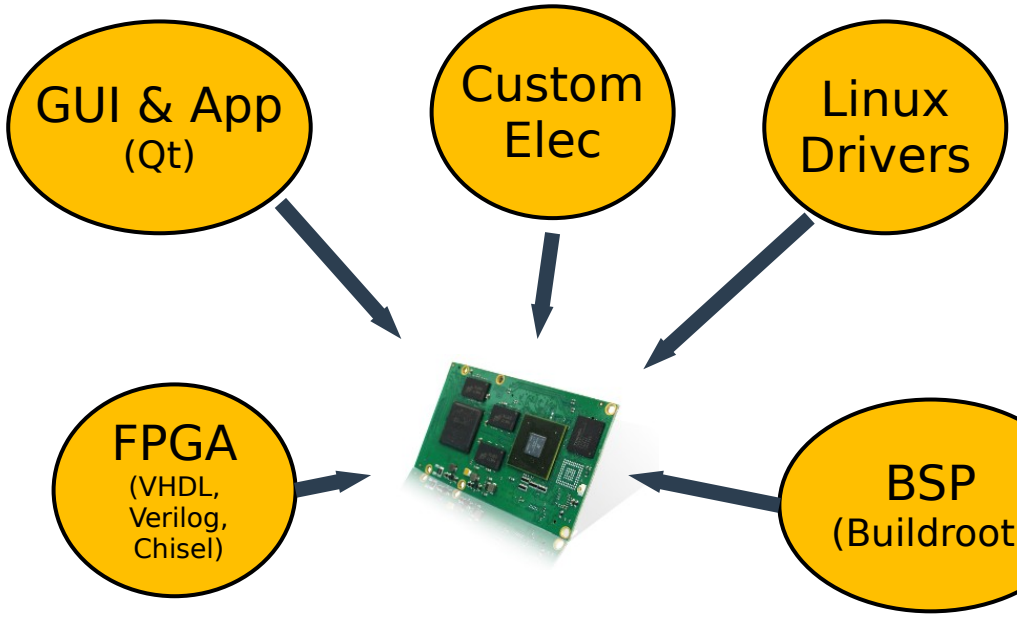
Les outils open-sources pour le FPGA

Fabien Marteau – Armadeus Systems

Introduction
Plan

Qui suis-je ?
Emploi

- **Armadeus systems**
  - http://www.armadeus.com
  - Ingénieur en Électronique
  - Responsable partie FPGA




