Tema 12 - Arquitectura de un soft processor

Sistemas Electrónicos para Automatización

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Hipólito Guzmán Miranda

Tema 12 - El procesador neorv32

Sistemas Electrónicos para Automatización

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Hipólito Guzmán Miranda

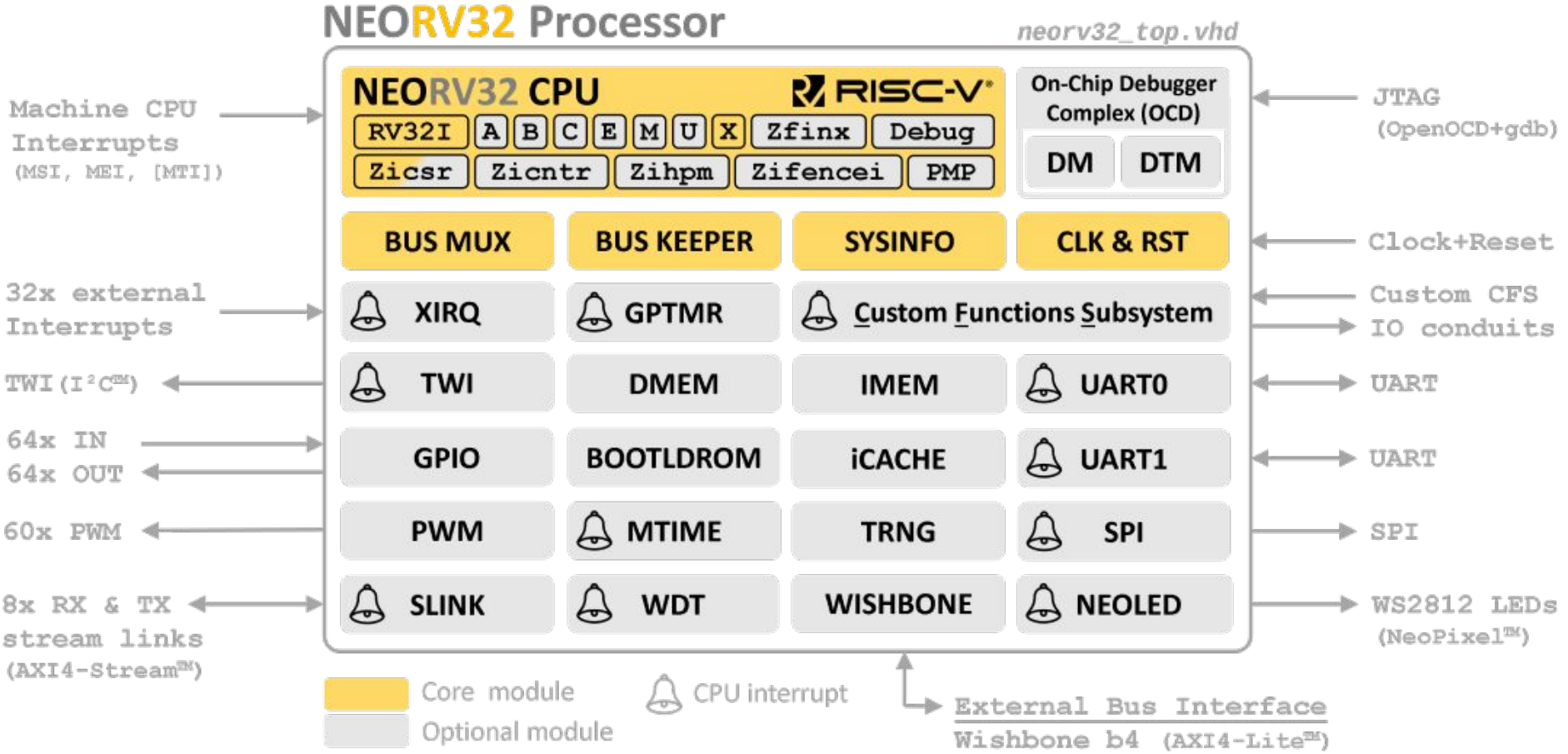
Contenido

- Arquitectura
- Buses
- Configuración
- Herramientas para implementación hardware y compilación software
- Periféricos personalizados por el usuario
- Conclusiones

Contenido

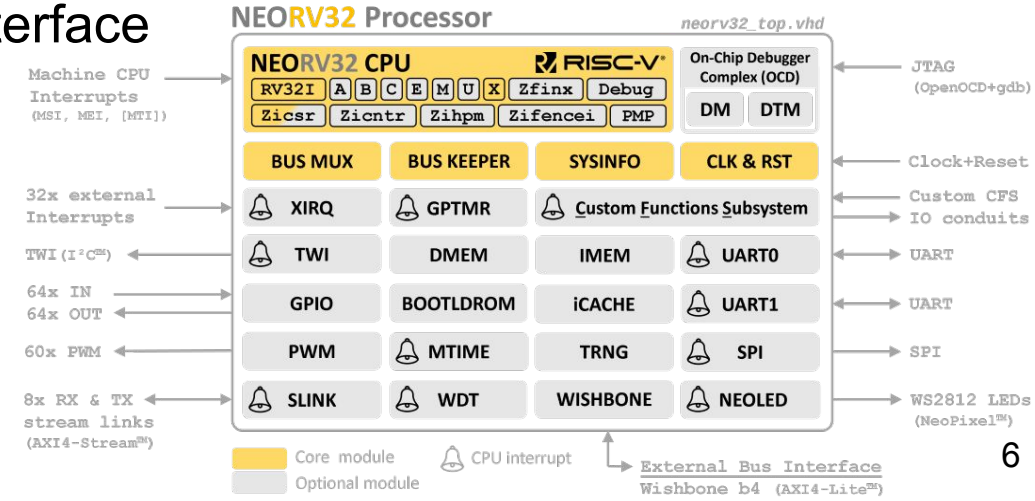
- Arquitectura
- Buses
- Configuración
- Herramientas para implementación hardware y compilación software
- Periféricos personalizados por el usuario
- Conclusiones

Arquitectura



Siglas

- I-* : Instruction *, D-*: Data *
- TWI: Twin-Wire Interface
- TRNG: True Random Number Generator
- DM: Debug Module
- DTM: Debug Transport Module
- GPTMR: General Purpose Timer
- MTIME: Machine System Timer
- NEOLED: Smart Led Interface



Siglas (II)

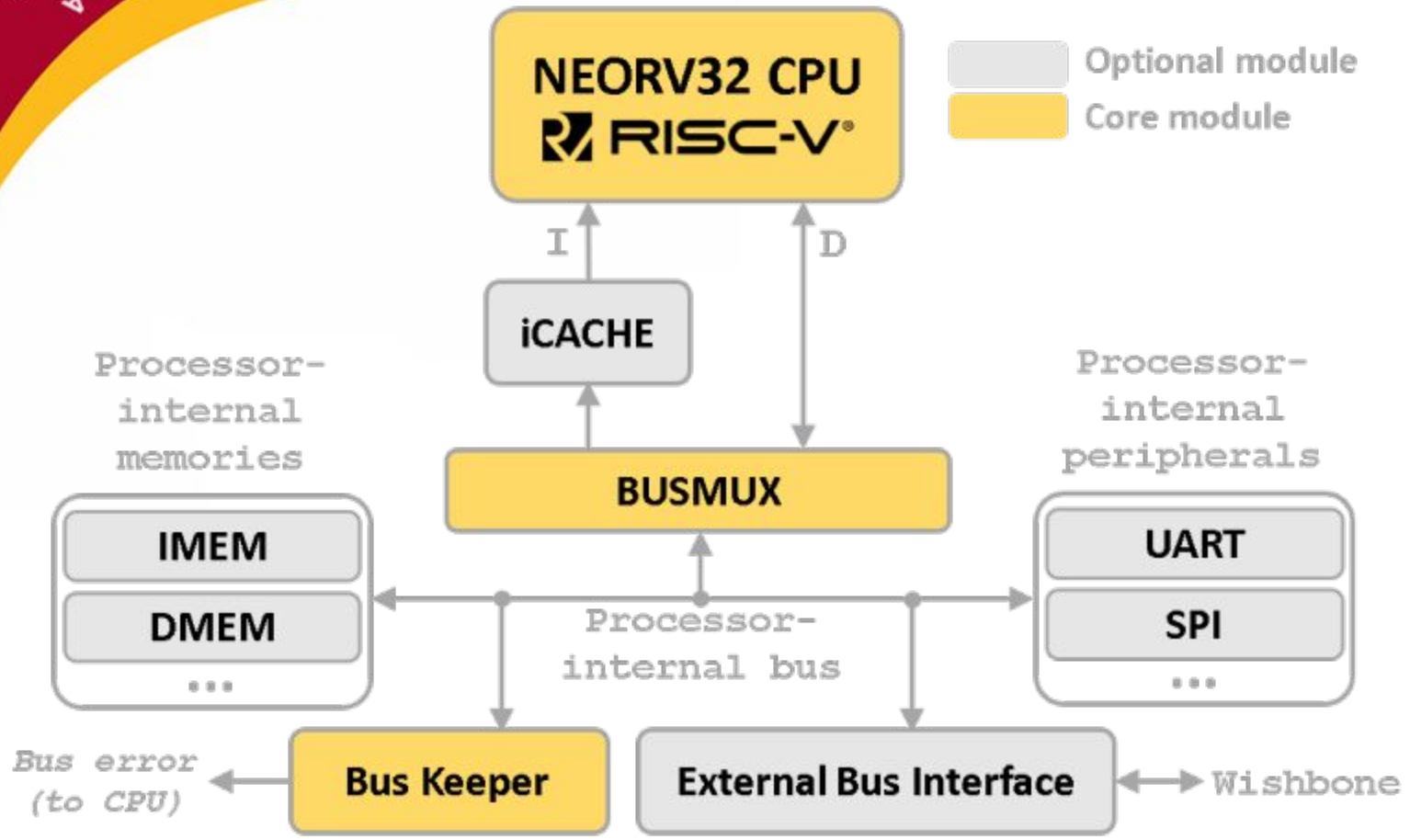
- RV32I: RISC-V Base Integer Instruction Set, 32-bit
- A: Standard extension for Atomic instructions
- B: Standard Extension for Bit Manipulation
- C: Standard Extension for Compressed Instructions
- E: Embedded RF Extension
- M: Standard Extension for Integer Multiplication and Division
- U: User Mode Extension
- X: NeoRV32 specific extension
- Zfinx: 32 bit floating-point extension
- Zicsr: Control and Status Register
- Zicntr: CPU Base Counters
- Zihpm: Hardware Performance Monitors
- Zifencei: Instruction Stream Synchronization
- Debug: CPU Debug Mode
- PMP: Physical Memory Protection

