -> TTL

Reseña histórica de la evolución de circuitos integrados reusables

- 1964: Primeras computadoras con circuitos integrados.
- 1965: Primeras PDP-8.
- 1968: PDP-8/I primera PDP-8 con C.I. (TTL)
- 1970: PDP-8/E U$S 6.500 0,8 MIPS 32 kWords 12 bits.

Reseña histórica de la evolución de circuitos integrados reusables

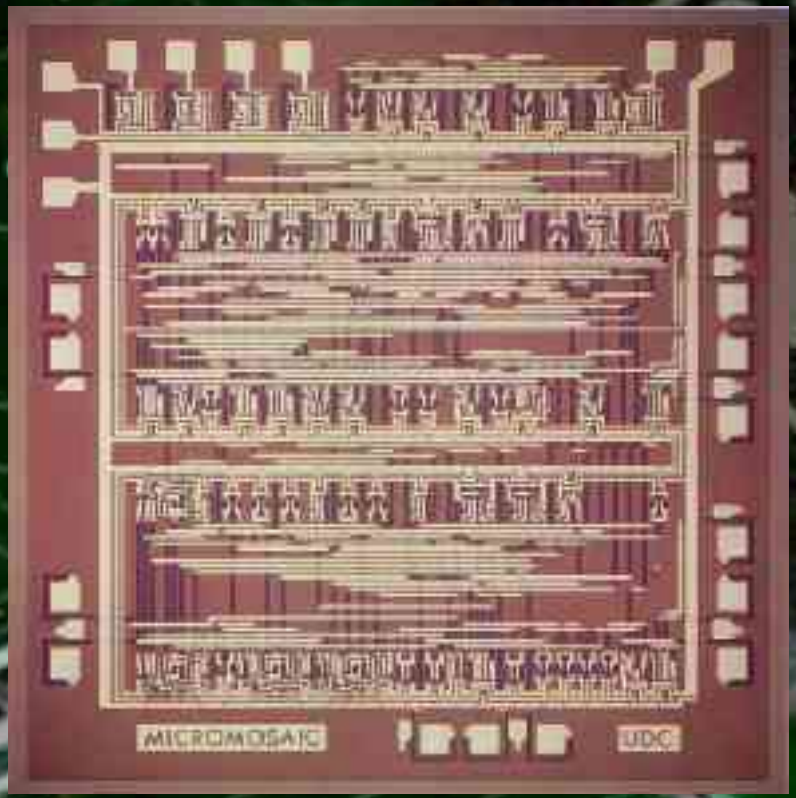


- 1968: Robert (Bob) Noyce funda Intel.
- 1971: Primer microprocesador. 4004 de Intel (Federico Faggin). Para ser usado por Busicom en calculadoras.
- 4004: 4-bits, 108 kHz, 2300 transistores, 0,06 MIPS

Reseña histórica de la evolución de circuitos integrados reusables

- Los microcontroladores dan nacimiento a computadoras rápidas y baratas.
- Problema: cálculo serializado => baja performance.
- Tampoco son buenos en consumo de energía.
- Solución: Chips custom ... demasiado caro.
- Mejor solución: Chips semi-cutom: ASICs
- ASIC: Application Specific Integrated Circuit.

Reseña histórica de la evolución de circuitos integrados reusables



- 1967: Primera idea (sin mucho éxito) Micromosaic de Fairchild. Sólo unos cientos de transistores.
- Sólo estan los transistores, lo que falta son las interconexiones.
- No era el momento en que el mercado lo requería.
- Muy pocos recursos (aprox. 300 gates).

Anécdota

- 1947: Shockley trabajando para Bell inventa el transistor.
- 1951: Primera versión práctica.
- 1956: Shockley funda Shockley Semiconductor para hacer las cosas a su modo.
- 1957: 8 de sus empleados cansados de su administración se van para crear Fairchild Semiconductors (los 8 traidores) R.N. y G.M.
- 1961: Robert (Bob) Noyce crea el primer C.I. práctico.
- 1967: Fairchild crea el ASIC (Micromosaic).
- 1968: Bob y Gordon Moore (ley) se van y junto con Andy Groove crean Intel [Shockley cierra].

Anécdota

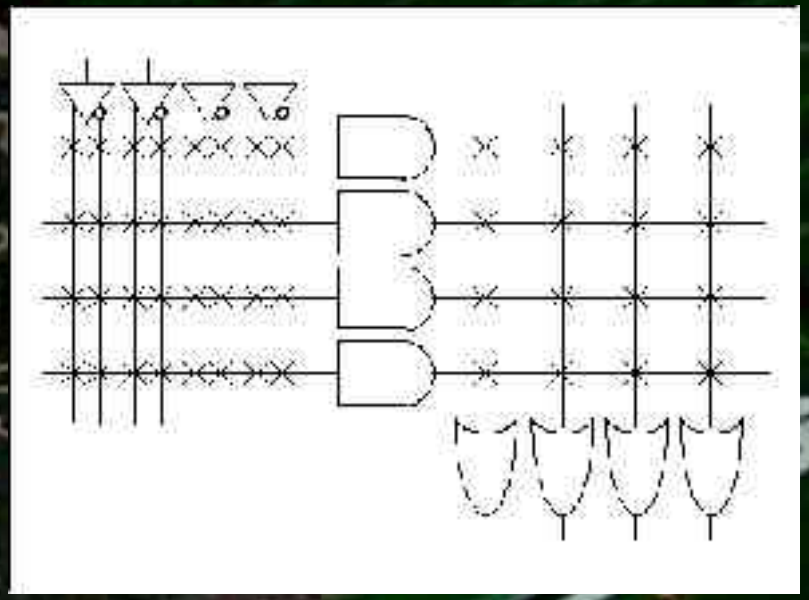
- 1970: Fairchild e Intel compiten por las primeras SRAM (256 v.s. 1024 bits).
- 1971: Federico Faggin (ex-Fairchild) en Intel crea el primer µ: 4004.
- 1974: Enojado FF se va y funda Zilog.
- 1976: Masatoshi Shima de Zilog crea el Z80 y le quita mercado a Intel. Se usa para las *home computer*.
- 1981: IBM introduce la *IBM PC* (code name Acorn, Project Chess). Usa una CPU de Intel y un OS licenciado de Microsoft.
- 1982: Aparecen los primeros *clones*.
- El resto ya lo sabemos todos ...

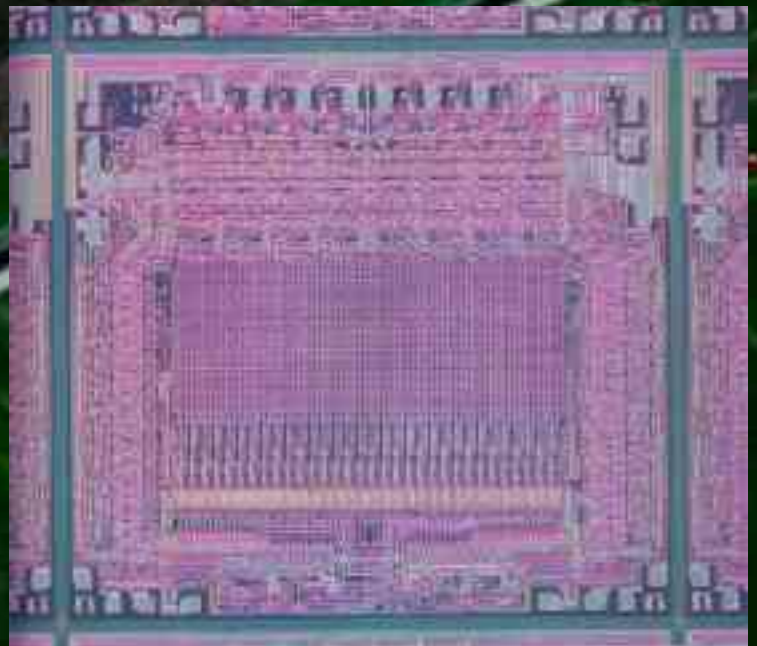
Reseña histórica de la evolución de circuitos integrados reusables

- 1980: Ferranti introduce las “Uncommited Logic Array”. Contienen compuertas sin interconectar. Se usan en las primeras *Home Computers* como la ZX81 de Sinclair.
- Más tarde aparecen las *Standard Cell* donde se usan bloques prediseñados.
- Problema 1: Sólo son viables cuando se hacen decenas de miles.
- Problema 2: Un error sale muy caro.
- Resultado: Sólo lo usan industrias importantes.

Reseña histórica de la evolución de circuitos integrados reusables

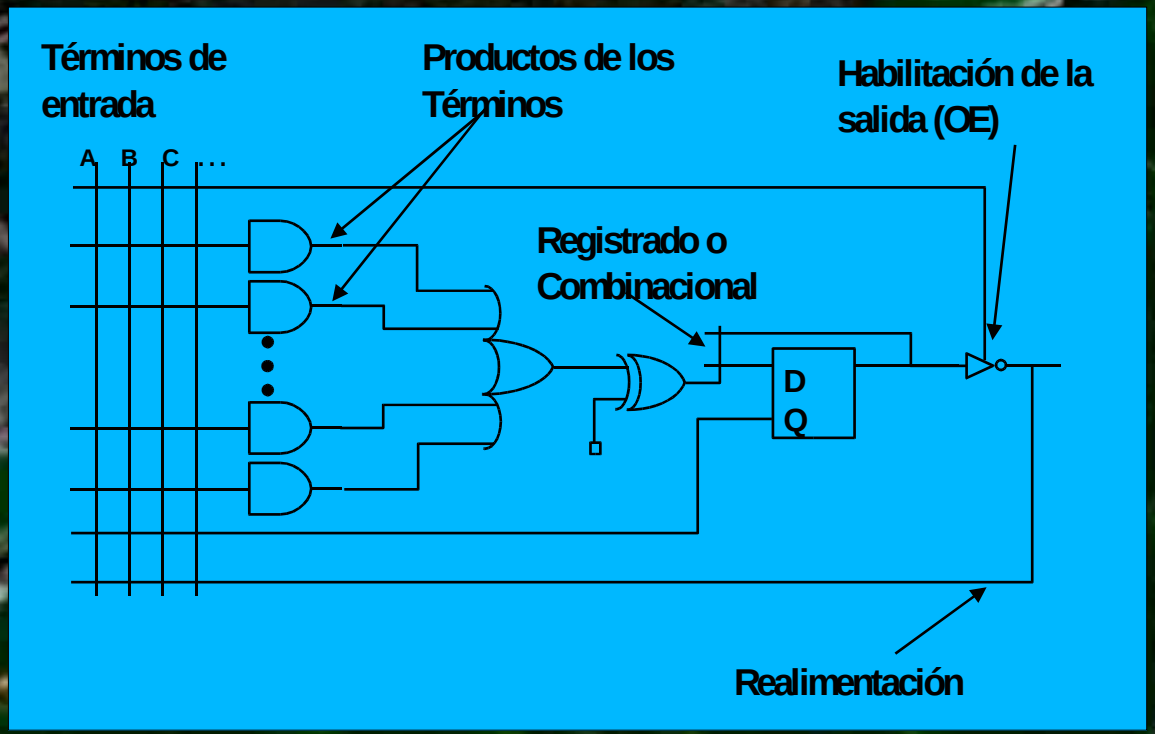
- Solución: Lógica programable.
- 1977: PAL 16L8 Programmable Array Logic. Monolithic Memories Inc. (J. Birkner y H.T. Chua)
- Lógica con interconexiones que se pueden remover por el usuario.
- Problema: muy poca funcionalidad.
- Problema: **no** reconfigurable.
- Problema: sólo combinacional.
- Nota: En 1975 Signetics e Intersil introdujeron las FPLA, pero fue la PAL 16L8 la que tuvo éxito comercial.