APF27


OPOS6UL_SP

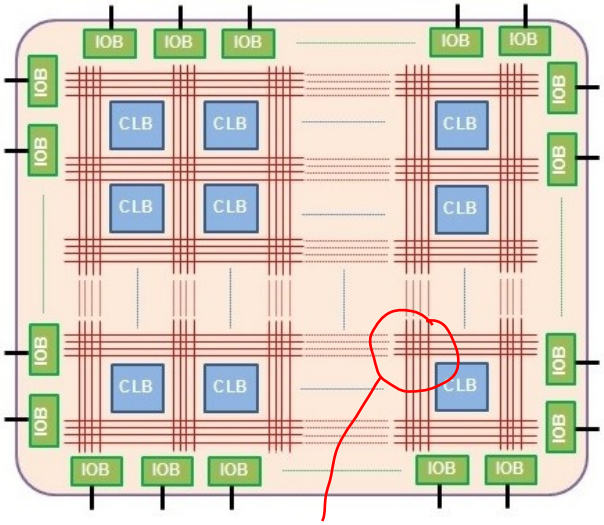

APF51


APF6_SP

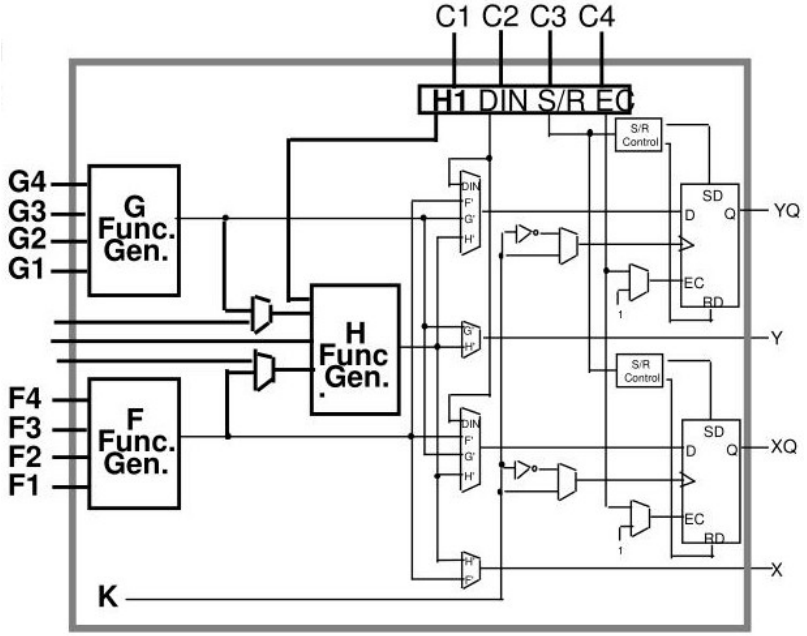
Qu’est-ce qu’un FPGA ?
Plan

Qu’est-ce qu’un FPGA ?
Architecture

- **FPGA : Field Programmable Gates Arrays .**
  - Champs de portes programmable



switchbox



CLB : Configurable Logic Bloc
(XC4000 xilinx)

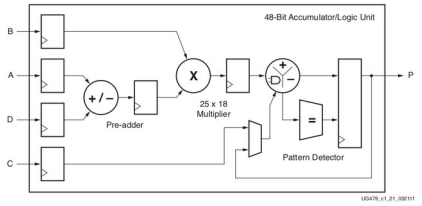
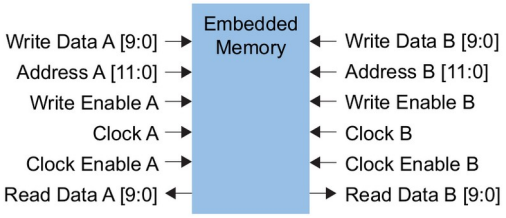
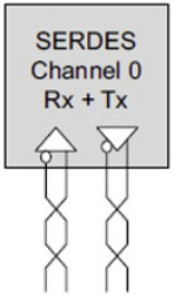