RISC-V es un ISA (Instruction Set Architecture) extensible

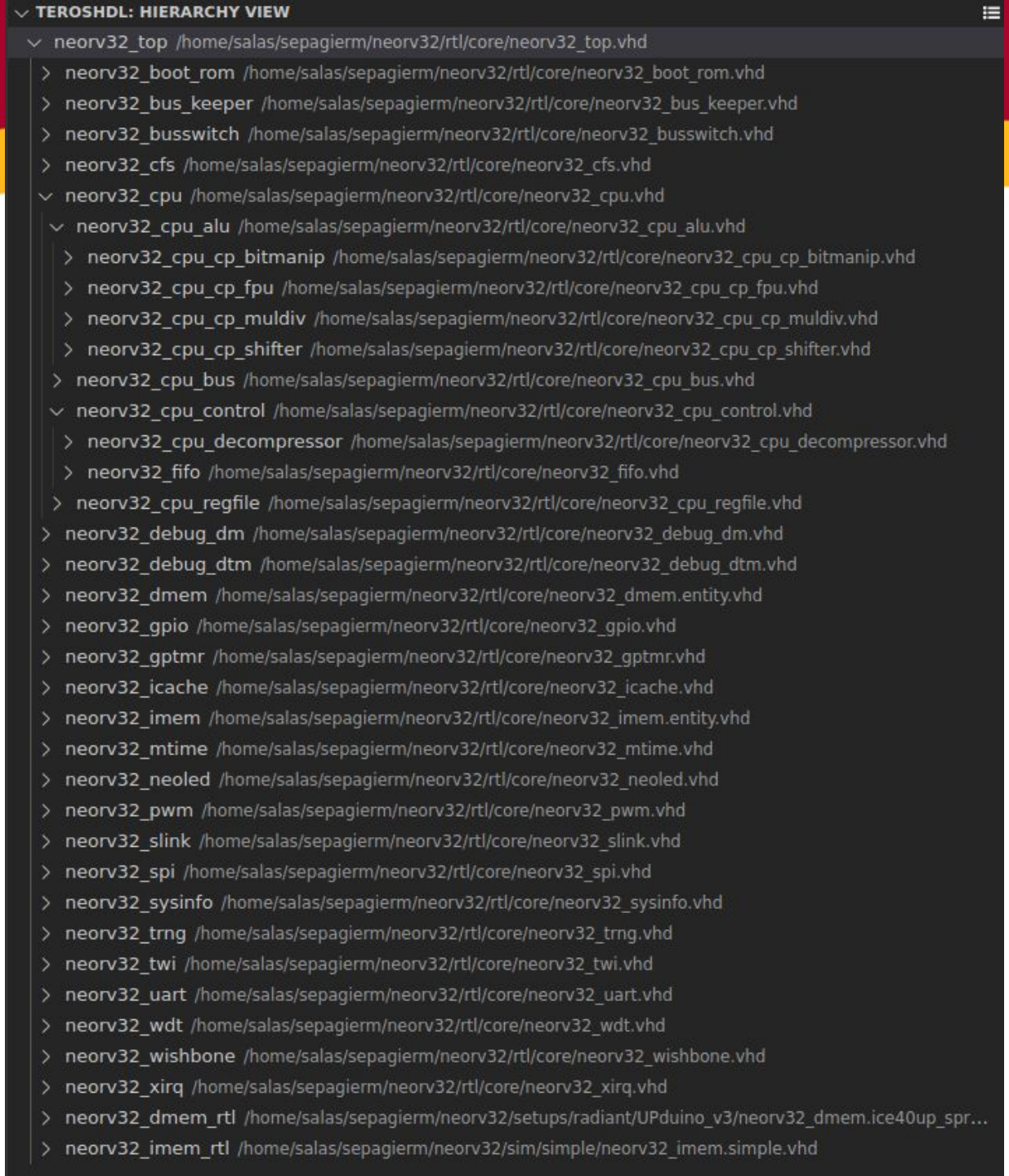
Arquitectura

- Ya que es un soft processor, su arquitectura es configurable
- Bloques básicos (‘core modules’, en amarillo) y bloques opcionales (‘optional modules’, en gris)
- Cachés, multiplicador, FPU, Divisor, etc, opcionales
- El ‘coste’ es en recursos ocupados en la FPGA