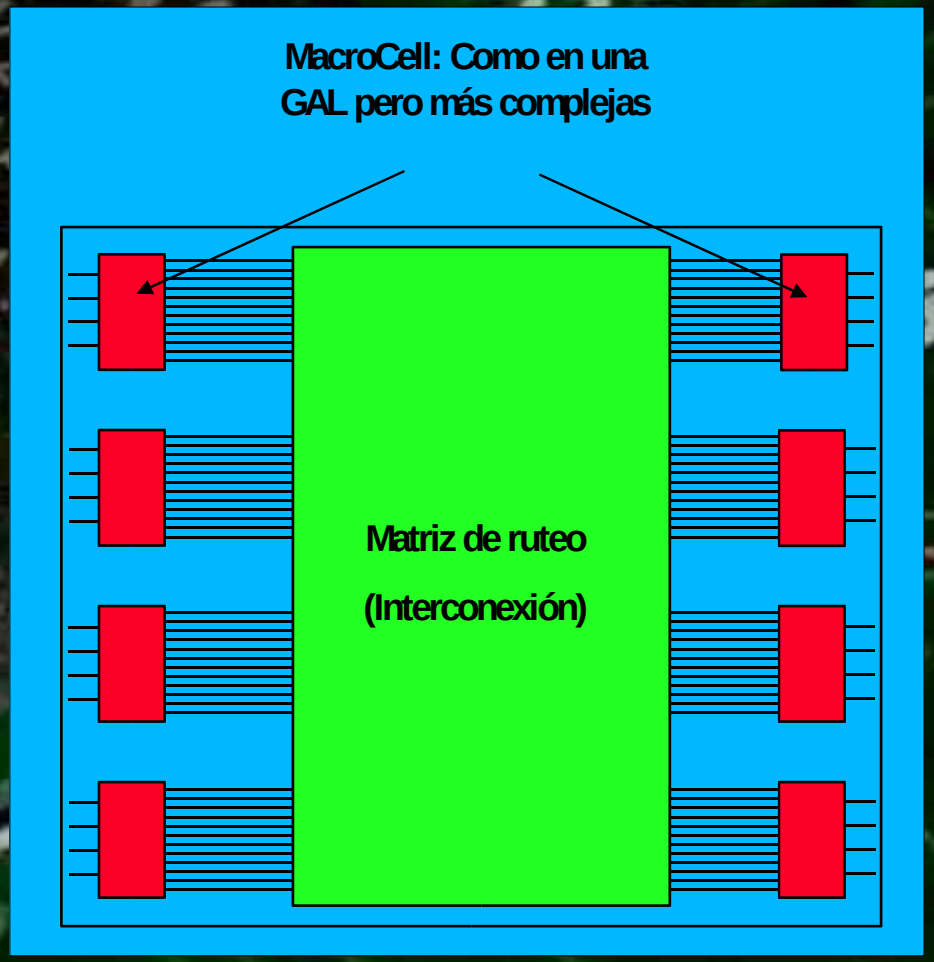
Reseña histórica de la evolución de circuitos integrados reusables

- Aparecen las “Advanced PAL” con registros (Flip Flops) en las salidas.
- Lattice crea la GAL (Generic Array Logic) que es reconfigurable (EEPROM)
- Introducen el concepto de *MacroCell*
- Siguen siendo de funcionalidad limitada y sólo reemplazan un poco de lógica, pero no a un microcontrolador



Reseña histórica de la evolución de circuitos integrados reusables

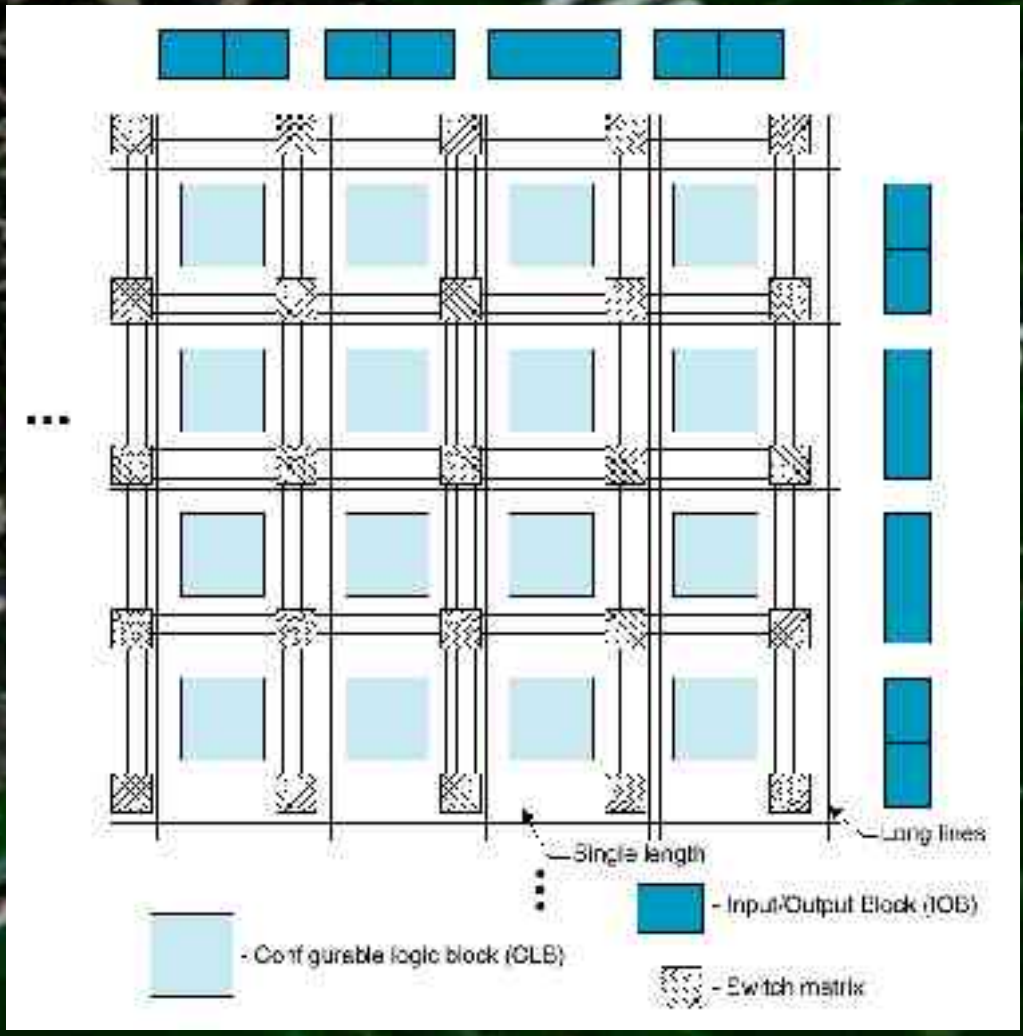
- Agregando una matriz de interconexión y un mayor número de macroceldas aparecen las primeras CPLD.
- Problema: la interconexión entre macroceldas es limitada.
- Problema: poca funcionalidad, las macroceldas se encuentran en la “periferia”.



Reseña histórica de la evolución de circuitos integrados reusables

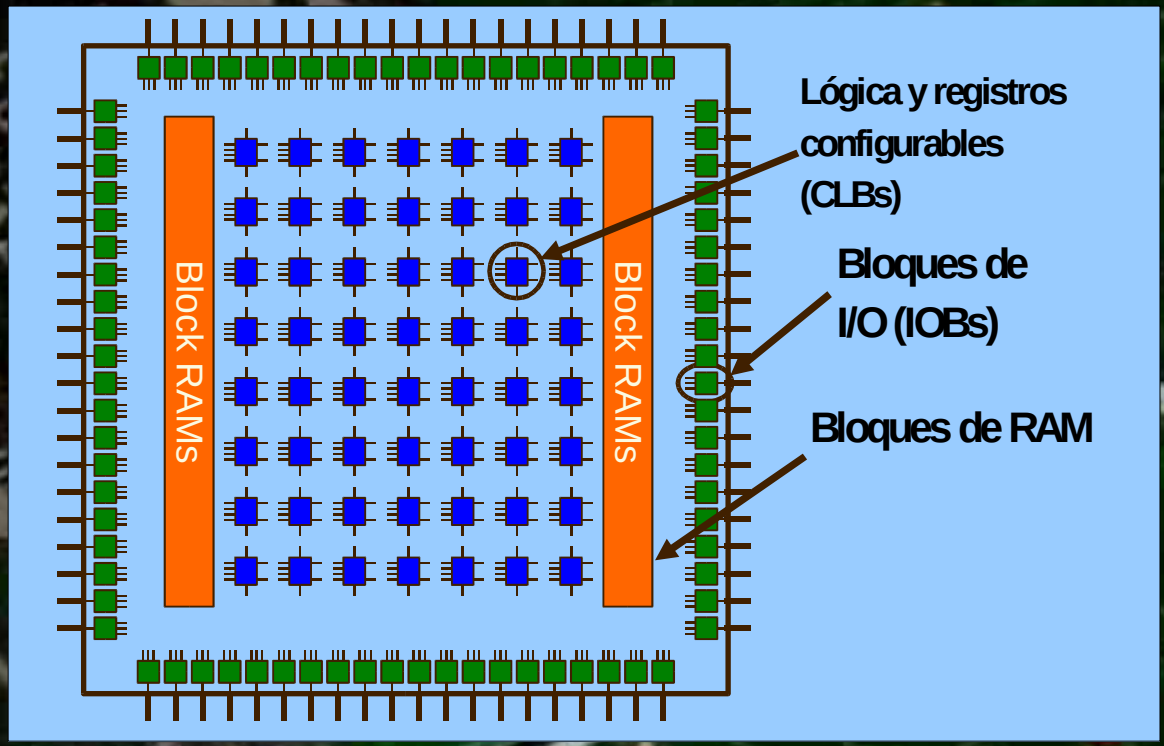
- Solución: Field Programmable Gate Array FPGA.
- Inventada por Ross Freeman en 1984 (co-fundador de Xilinx)
- Lógica en todo el chip, no sólo las I/O.
- Interconexión mucho más rica.
- Introducidas a mediados de los '80 por Altera, Xilinx y otros.
- Impulsadas por la aparición de lenguajes capaces de manejar lógica más compleja.

...



¿Qué son las FPGAs?

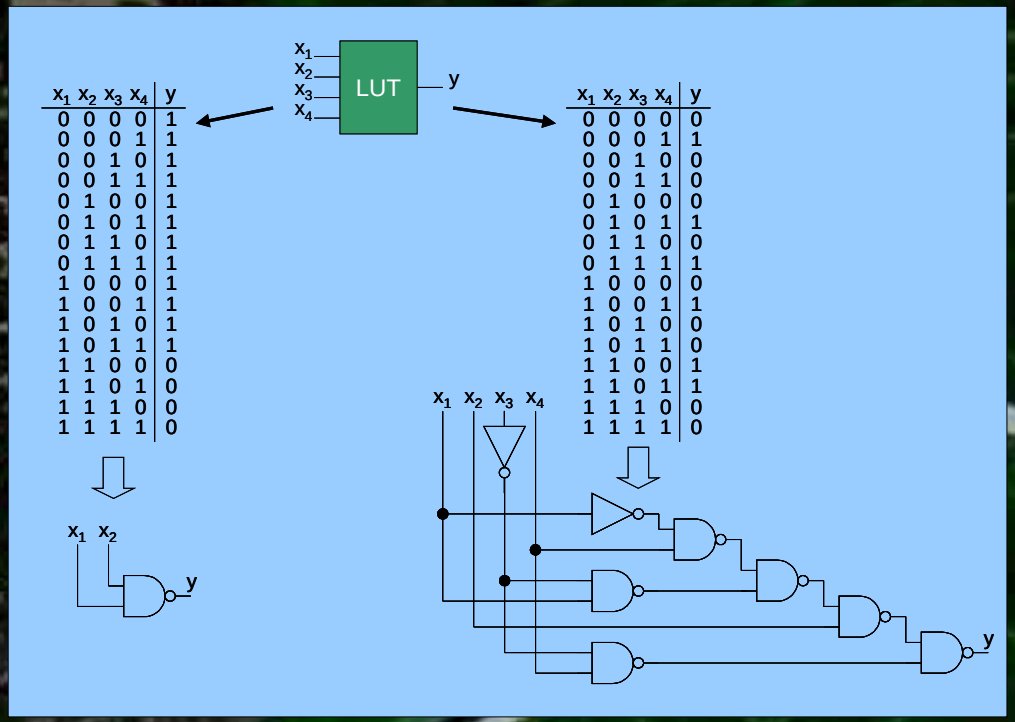
- Las FPGAs son circuitos integrados (re)configurables.
- Poseen lógica combinacional (re)configurable.
- Poseen registros (flip/flops) (re) configurables.
- Poseen un sistema poderoso de interconexión (re)configurable.



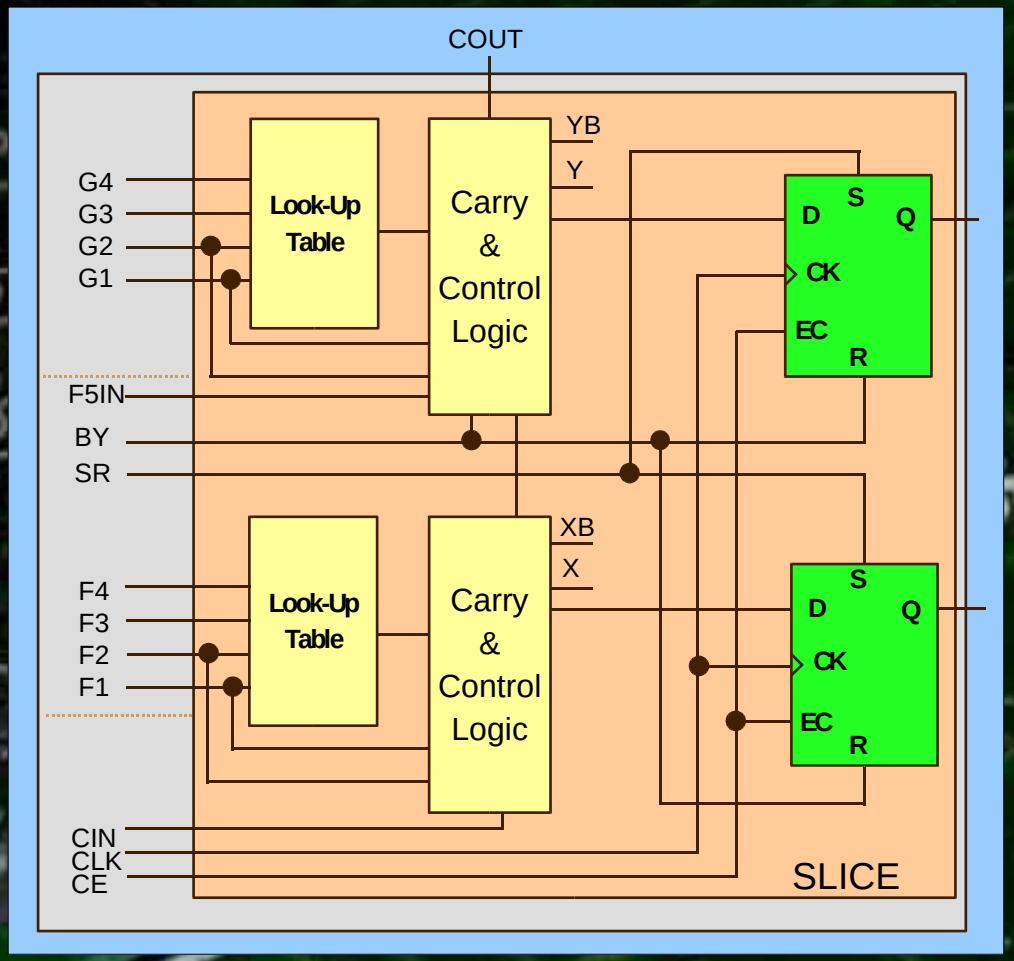
- Poseen entradas/salidas muy flexibles y (re)configurables.
- Agregan más y más funcionalidad: Bloques de memoria, multiplicadores, bloques “multiply/accumulate”, PLLs, CPUs, etc.

¿Qué son las FPGAs?

- Lógica combinacional (re) programable.
- Basada en LUTs (Look-Up Tables) de 4 entradas.
- Permiten implementar cualquier función lógica de 4 variables de entrada y una salida.
- Son simplemente memorias de 16 bits, cada combinación de entrada selecciona una posición de memoria que es presentada a la salida.



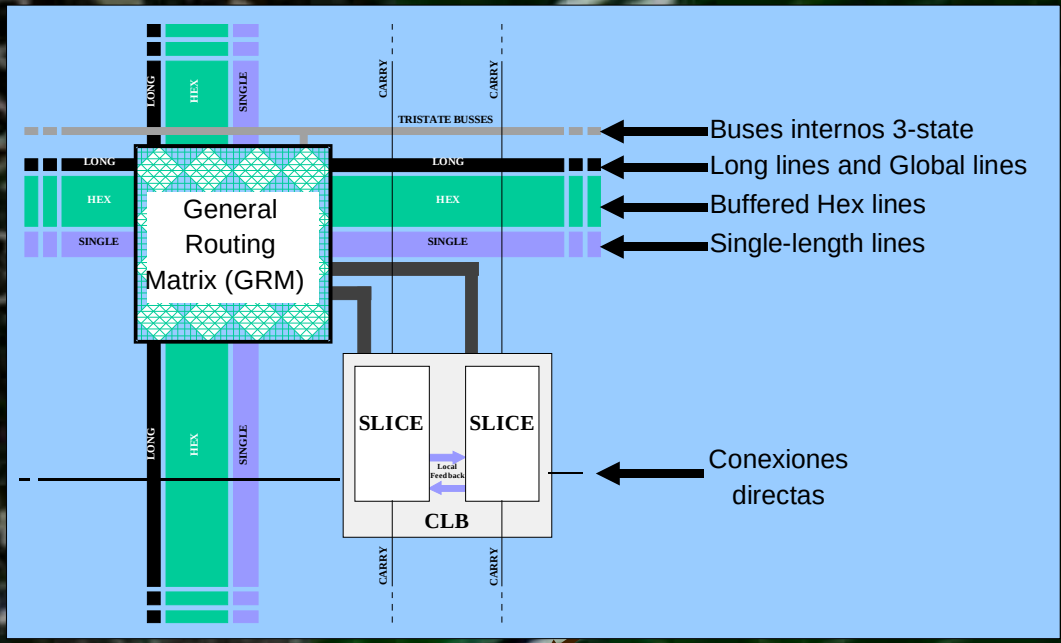
¿Qué son las FPGAs?