Arquitectura



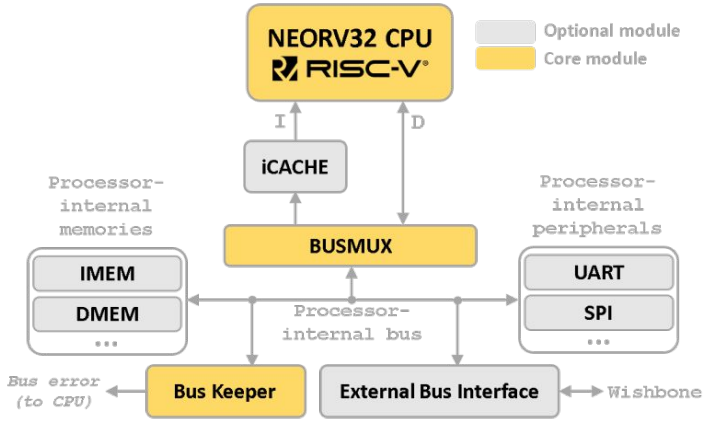
Jerarquía


TEROSHDL: HIERARCHY VIEW
neorv32_top /home/salas/sepagierm/neorv32/rtl/core/neorv32_top.vhd
neorv32_boot_rom /home/salas/sepagierm/neorv32/rtl/core/neorv32_boot_rom.vhd
neorv32_bus_keeper /home/salas/sepagierm/neorv32/rtl/core/neorv32_bus_keeper.vhd
neorv32_busswitch /home/salas/sepagierm/neorv32/rtl/core/neorv32_busswitch.vhd
neorv32_cfs /home/salas/sepagierm/neorv32/rtl/core/neorv32_cfs.vhd
neorv32_cpu /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu.vhd
neorv32_cpu_alu /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_alu.vhd
neorv32_cpu_cp_bitmanip /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_cp_bitmanip.vhd
neorv32_cpu_cp_fpu /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_cp_fpu.vhd
neorv32_cpu_cp_muldiv /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_cp_muldiv.vhd
neorv32_cpu_cp_shifter /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_cp_shifter.vhd
neorv32_cpu_bus /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_bus.vhd
neorv32_cpu_control /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_control.vhd
neorv32_cpu_decompressor /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_decompressor.vhd
neorv32_fifo /home/salas/sepagierm/neorv32/rtl/core/neorv32_fifo.vhd
neorv32_cpu_regfile /home/salas/sepagierm/neorv32/rtl/core/neorv32_cpu_regfile.vhd
neorv32_debug_dm /home/salas/sepagierm/neorv32/rtl/core/neorv32_debug_dm.vhd
neorv32_debug_dtm /home/salas/sepagierm/neorv32/rtl/core/neorv32_debug_dtm.vhd
neorv32_dmem /home/salas/sepagierm/neorv32/rtl/core/neorv32_dmem.entity.vhd
neorv32_gpio /home/salas/sepagierm/neorv32/rtl/core/neorv32_gpio.vhd
neorv32_gptmr /home/salas/sepagierm/neorv32/rtl/core/neorv32_gptmr.vhd
neorv32_icache /home/salas/sepagierm/neorv32/rtl/core/neorv32_icache.vhd
neorv32_imem /home/salas/sepagierm/neorv32/rtl/core/neorv32_imem.entity.vhd
neorv32_mtime /home/salas/sepagierm/neorv32/rtl/core/neorv32_mtime.vhd
neorv32_neoled /home/salas/sepagierm/neorv32/rtl/core/neorv32_neoled.vhd
neorv32_pwm /home/salas/sepagierm/neorv32/rtl/core/neorv32_pwm.vhd
neorv32_slink /home/salas/sepagierm/neorv32/rtl/core/neorv32_slink.vhd
neorv32_spi /home/salas/sepagierm/neorv32/rtl/core/neorv32_spi.vhd
neorv32_sysinfo /home/salas/sepagierm/neorv32/rtl/core/neorv32_sysinfo.vhd
neorv32_trng /home/salas/sepagierm/neorv32/rtl/core/neorv32_trng.vhd
neorv32_twi /home/salas/sepagierm/neorv32/rtl/core/neorv32_twi.vhd
neorv32_uart /home/salas/sepagierm/neorv32/rtl/core/neorv32_uart.vhd
neorv32_wdt /home/salas/sepagierm/neorv32/rtl/core/neorv32_wdt.vhd
neorv32_wishbone /home/salas/sepagierm/neorv32/rtl/core/neorv32_wishbone.vhd
neorv32_xirq /home/salas/sepagierm/neorv32/rtl/core/neorv32_xirq.vhd
neorv32_dmem_rtl /home/salas/sepagierm/neorv32/setups/radiant/UPduino_v3/neorv32_dmem.ice40up_spr...
neorv32_imem_rtl /home/salas/sepagierm/neorv32/sim/simple/neorv32_imem.simple.vhd

Contenido

- Arquitectura
- Buses
- Configuración
- Herramientas para implementación hardware y compilación software
- Periféricos personalizados por el usuario
- Conclusiones

Buses



- Buses de Datos separados de buses de Instrucciones en el core
- Pero luego se unen en un único bus
  - “Modified Von Neumann architecture”
- Processor-internal bus para comunicar los distintos elementos
- Bus Wishbone para los periféricos que añadamos

Buses para periféricos

Dos opciones:

- Wishbone
- AXI (Advanced eXtensible Interface)

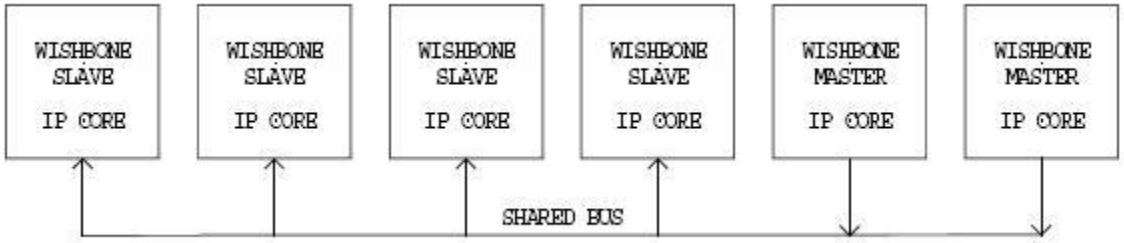
El desarrollador de neorv32 proporciona un ‘wrapper’ para conectar periféricos AXI al bus Wishbone

El bus Wishbone

- Un bus ‘lógico’, es decir, que no especifica al respecto de niveles de tensión
  - Pensado para diseños HDL
- Estándar de facto para diseños HW libres
- Muchos cores en opencores.org y github.com son “Wishbone compliant”