- Registros (re)programable.
- Flanco de subida/bajada, reset/set a/sincrónico, por flanco o nivel (latch).
- Permiten retener los estados actuales.
- Indispensables para el diseño síncrono.
- En el esquema: dos LUTs con sus dos Flip/Flops forman un bloque lógico básico en FPGAs como la Spartan o Virtex de Xilinx. Lógica dedicada agrega flexibilidad extra.

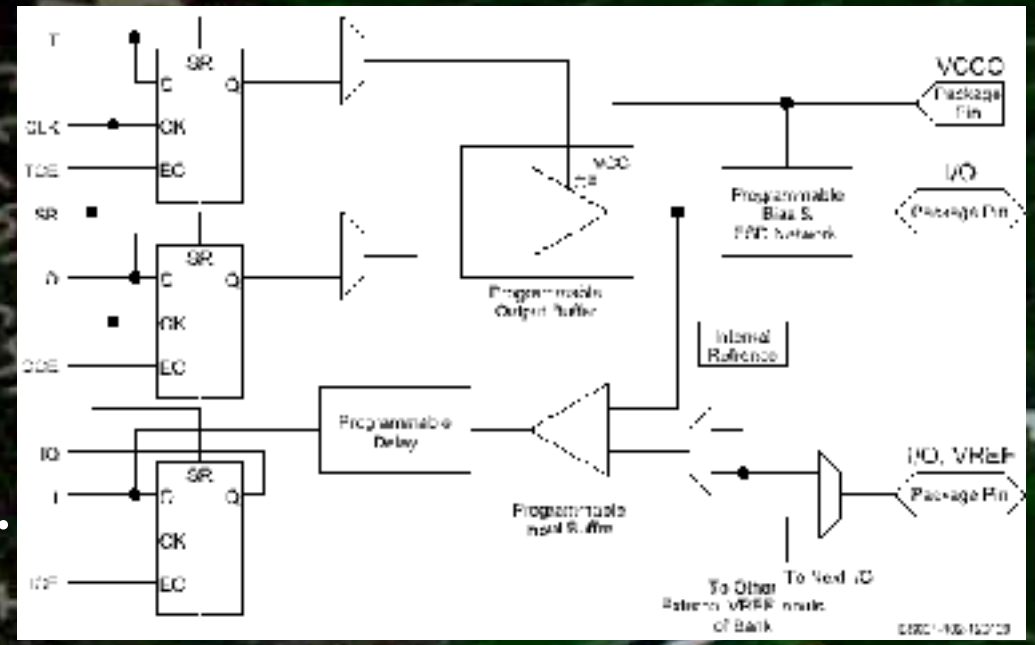
¿Qué son las FPGAs?

- Poderoso sistema de interconexión.
- Permite interconectar cualquier CLB con otro o con un IOB.
- Sistema jerárquico con conexiones locales a globales.
- En algunos casos incluso 3-state. Hoy menos común por cuestiones de performance.

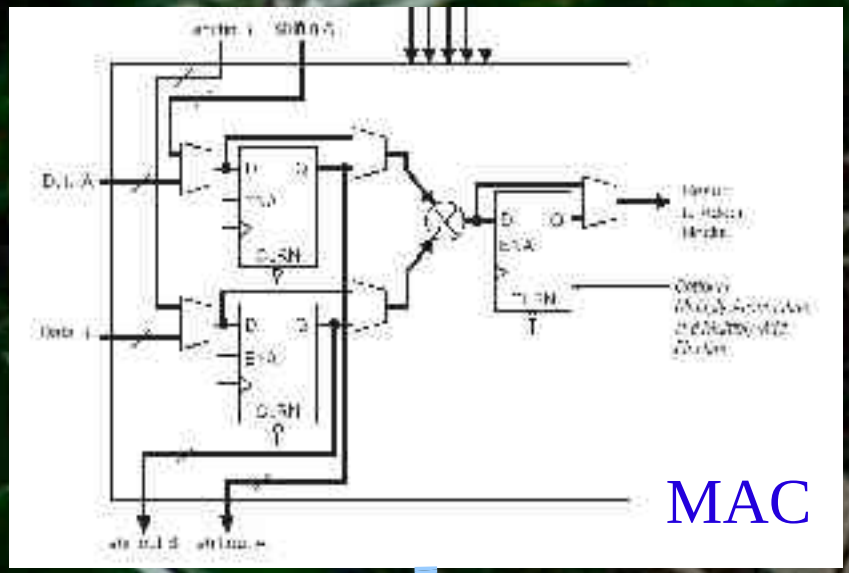


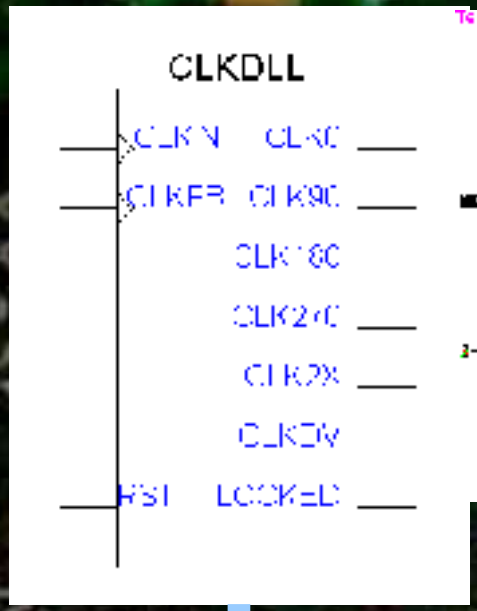
¿Qué son las FPGAs?

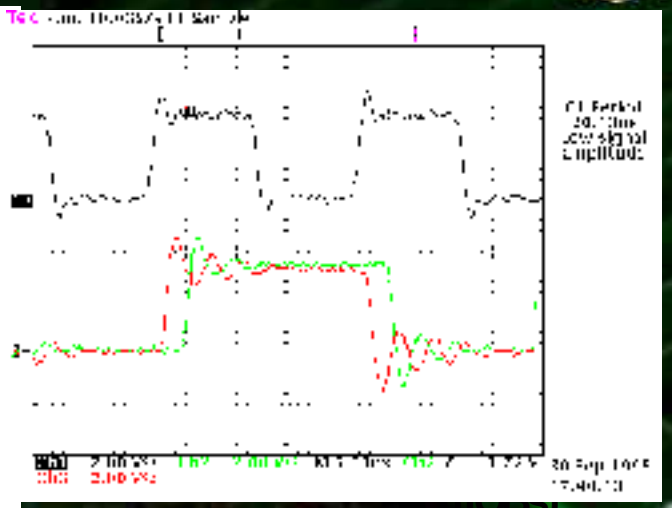
- Entradas/Salidas flexibles.
- Pueden configurarse como entradas o como salidas.
- Three state, pull-up, pull-down, bus keeper, input delay.
- Diferentes estándares de niveles de tensión (ej. 16 en Spartan II de Xilinx)
- Suelen incluir F/F para registrar la entrada/salida.
- Vref separada por bancos.
- Soporte para DDRs en Spartan 3.

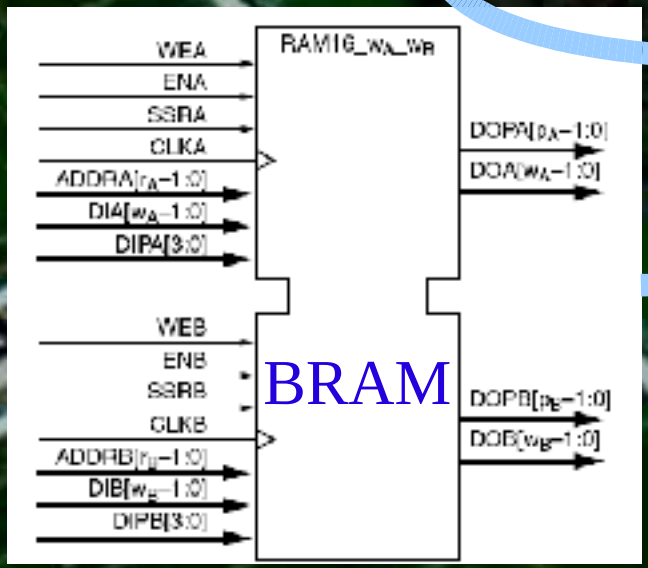


¿Qué son las FPGAs?

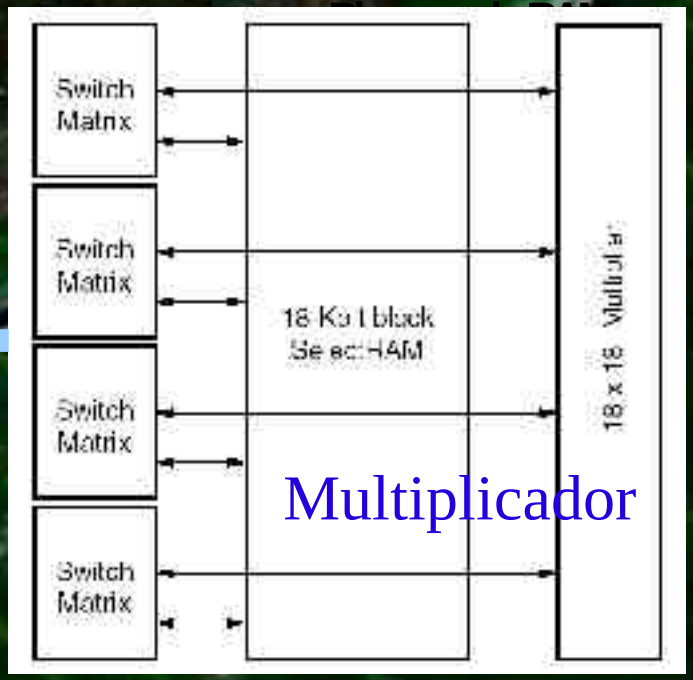




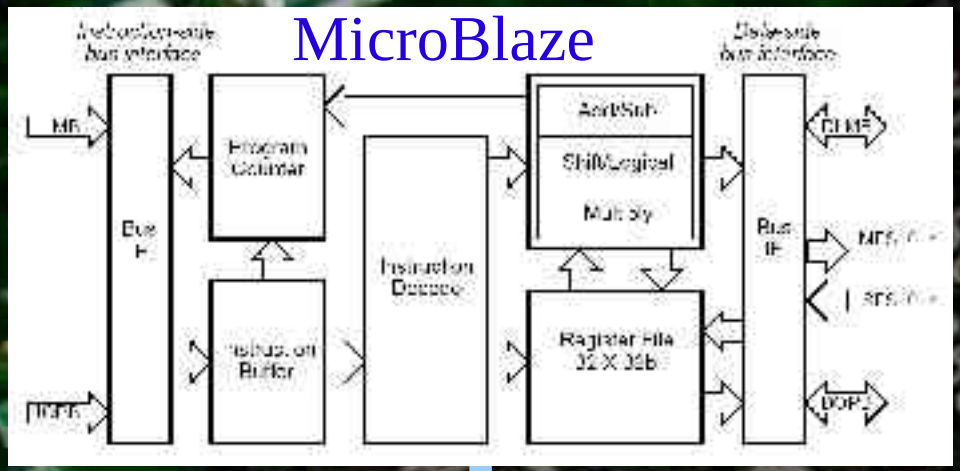


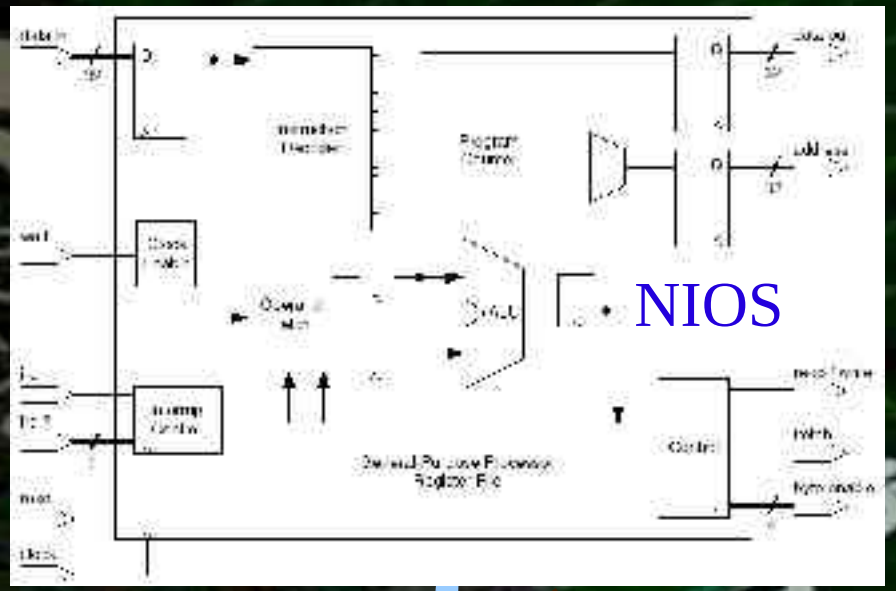


Sólo ilustrativo



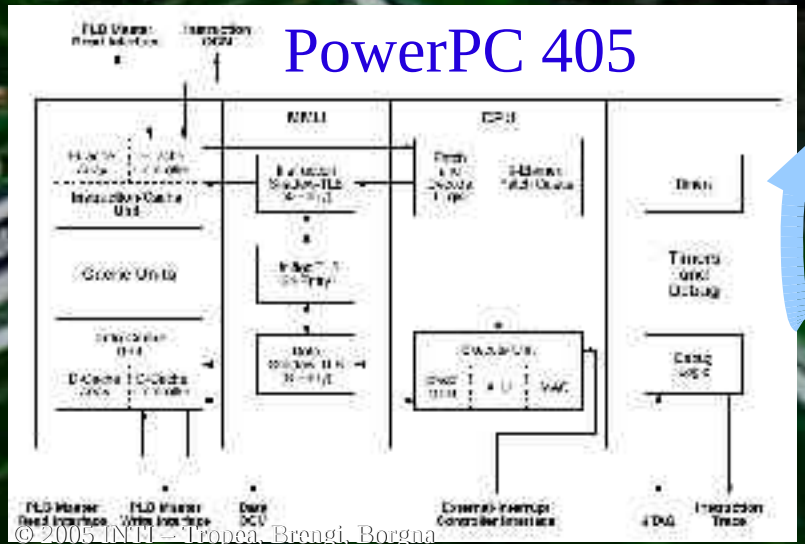
¿Qué son las FPGAs?



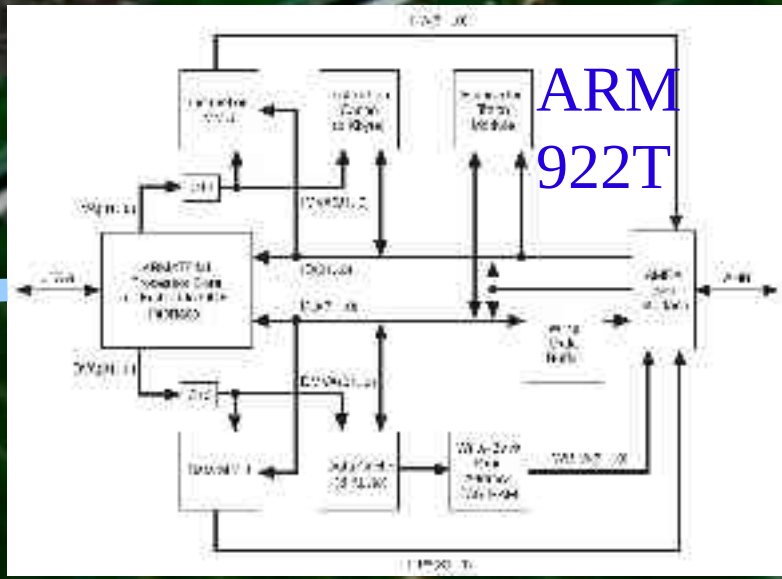


XILINX®

ALTERA®

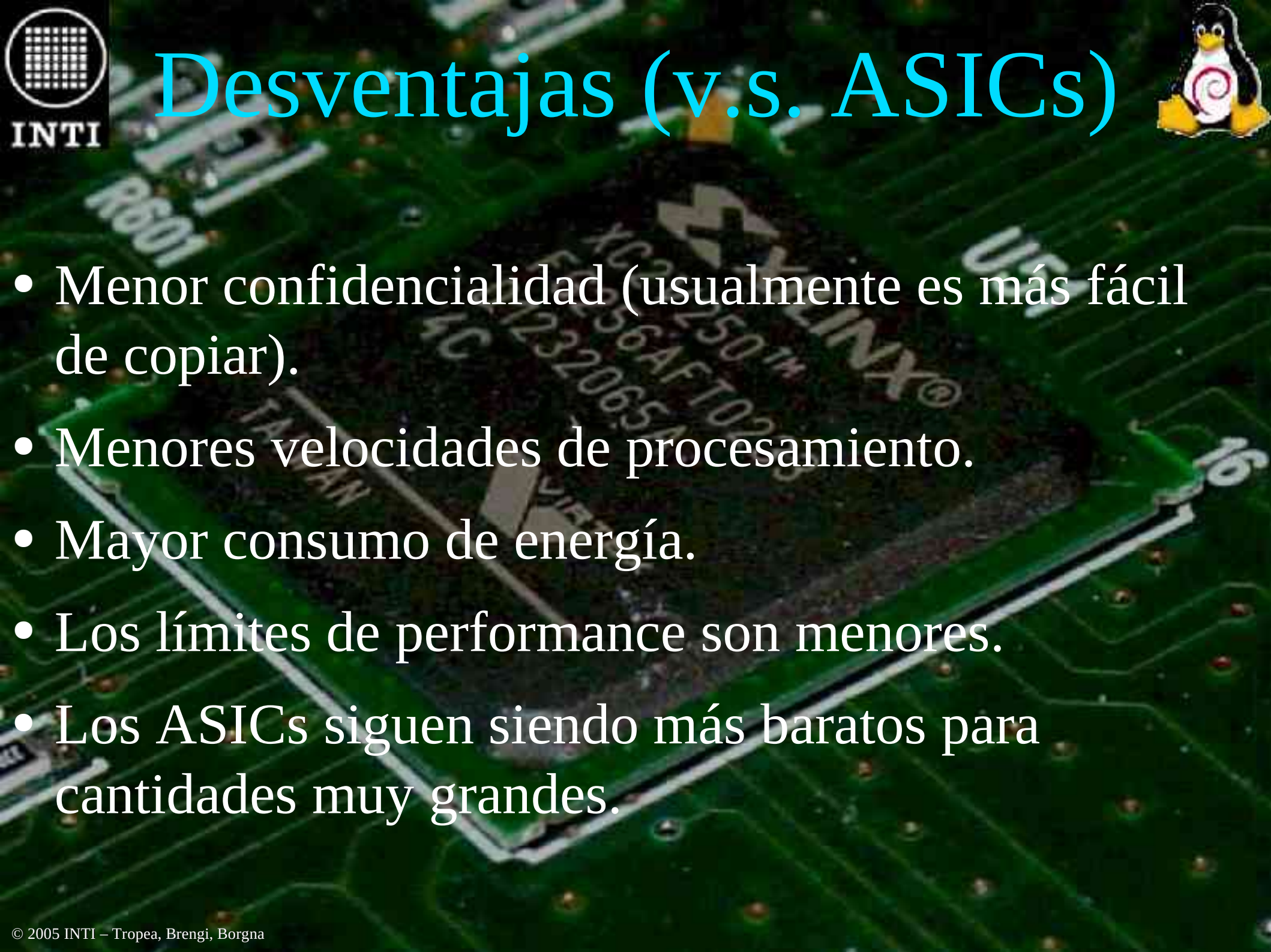


Sólo ilustrativo



Ventajas (v.s. ASICs)

- Apto para desarrollos en pequeñas cantidades. Actualmente comienzan a serlo también para grandes cantidades.
- Reconfigurables, se pueden corregir errores en productos ya instalados.
- Herramientas de desarrollo simples y accesibles.
- Tiempos de desarrollo más cortos.
- Costos de desarrollo mucho menores.



Desventajas (v.s. ASICs)

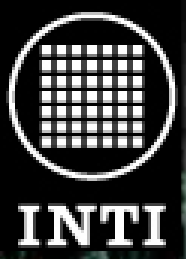
- Menor confidencialidad (usualmente es más fácil de copiar).
- Menores velocidades de procesamiento.
- Mayor consumo de energía.
- Los límites de performance son menores.
- Los ASICs siguen siendo más baratos para cantidades muy grandes.