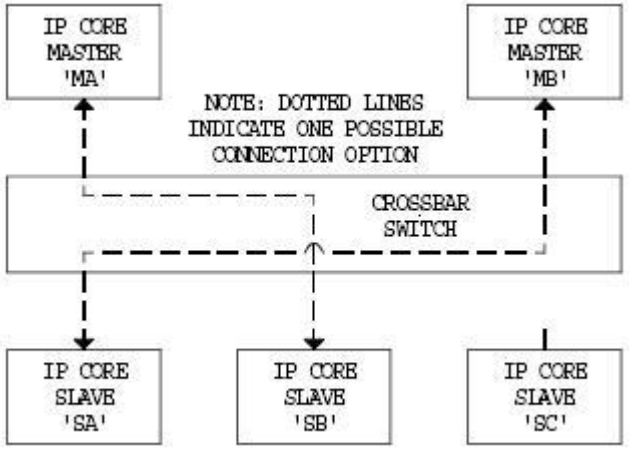
El bus wishbone

Permite múltiples topologías

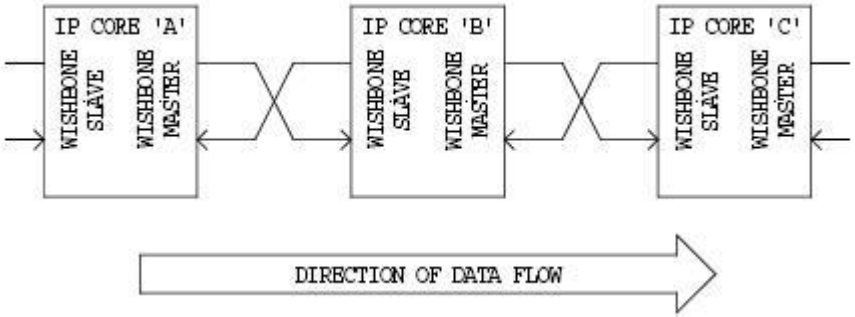


shared bus

crossbar switch



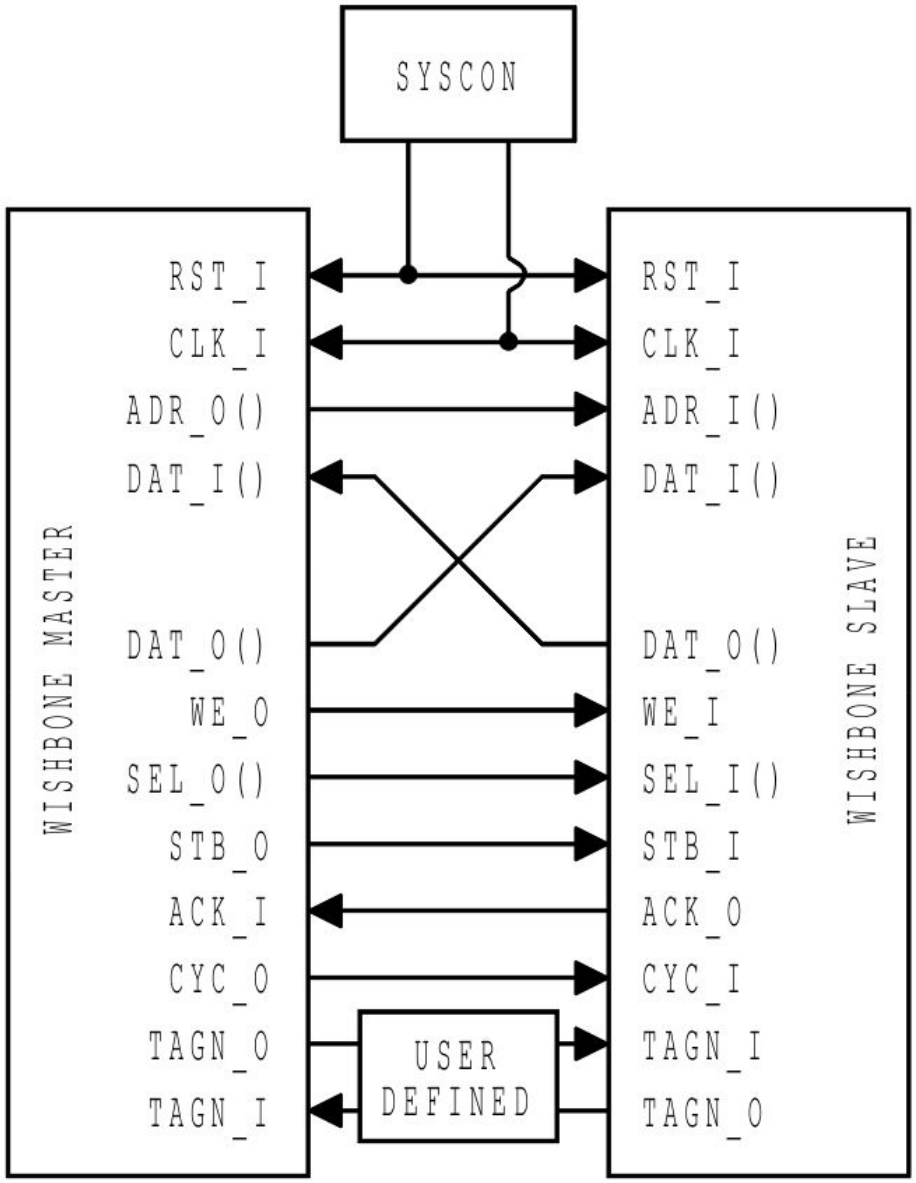
dataflow/pipeline



Buses

reset
clock
address
data_in

data_out
write_enable
select output
strobe
acknowledge
cycles
tag



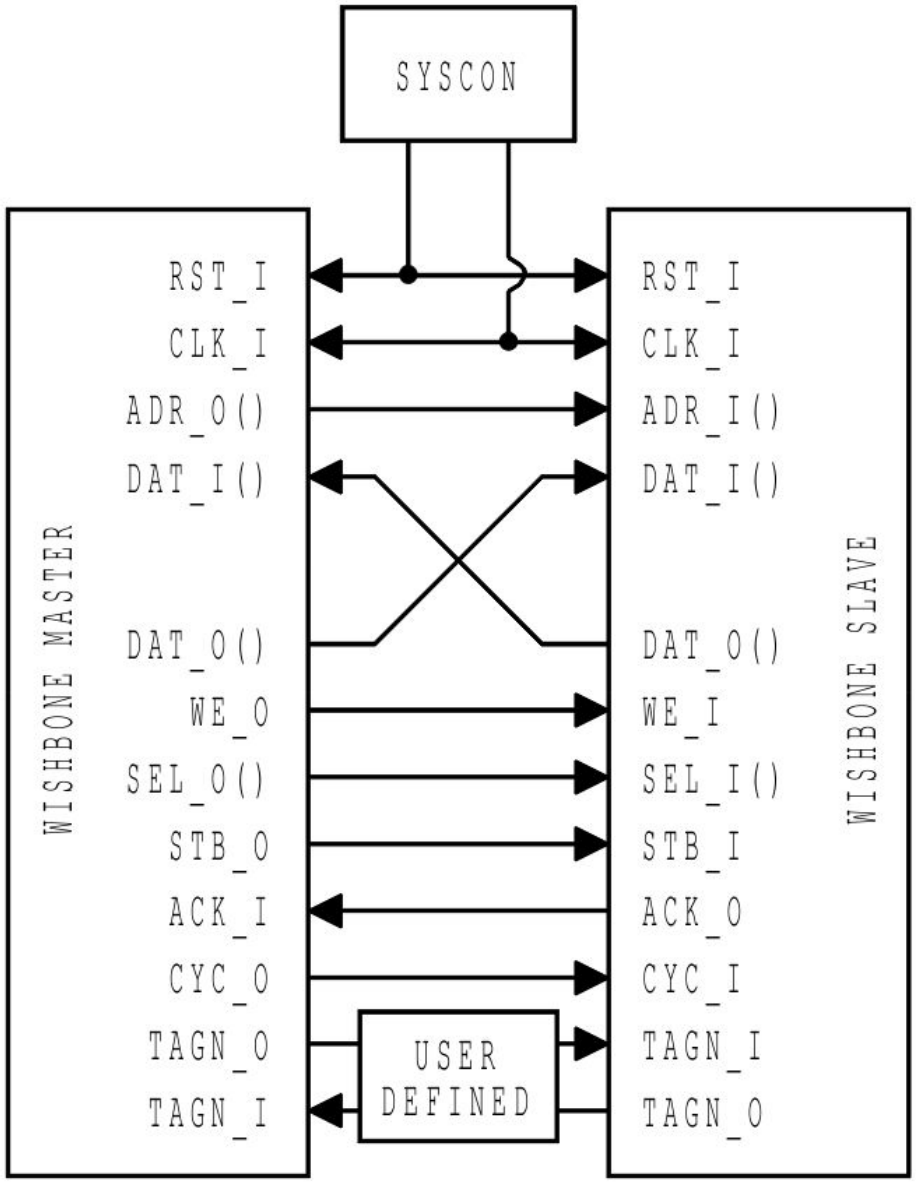
Buses

rst y clk son entradas para master y slaves

master pone addr, data, we, sel, stb, cyc al slave

slave devuelve data, ack

tag* son definidas por el usuario



Señales de Wishbone

- **clk, rst, adr** (address)**, dat** (data)**:** auto-explicativas
- **we** (write enable)**:** indica si el ciclo de bus es de lectura (‘0’) o escritura (‘1’)
- **sel** (select input array): indica en qué bytes de **dat** hay datos válidos (es realmente un *byte enable*)

Señales de Wishbone

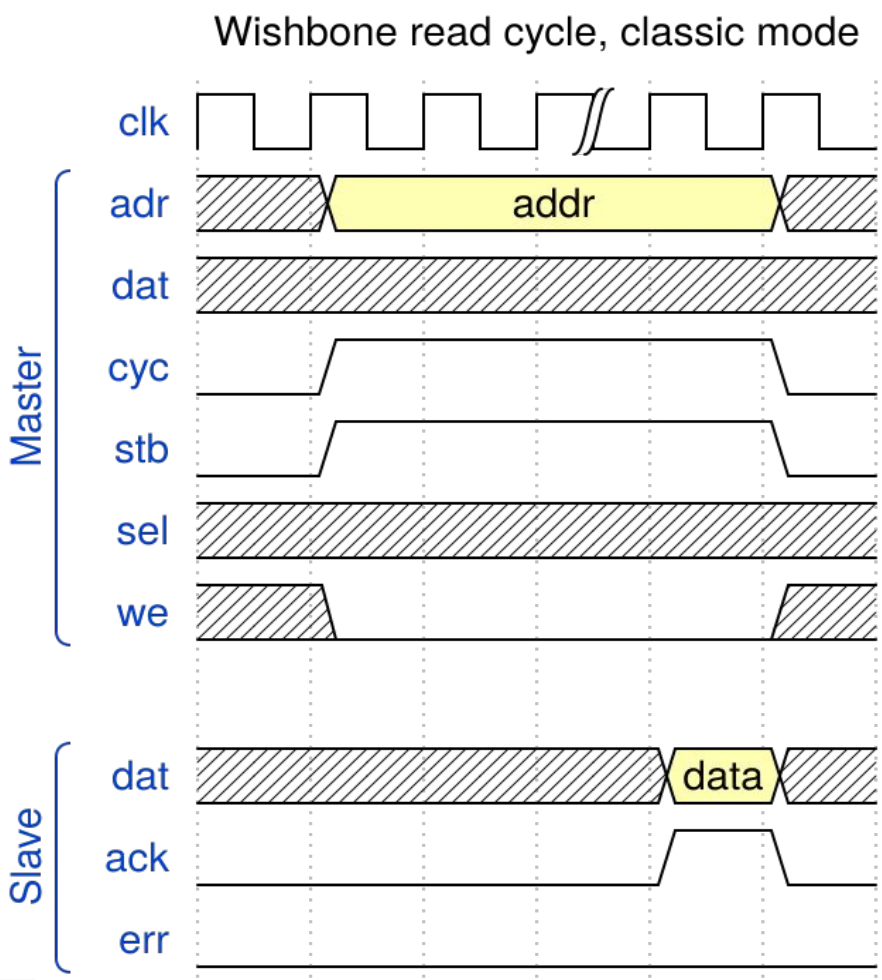
- **cyc** (cycle input): indica que hay un ciclo de bus válido en progreso
- **stb** (strobe): indica al esclavo que está seleccionado. El esclavo responde activando **ack**, u opcionalmente **err** (error) o **rty** (retry)
- **ack** (acknowledge): indica fin de un ciclo de bus normal

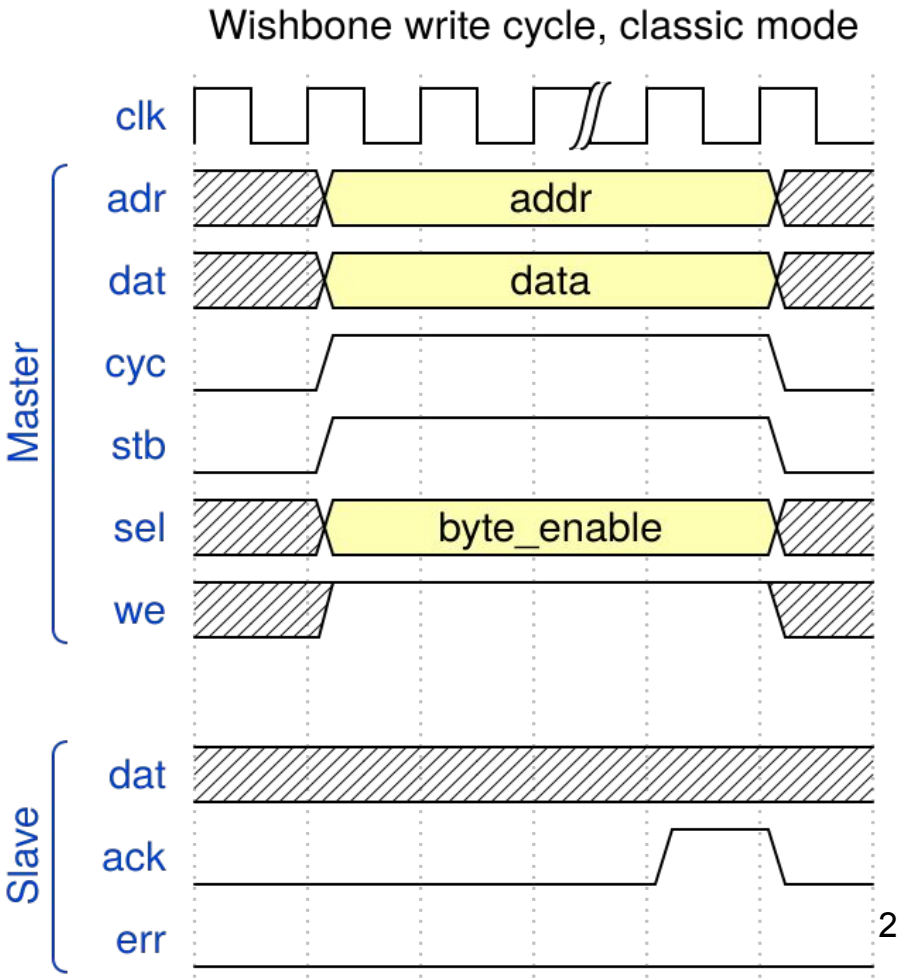
Señales (opcionales) de Wishbone

- **tagN** (tag, etiqueta): señales opcionales: **tga** (tag address), **tgd** (tag data), **tgc** (cycle)
- **lock**: el ciclo actual no se puede interrumpir
- **rty**: el interfaz no está listo, se debe reintentar el ciclo de bus
- **err**: terminación anormal de ciclo de bus
- **stall**: el esclavo no puede aceptar más transacciones en su cola (sólo en modo pipeline)

Más info en la especificación de Wishbone!

El bus Wishbone





AXI (Advanced eXtensible Interface)

- AXI es una especificación de interconexión
- Mayores anchos de banda
- Es parte de AMBA (Advanced Microprocessor Bus Architecture)
  - Junto con AHB (Advanced High-performance Bus) y APB (Advanced Peripheral Bus)
- Utilizado también en FPGA/SoC de Xilinx
- En neorv32, usable a través de un wrapper AXI-Wishbone

Contenido

- Arquitectura
- Buses
- Configuración
- Herramientas para implementación hardware y compilación software
- Periféricos personalizados por el usuario
- Conclusiones

Configuración

- neorv32 está descrito en VHDL
- La configuración se realiza dando valor a los GENERICS del microprocesador al instanciarlo
- Si el tamaño de la memoria de datos (DMEM) no es 8kB, es necesario modificar el script de linkado

Generics

Descritos en el datasheet:

- CLOCK_FREQUENCY, INT_BOOTLOADER_EN, CPU_EXTENSION_RISCV_*, MEM_INT_IMEM_EN, MEM_INT_IMEM_SIZE, MEM_INT_DMEM_EN, MEM_INT_DMEM_SIZE, IO_GPIO_EN, etc...

Configuración

```
-- ------------------------------------------------------------------------------
-- Instance the microprocessor
-- ------------------------------------------------------------------------------

neorv32_inst: entity neorv32.neorv32_top
generic map (
  -- General --
  CLOCK_FREQUENCY              => 12_000_000, -- clock frequency of clk_i in Hz
  INT_BOOTLOADER_EN            => true,       -- boot configuration: true = boot explicit bootloade
  HW_THREAD_ID                 => 0,          -- hardware thread id (32-bit)

  -- On-Chip Debugger (OCD) --
  ON_CHIP_DEBUGGER_EN          => false,      -- implement on-chip debugger?

  -- RISC-V CPU Extensions --
  CPU_EXTENSION_RISCV_A        => true,       -- implement atomic extension?
  CPU_EXTENSION_RISCV_C        => true,       -- implement compressed extension?
  CPU_EXTENSION_RISCV_E        => false,      -- implement embedded RF extension?
  CPU_EXTENSION_RISCV_M        => true,       -- implement mul/div extension?
  CPU_EXTENSION_RISCV_U        => false,      -- implement user mode extension?
  CPU_EXTENSION_RISCV_Zfinx    => false,      -- implement 32-bit floating-point extension (using
  CPU_EXTENSION_RISCV_Zicsr    => true,       -- implement CSR system?
  CPU_EXTENSION_RISCV_Zicntr   => true,       -- implement base counters?
```

MEM_INT_DMEM_EN, MEM_INT_DMEM_SIZE, IO_GPIO_EN, etc...

Contenido

- Arquitectura
- Buses
- Configuración
- Herramientas para implementación hardware y compilación software
- Periféricos personalizados por el usuario
- Conclusiones

Software para implementación

Síntesis: yosys + ghdlsynth (front-end vhdl)

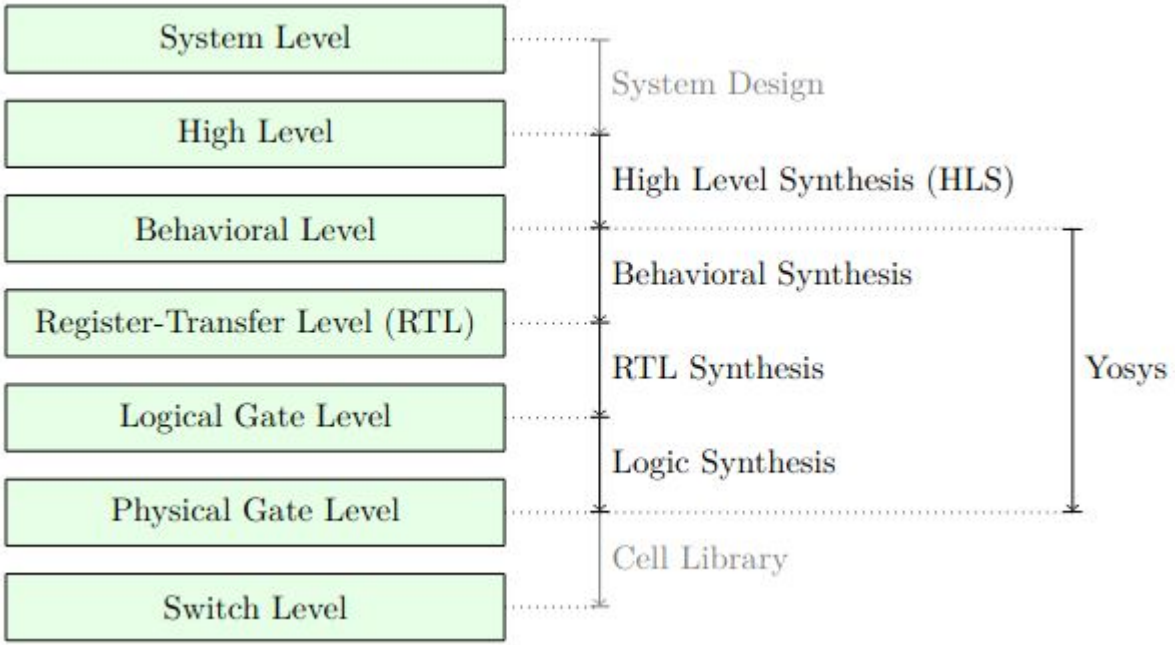
Place & Route: nextpnr

Generación bitstream: icestorm

Gestión del build: Makefiles (GNU Make)