Ventajas (v.s. µs)

- Se puede obtener performance muy superior ya que todo sucede en paralelo. Se pueden hacer N productos en forma simultanea en lugar de serie.
- Mayor grado de integración ya que se pueden integrar los periféricos (RS-232, I2C, Ethernet, USB, etc.).
- Se puede sintetizar un µ con lo que es un superset.
- Menor consumo.
- Número de pines muy superior (+1000)

Desventajas (v.s. µs)

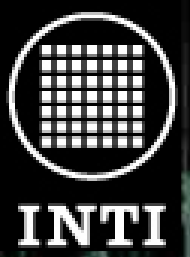
- Mayor costo cuando las tareas a realizar no son de alta performance y pueden resolverse con µs. Se espera que se equipare en los próximos años.
- Mayor complejidad de diseño. Se diseña hardware (y software) y no software.
- Encapsulados más difíciles de usar. Aunque actualmente los µs de alta performance también los usan.



Introducción

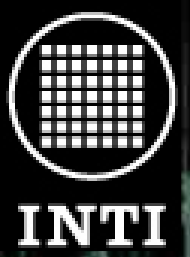
Software

Lenguajes de descripción de hardware



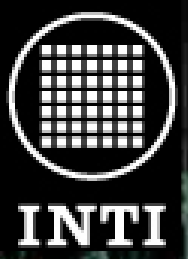
Lenguajes de descripción de hardware

- Se parecen a los lenguajes de programación pero describen el funcionamiento de un dispositivo.
- Los más avanzados se pueden usar para describir como se comporta cualquier objeto/mecanismo.
- Los eventos suceden todos al mismo tiempo: inherentemente concurrentes.
- Se los suele llamar HDLs (Hardware Desciption Languages)



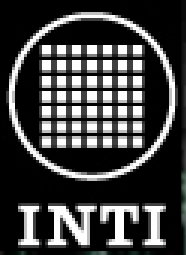
Lenguajes de descripción de hardware

- PALASM: Introducido a principios de los '80 para programar las PAL.
- Muy simple y limitado a la funcionalidad de una PAL.
- Describe los modos de trabajo y las ecuaciones lógicas de cada salida.
- ABEL: Creado en 1983 agrega funcionalidad para máquinas de estados.
- Poco abstractos y muy atados a la arquitectura de los dispositivos. Describían las cosas a muy bajo nivel.



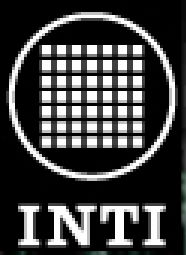
VHDL

- **V**HSIC **H**ardware **D**escription **L**anguage
- **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit
- Desarrollado para el US Department of Defense para describir el funcionamiento de los ASICs que compraba.
- Más tarde aparecieron simuladores y luego sintetizadores.
- IEEE 1076-1987, 1076-1993, 1164
- Basado en ADA (también creado por el DoD)
- Describe comportamiento e interconexión.
- Muy usado en la costa Este de USA y en Europa.



Verilog

- Creado alrededor de 1984 por Gateway Design Automation, adquirida en 1990 por Cadence.
- Debido a la presión del VHDL Cadence lo abre.
- IEEE 1364-1995 y 1364-2001.
- Más compacto que el VHDL, menos prolijo.
- Hace recordar al lenguaje C.
- Muy usado en la costa Oeste de USA.



SystemC

- Creado con la idea de describir sistemas completos.
- Son librerías C++.
- Actúa como descripción y kernel simulador.
- Actualmente las empresas EDA tratan de seguir el camino del VHDL para que sea posible sintetizarlo. Resultados variables.

Ciclo básico de Desarrollo

Búsqueda
ctags

Plantillas
Pmacros y tpl2file

Edición
SETEdit

Bibliotecas y Cores

Automatizador
GNU Make

Otros
hex2vhdl
vhdlspp
natebege

Compilador
GHDL

Estilo (Lint)
bakalint

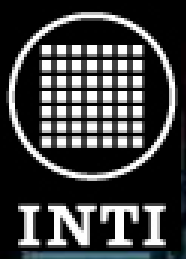
Forma de Onda
GTKWave y dinotrace

Testbench
GHDL

Recolección de errores

Error

Ciclo básico de Desarrollo



Testbench
GHDL

Forma de Onda
GTKWave y dinotrace

Error

Recolección de errores

Post-proceso
GCC y Perl

Síntesis
xst/map/par

No cumple

JTAG
JBit, naxjp, xilix_jtag

FPGA
Spartan II

Ok

SETEdit

Búsqueda
ctags

Plantillas
Pmacros y tpl2file

Edición
SETEdit

Bibliotecas y Cores

Automatizador
GNU Make

Otros
hex2vhdl
vhdlspp
natebege

Compilador
GHDL

Estilo (Lint)
bakalint

Forma de Onda
GTKWave y dinotrace

Testbench
GHDL

Recolección de errores

Error

SETEdit

No son muchos los editores libres que posean facilidades avanzadas para la edición de código VHDL. Uno de estos editores es el SETEdit, un editor pensado para programadores, con soporte para gran cantidad de lenguajes de programación. Estas son algunas de las características que lo hacen una buena elección para el trabajo con VHDL:

- Resaltado de sintáxis para VHDL.
- Macros específicas con construcciones típicas de VHDL (PMacros).
- Utilización de Exuberant C Tags con soporte específico para VHDL.
- Indentado coherente con los guidelines del proyecto.
- Configurable y con soporte para los lenguajes más populares.
- tpl2file: templates de Xilinx como

http://setedit.sourceforge.net

GNU/Make+SETEdit

Búsqueda
ctags

Plantillas
Pmacros y tpl2file

Edición
SETEdit

Bibliotecas y Cores

Automatizador
GNU Make

Otros
hex2vhdl
vhdlspp
natebege

Compilador
GHDL

Estilo (Lint)
bakalint

Forma de Onda
GTKWave y dinotrace

Testbench
GHDL

Recolección de errores

Error

GNU/Make

- Automatiza el proceso, incluyendo pasos específicos. Ejemplo: ensamblador.
- Integra con SETEdit para la recolección de errores.

GHDL

INTI

Búsqueda
ctags

Plantillas
Pmacros y tpl2file

Edición
SETEdit

Bibliotecas y Cores

Automatizador
GNU Make

Otros
hex2vhdl
vhdlspp
natebege

Compilador
GHDL

Estilo (Lint)
bakalint

Forma de Onda
GTKWave y dinotrace

Testbench
GHDL

Recolección de errores

Error

GHDL

Debido a su excelente soporte para VHDL y a la capacidad de compilar sin problemas proyectos tales como el procesador LEON y el DLX se ha seleccionado el GHDL como herramienta principal de simulación para VHDL.

http://ghdl.free.fr/

GHDL utiliza la tecnología del GCC, el compilador de software libre más utilizado en todo el mundo.

GTKWave

INTI

Búsqueda
ctags

Plantillas
Pmacros y tpl2file

Edición
SETEdit

Bibliotecas y Cores

Automatizador
GNU Make

Otros
hex2vhdl
vhdlspp
natebege

Compilador
GHDL

Estilo (Lint)
bakalint

Forma de Onda
GTKWave y dinotrace

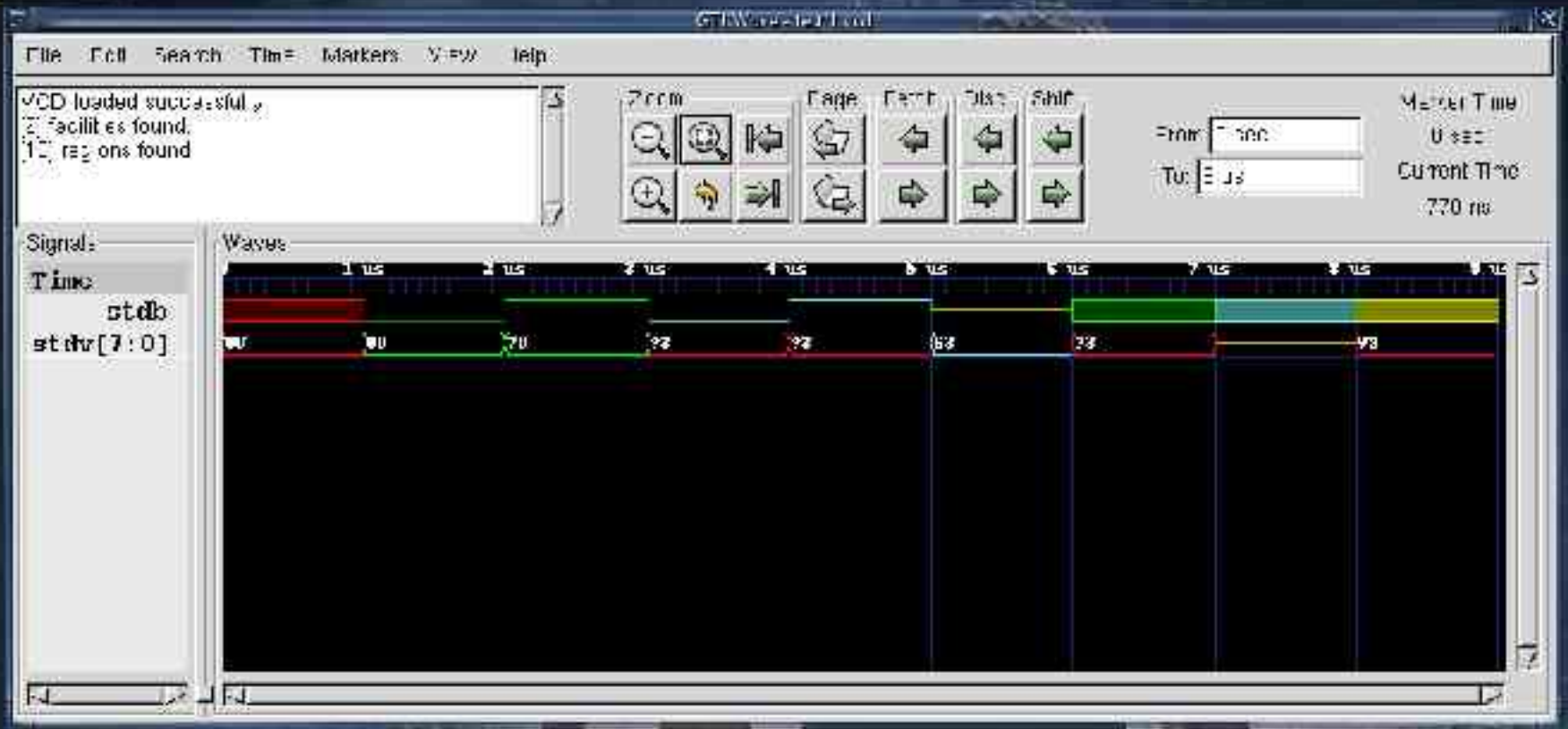
Testbench
GHDL

Recolección de errores

Error

GTKWave

- Interfaz de usuario simple escrita en GTK.
- Soporte especial para VHDL (originalmente sólo Verilog).
- Soporte para GHW