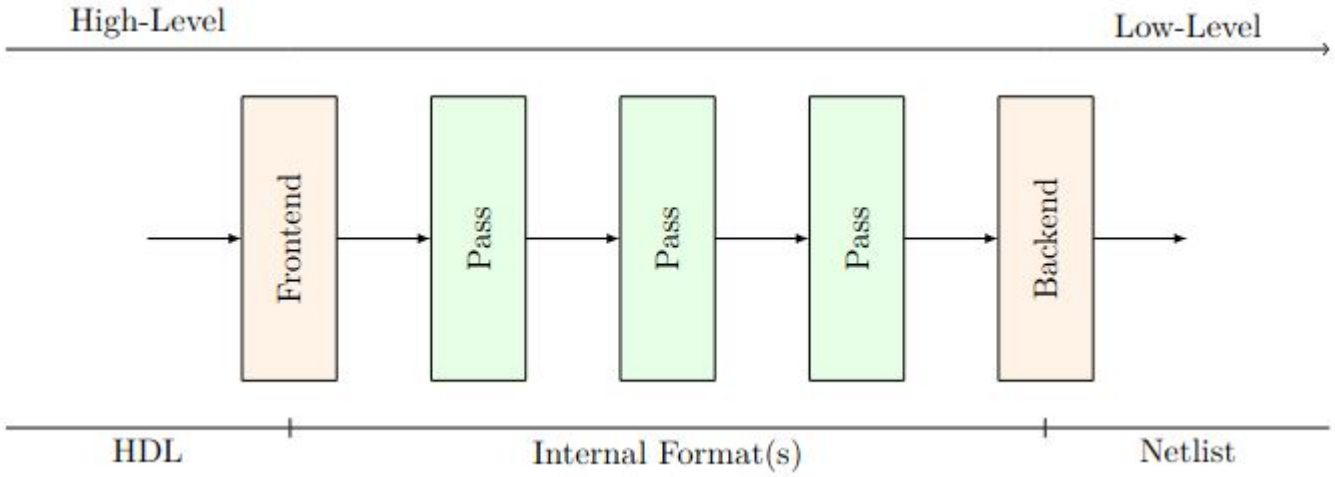
Yosys: Yosys Open SYnthesis Suite

- Herramienta de síntesis de código abierto
- Licencia ISC (permisiva)
  - ISC: Internet Systems Consortium



Yosys: Yosys Open SYnthesis Suite

- Estructurado en ‘pasadas’ (passes)
- Front-end + número de pasadas + back-end

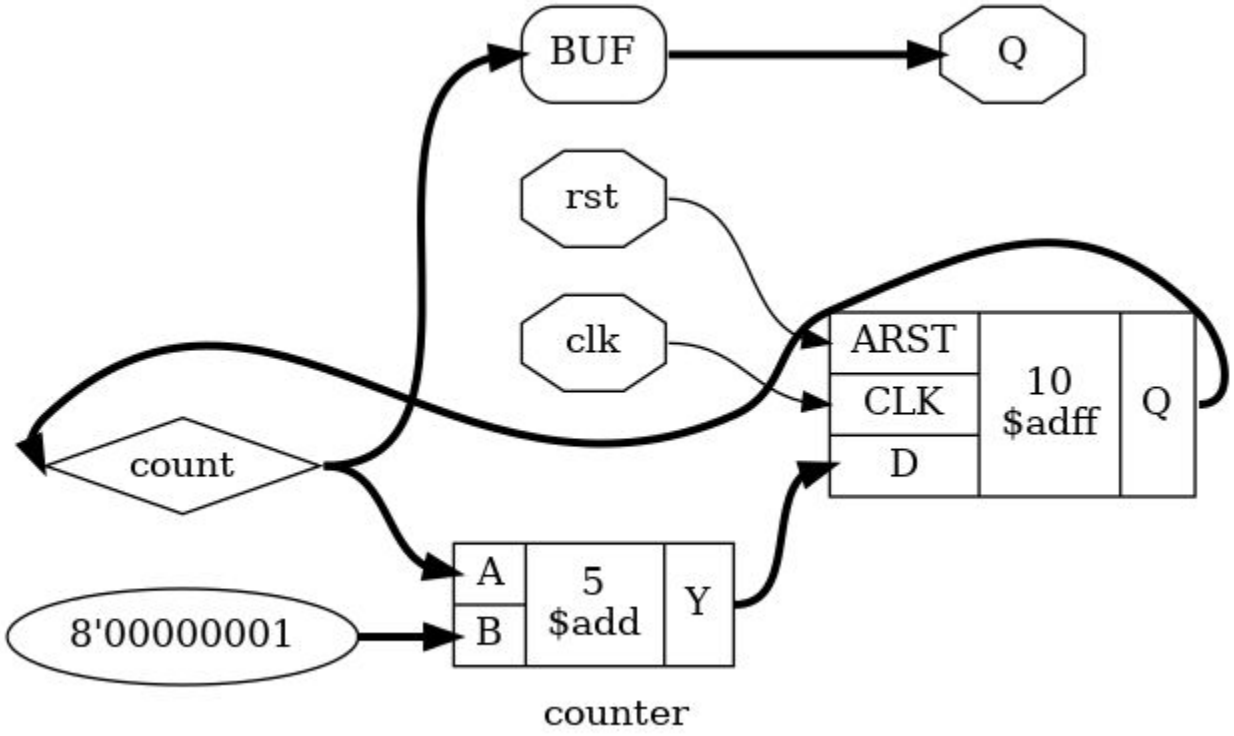


Yosys: Yosys Open SYnthesis Suite

- Front-ends:
  - Verilog (por defecto)
  - ghdlsynth (VHDL, open source)
  - Verific (VHDL + SystemVerilog, propietario, requiere licencia)
  - json, ast y otros para interoperabilidad con otras herramientas
- Back-ends:
  - Json, edif, verilog, y muchos otros
- Posibilidad de añadir ‘custom passes’, front-end y back-ends usando C++
  - Y también usando python, con pyosys

Yosys: ejemplo de uso

```
$ yosys -m ghdl
yosys> ghdl counter.vhd -e counter
yosys> show
```



Software para

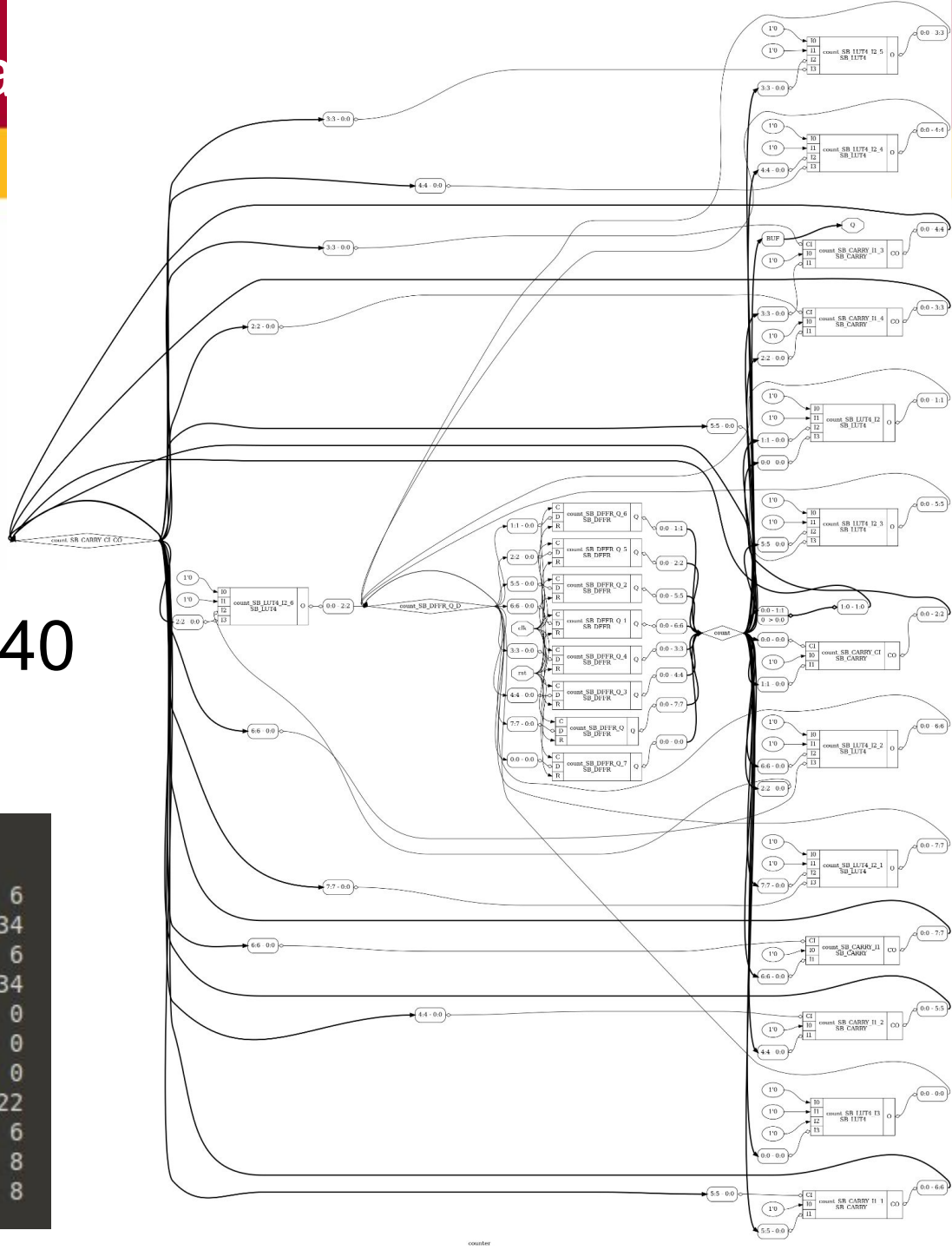
Yosys: ejemplo de uso (cont)