ISE WebPack

INTI

Testbench
GHDL

Forma de Onda
GTKWave y dinotrace

Error

Post-proceso
GCC y Perl

Recolección de errores

Síntesis
xst/map/par

No cumple

JTAG
JBit, naxjp, xilix_jtag

FPGA
Spartan II

Ok

Síntesis

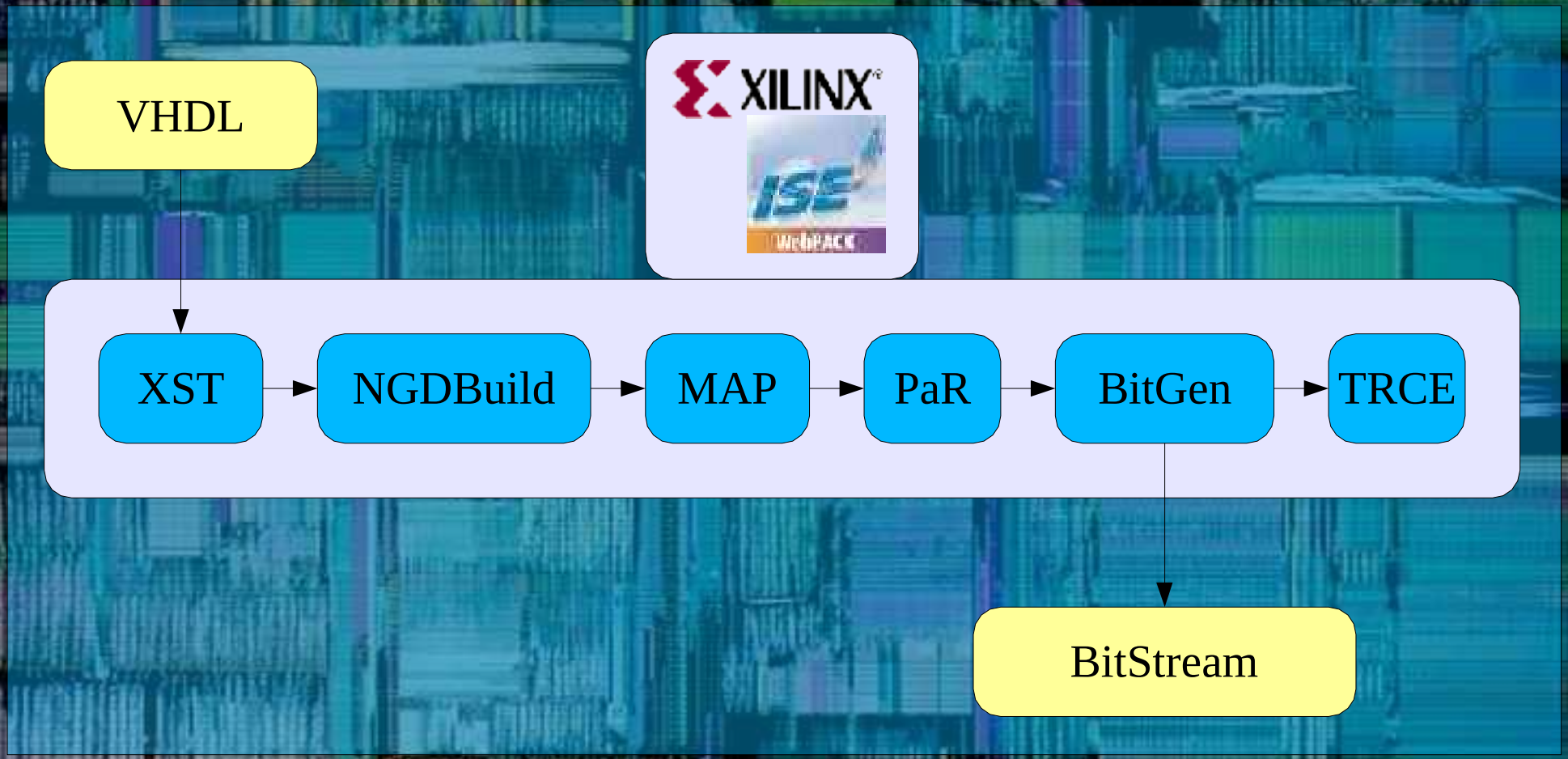
Lamentablemente no conocemos herramientas de síntesis con licencias free software u open source. Los dos fabricantes más importantes de dispositivos FPGA brindan herramientas de síntesis, cada uno para sus propios productos, de uso gratuito.

El proyecto FPGA Libre está abierto a cualquier tecnología y dispositivo FPGA, siempre que pueda lograrse su utilización bajo entornos de software libre, como es el caso de GNU/Linux.

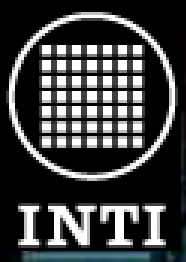
La herramienta de síntesis ISE WebPack de Xilinx es de uso gratuito (no libre), tiene una versión para sistemas GNU/Linux y permite su utilización con línea de comandos, lo que facilita la automatización del proceso con herramientas como GNU Make.

© 2005 INTI – Tropea, Brengi, Borgna

ISE WebPack



Soft JTAG



Forma de Onda
GTKWave y dinotrace

Testbench
GHDL

Recolección de errores

Error

Post-proceso
GCC y Perl

No cumple

Síntesis
xst/map/par

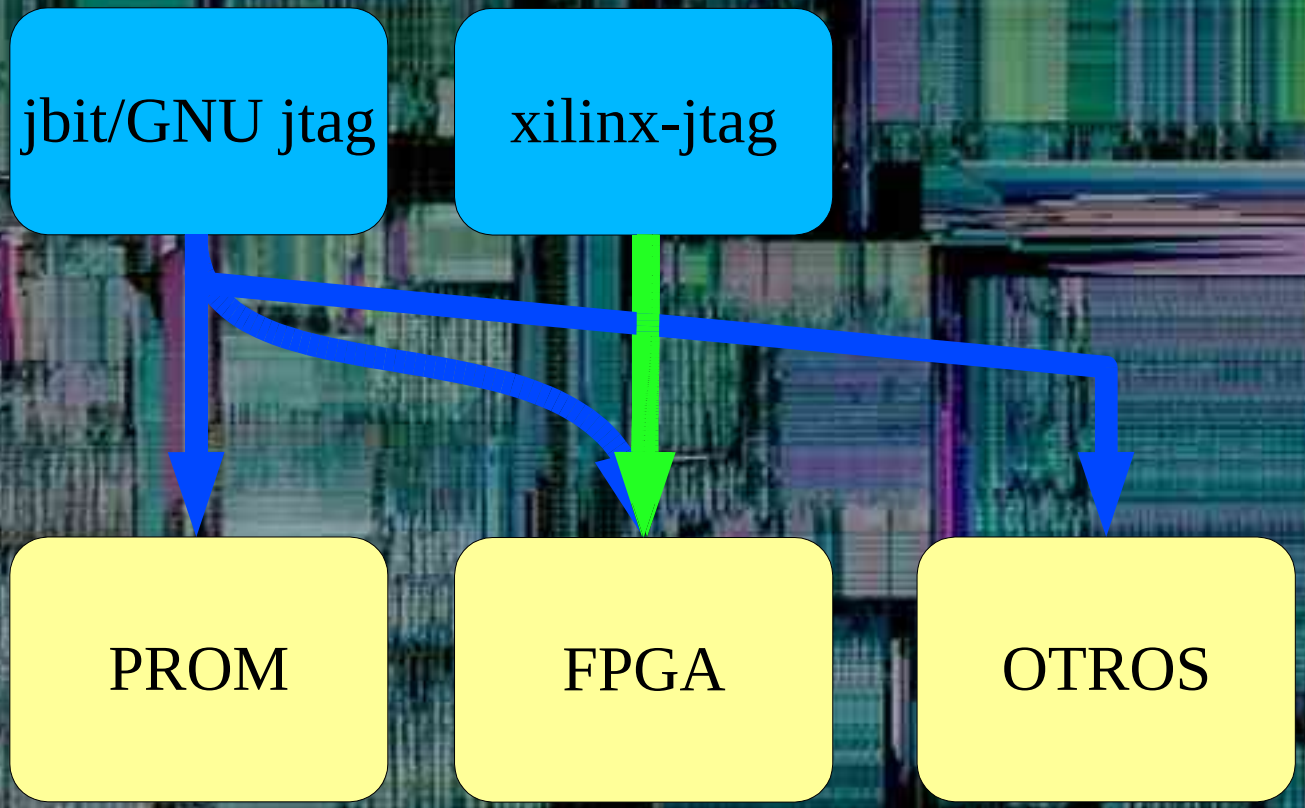
JTAG
JBit, naxjp, xilix_jtag

FPGA
Spartan II

Ok

Soft JTAG

- Estándar IEEE 1149.1 del JTAG (Joint Test Action Group).
- Originalmente pensado para testeo.
- Independiente del fabricante.



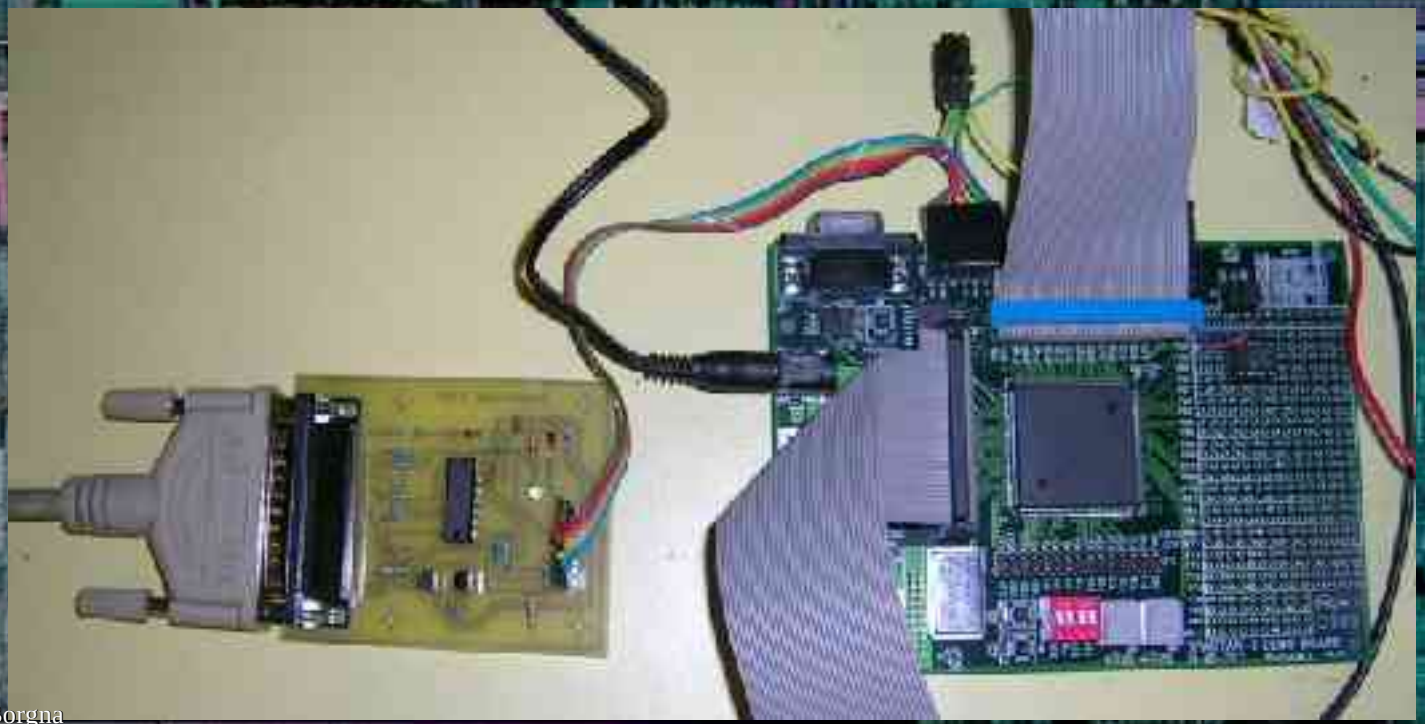
© 2005 INTI – Tropea, Brengi, Borgna

Hardware y FPGA: Programador JTAG

Para nuestros proyectos usamos un cable compatible con el cable Parallel III de Xilinx, también conocido como DLC5.

Este es un circuito simple y barato que se puede conectar al puerto paralelo de una PC.

Todos los archivos de diseño se encuentran disponibles en formato Kicad.



Soft JTAG

Testbench
GHDL

Forma de Onda
GTKWave y dinotrace

Error

Post-proceso
GCC y Perl

Recolección de errores

Síntesis
xst/map/par

No cumple

JTAG
JBit, naxjp, xilix_jtag

FPGA
Spartan II

Ok

Hardware y FPGA: Placa de desarrollo

Para desarrollar una aplicación, además del chip FPGA es necesaria una electrónica de soporte: Circuito impreso, circuitos de alimentación, memoria, conectores, etc.

La forma más fácil de abordar este tema es comprando algún kit de desarrollo para FPGA (Plaqueta con FPGA). Nuestro primer kit de desarrollo comprado es una placa Memec Spartan II LC.

Hardware y FPGA: Placa de hardware

El intercambio y la copia de circuitos físicos (hardware) es complejo debido al costo de replicación. Aquí ya no pueden aplicarse los mismos criterios que para el software libre.

Sin embargo, sí puede mantenerse el mismo espíritu con los archivos y toda la información de diseño asociada a un circuito. A este concepto se lo llama "Hardware Libre", Hardware Abierto" o "Free Hardware". En este punto es muy importante comprender que las herramientas de software para realizar estos diseños también deberían ser libres.

Para cumplir este propósito con el desarrollo de circuitos electrónicos bajo esta modalidad se ha seleccionado el software KICAD:

http[illegible]g.fr/realise/au[illegible]lis/kicad/.

KICAD GPL PCB SUITE

Ejemplo práctico

Con la finalidad de:

- Familiarizarnos con la tecnología.
- Validar las herramientas.
- Buscar metodologías para depurar los errores.

Encaramos los siguientes desarrollos.

PIC16C84

- Microcontrolador tipo RISC
- Instrucciones de 14 bits

Original: MICROCHIP

- **2,5 MIPS (50xENIAC)**
- **36 bytes de memoria**
- **13 pines de I/O**

Sitentizado:

- **7,5 MIPS (150xENIAC)**
- **464 bytes de memoria**
- **24 pines de I/O**

WISHBONE COMPATIBLE

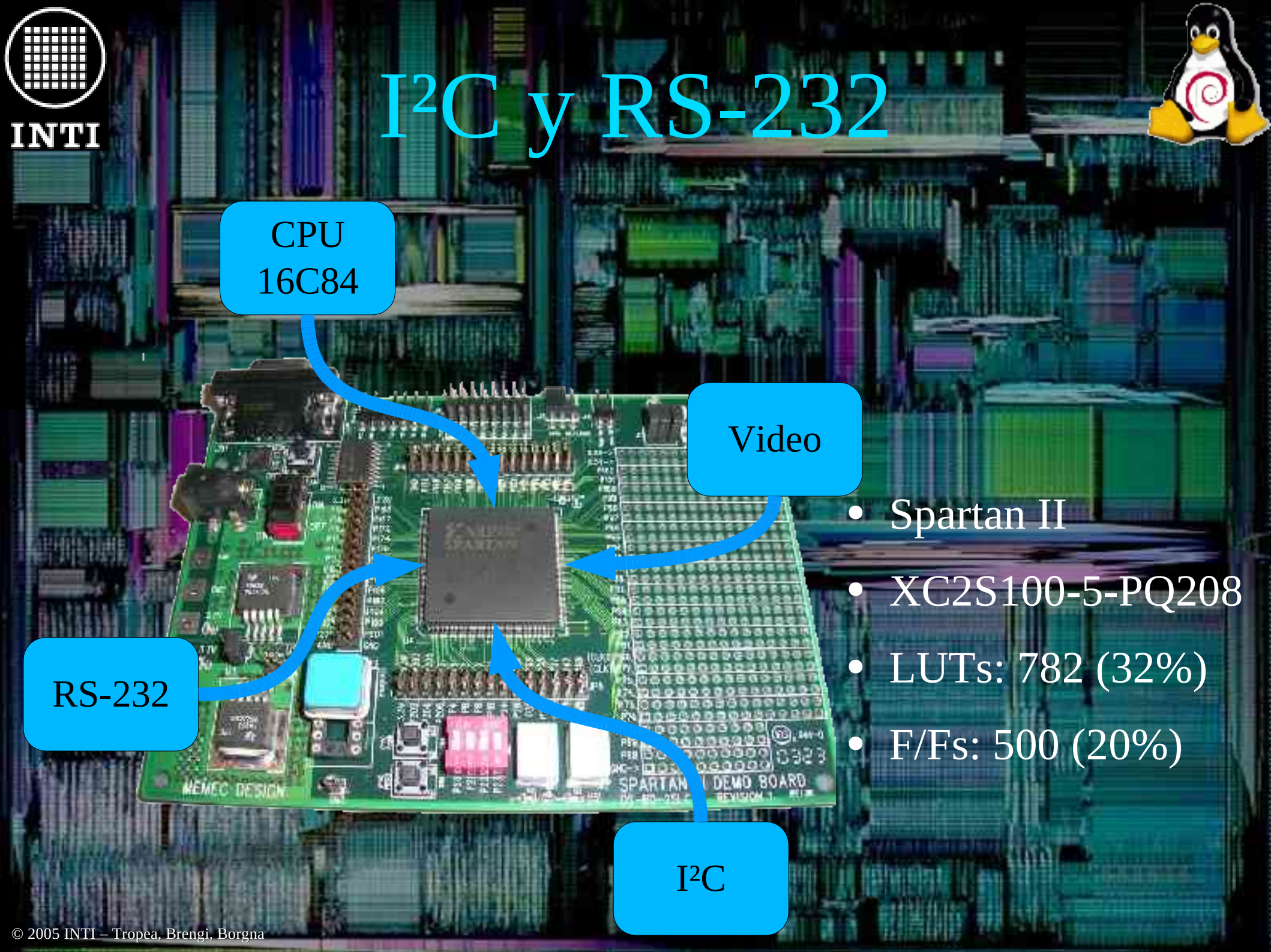
OPENCORES.ORG

Video

- Controlador de video en modo texto.
- Señales compatibles con VGA.
- 40 x 25 caracteres.
- 8 colores de fondo y primer plano
- 64 caracteres diferentes.

- Comunicación sincrónica serie I²C. Probado a 367 kHz con una memoria EEPROM 24LC02B
- Comunicación asincrónica serie (RS-232). Probado con una PC.
- Cores de OpenCores.org con pocas modificaciones.

I²C y RS-232



- Spartan II
- XC2S100-5-PQ208
- LUTs: 782 (32%)
- F/Fs: 500 (20%)

FPGALibre

- Facilitar el intercambio de conocimientos y *cores*.
- Impulsar el uso de herramientas de S.L.
- Hosteado por SourceForge
- fpgalibre.sourceforge.net
- Abierto (OSs y tecnologías)
- Actualmente basado en Debian GNU/Linux

Contacto

Integrantes:

- Borgna, Juan P. D. <jpborgna@inti.gov.ar>
- Brengi, Diego J. <brengi@inti.gov.ar>
- Trapanotto, Andrés <andres_t@inti.gov.ar>
- Tropea, Salvador E. <salvador@inti.gov.ar>

¡Muchas gracias!