yosys> synth_ice40

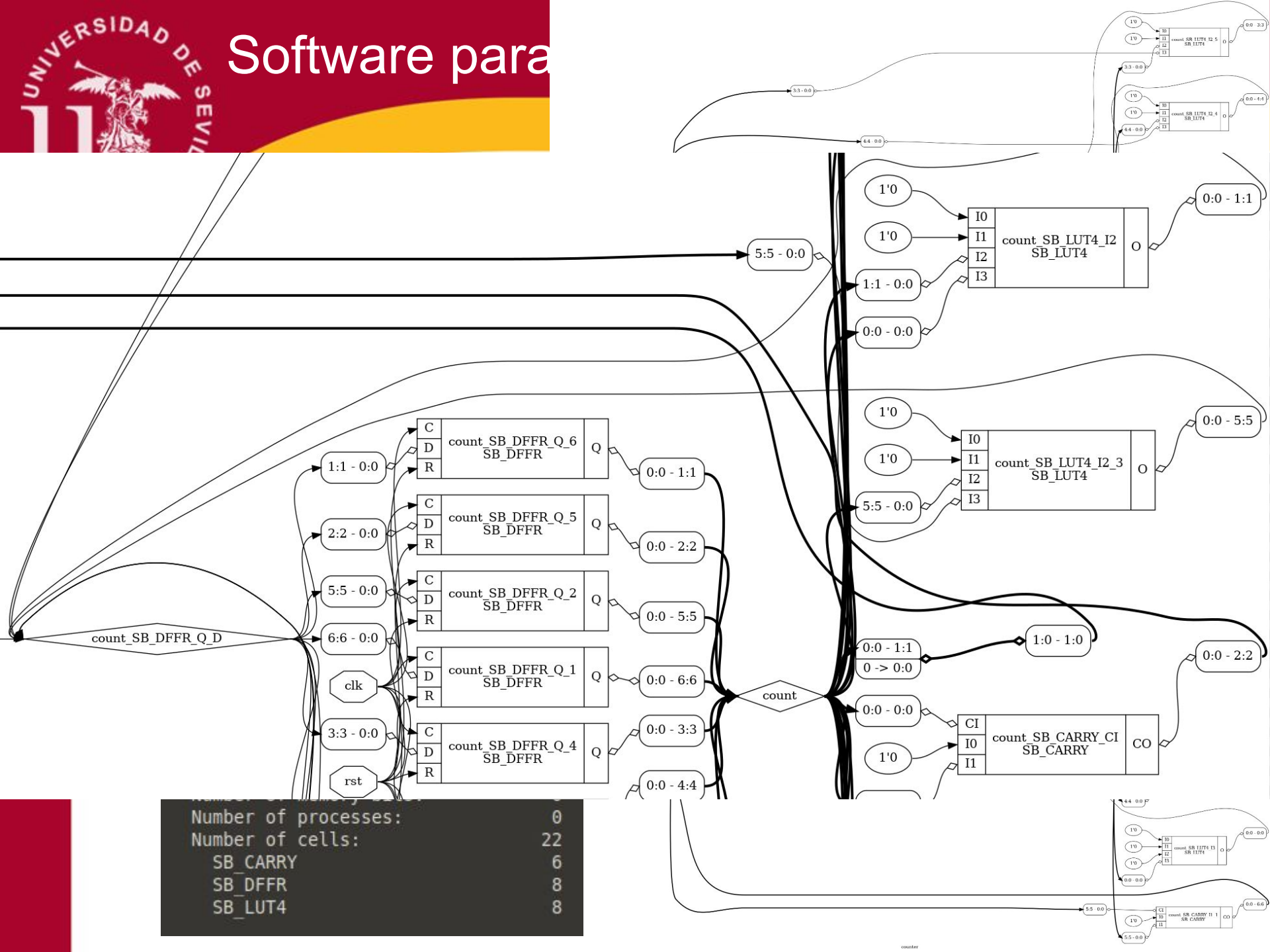
yosys> show

```
=== counter ===

   Number of wires:                 6
   Number of wire bits:            34
   Number of public wires:          6
   Number of public wire bits:     34
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                22
     SB_CARRY                       6
     SB_DFFR                        8
     SB_LUT4                        8
```



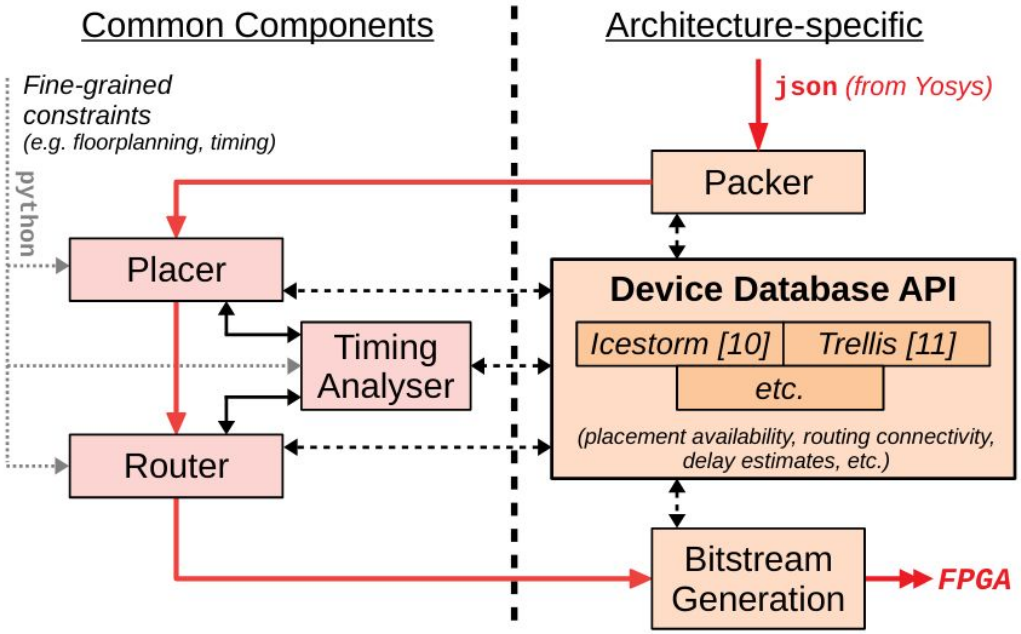
Software para

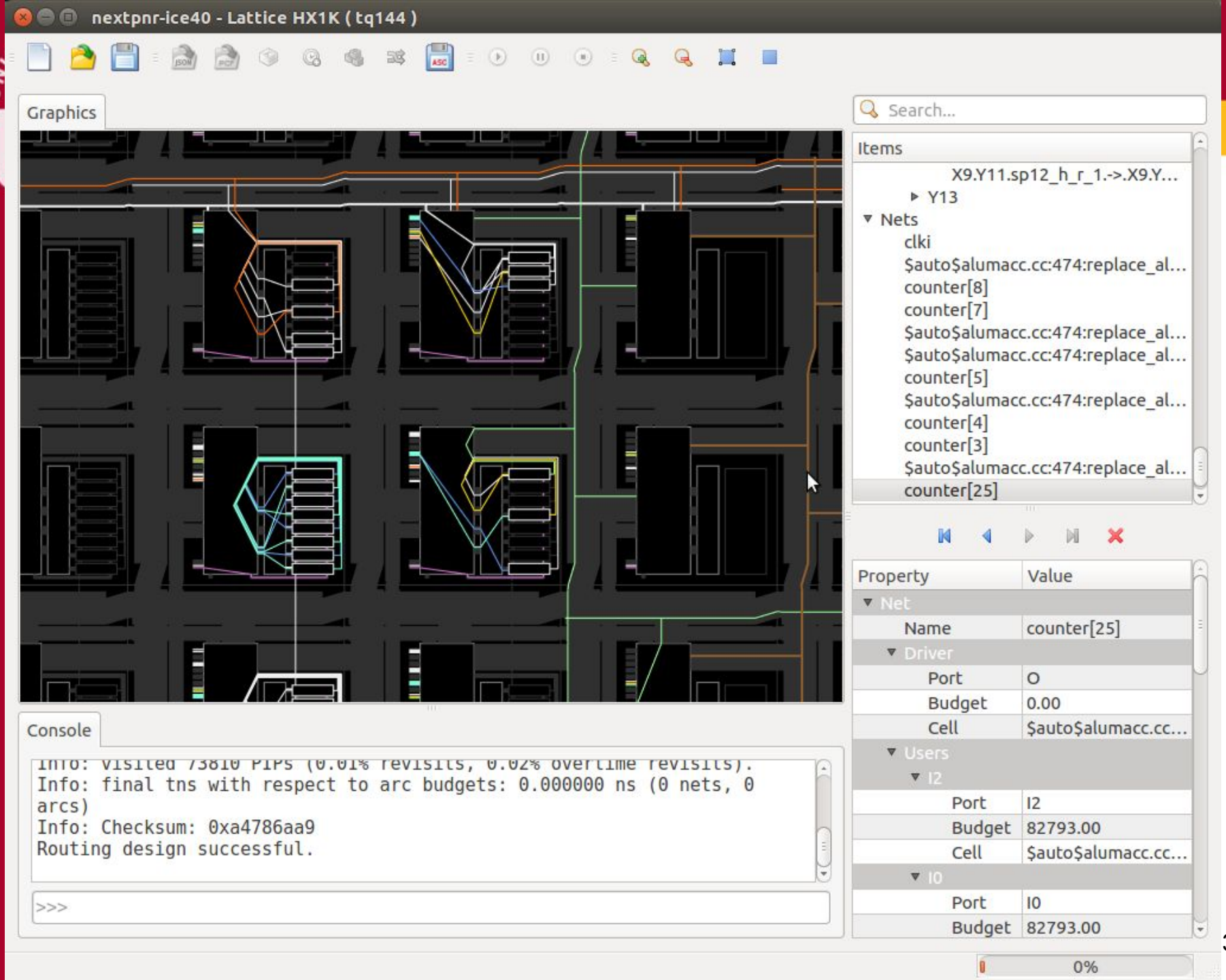



```
Number of processes:          0
Number of cells:             22
  SB_CARRY                    6
  SB_DFFR                     8
  SB_LUT4                     8
```

Nextpnr

- Nexpnr es una herramienta de Place & Route para FPGAs
  - vendor-neutral (capaz de soportar diferentes arquitecturas de FPGA)
  - timing-driven
- Licencia ISC




nextpnr-ice40 - Lattice HX1K (tq144)
Graphics
Search...
Items
X9.Y11.sp12_h_r_1.->.X9.Y...
Y13
Nets
clki
$auto$alumacc.cc:474:replace_al...
counter[8]
counter[7]
$auto$alumacc.cc:474:replace_al...
$auto$alumacc.cc:474:replace_al...
counter[5]
$auto$alumacc.cc:474:replace_al...
counter[4]
counter[3]
$auto$alumacc.cc:474:replace_al...
counter[25]
Property
Value
Net
Name
counter[25]
Driver
Port
O
Budget
0.00
Cell
$auto$alumacc.cc...
Users
I2
Port
I2
Budget
82793.00
Cell
$auto$alumacc.cc...
I0
Port
I0
Budget
82793.00
Console
Info: visited 73810 PIPs (0.01% revisits, 0.02% overtime revisits).
Info: final tns with respect to arc budgets: 0.000000 ns (0 nets, 0 arcs)
Info: Checksum: 0xa4786aa9
Routing design successful.
>>>
0%

icestorm

- Project icestorm es la **documentación** del formato de bitstream de las FPGAs iCE40 de Lattice
- Licencia ISC

icestorm

Contiene múltiples herramientas:

- **icepack**: generación del bitstream
- **iceprog**: configuración de la FPGA (descargar el bitstream a la FPGA)
- **icepll**: genera configuraciones para los PLL (Phase-Locked Loop) de la FPGA

icestorm

Y otras (que nosotros no vamos a usar):

- **icebox**: herramientas para entender el bitstream (explain, html view, diff, etc)
- **icebram**: reemplazar contenidos de una BRAM sin necesidad de volver a hacer P&R
- **icecompr**: comprimir bitstreams
- **icefuzz**: ‘fuzzing’ para hacer ingeniería inversa
- **icetime**: genera estimaciones de timing
- **icemulti**: para empaquetar varios bitstreams en una imagen multi-boot

GCC toolchain para RISC-V

- GCC = GNU Compiler Collection
- Licencia GNU GPL (General Public License)
- Ejecutables se llaman como los de gcc, pero con prefijo ‘riscv32-unknown-elf-’
  - riscv32-unknown-elf-gcc, riscv32-unknown-elf-g++, riscv32-unknown-elf-ld, riscv32-unknown-elf-gprof, etc…
  - Escribe el prefijo en el shell y pulsa ‘Tab’ para verlos todos

Flujo de diseño

- Configuración
- Implementación
- Compilación del software
- Configuración de la FPGA
- Descarga del software a través del bootloader

Flujo de diseño

También se podría no utilizar el bootloader:

- Se inicializarían las memorias internas de la FPGA con el programa
- Pero tendríamos que implementar el microprocesador cada vez que cambiáramos el programa
- Puede ser útil para desplegar la aplicación, al final del desarrollo, cuando el software no debería cambiar demasiado
- En la iCEBreaker es complicado porque las SPRAM256 no pueden inicializarse en el .bit

Flujo de diseño

Una tercera opción es almacenar el programa en memoria externa:

- Tras 8 segundos de espera, el bootloader intentará arrancar de la memoria externa
- Antes de desplegar la aplicación, podríamos modificar el bootloader para reducir ese tiempo de espera
- La mejor opción para programas más grandes

Contenido

- Arquitectura
- Buses
- Configuración
- Herramientas para implementación hardware y compilación software
- Periféricos personalizados por el usuario
- Conclusiones

Periféricos personalizados por el usuario

Principalmente dos opciones:

- Conectar nuestro periférico a un GPIO, UART, SPI o TWI
  - Son interfaces para comunicación off-chip
  - Menores prestaciones
  - Más sencillo y rápido de prototipar
- Conectar nuestro periférico a interfaces específicos para periféricos
  - Mayores prestaciones
  - Mayor dificultad

Interfaces específicos para periféricos

- Bus Wishbone
  - Disponible wrapper AXI-Wishbone
- Stream Link
  - Canales para ‘data streaming’
- Usar el subsistema de ‘Custom functions’
  - 32 registros de 32 bits en el mapa de memoria que se conectan a la lógica ‘custom’ que definamos

Contenido

- Arquitectura
- Buses
- Configuración
- Herramientas para implementación hardware y compilación software
- Periféricos personalizados por el usuario
- Conclusiones

Conclusiones

- Se pueden crear System-on-Chip complejos en FPGAs, con relativa facilidad, utilizando neorv32 u otros
- La enorme configurabilidad del sistema nos obliga a mirar despacio la documentación
- También podemos desarrollar nuestros propios periféricos
- Vayamos poco a poco e intentemos que el microprocesador arranque ;)

Referencias

- Stefan Nolting, The NEORV32 RISC-V Processor (GitHub repository)
- Stefan Nolting, The NEORV32 RISC-V Processor: Datasheet
- RISC-V Foundation, The RISC-V Instruction Set Manual Volume I: User-Level ISA. Document Version 2.2
- Opencores, Wishbone B4: WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores

Referencias (II)

- Claire Xenia Wolf, Yosys User Manual
- D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist, M. Milanović, Yosys+nextpnr: an Open Source Framework from Verilog to Bitstream for Commercial FPGAs
- Yosys git repository
- Nextpnr git repository
- Icestorm git repository