Figure 1-1: **Basic DSP48E1 Slice Functionality**

Multiplieur

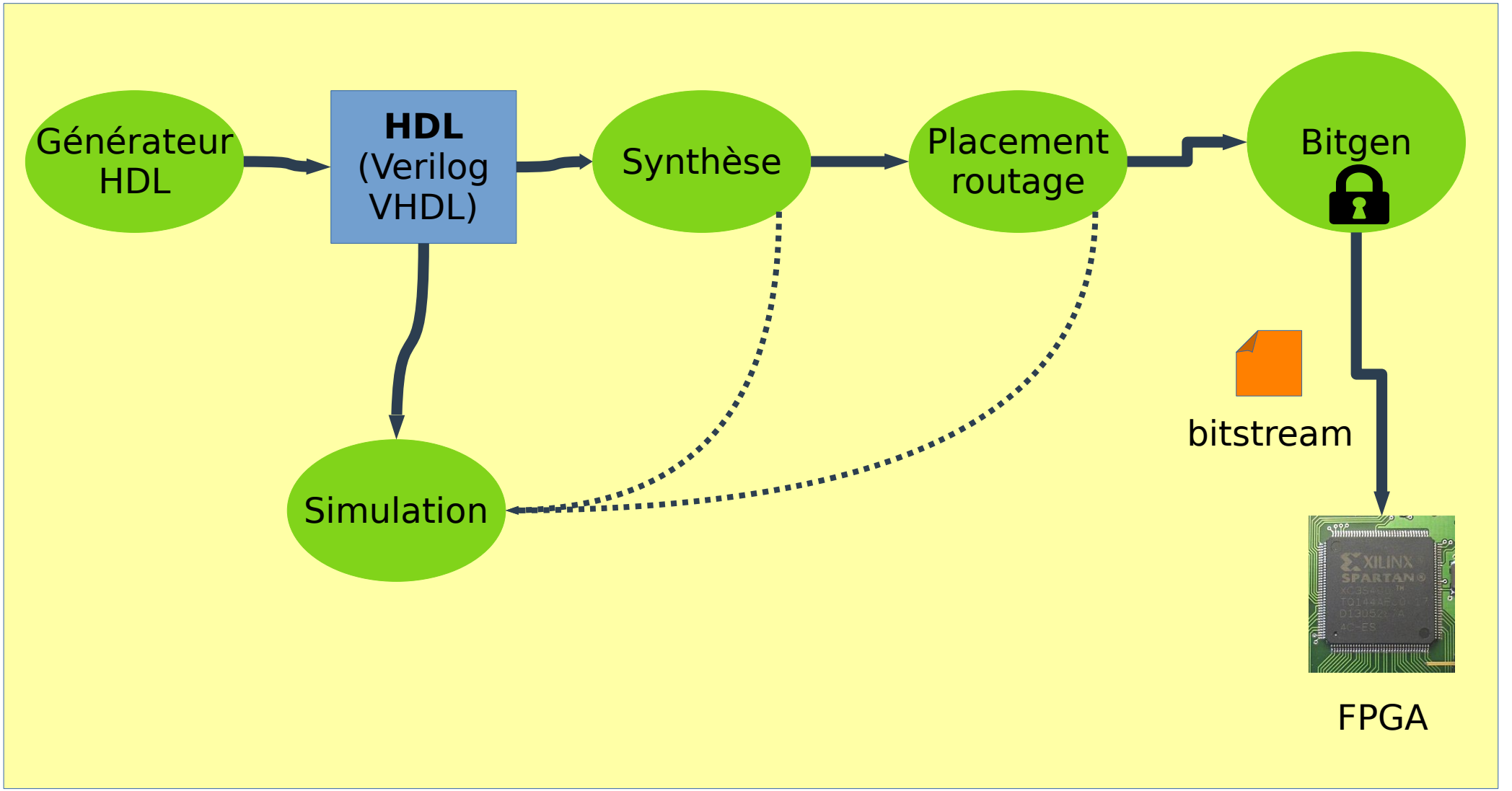


Bram



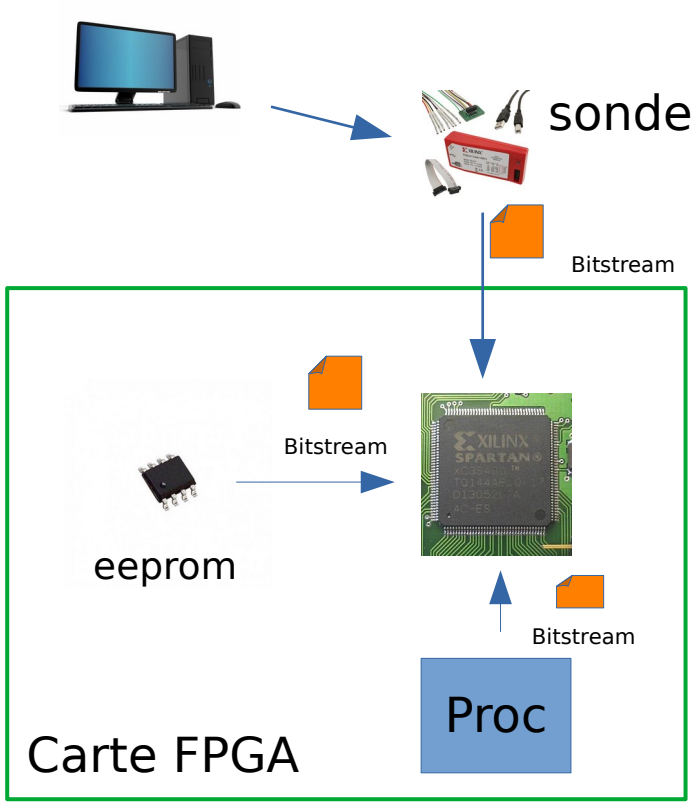
Qu'est-ce qu'un FPGA ?

La chaîne de développement FPGA



Qu'est-ce qu'un FPGA ?
La configuration : le bitstream

- **Le drame du Bitstream**
  - En RAM → Transferé à chaque mise en route
  - Facile à copier/relire
  - Format gardé secret par tous les constructeurs
    - Exception de **QuickLogic** : **EOS S3**
  - Protection/sécurité par l'obscurité
    - «From the bitstream to the netlist» – Jean Baptiste Note & Éric Rannaud (2008)



Les langages de simulation synthétisable
Synthèse

Synthèse
VHDL – exemple

- **VHSIC Hardware Description Language**
  - Very High Speed Integrated Circuit
- **IEEE 1076-1987, *1993*, 2002, *2008*, 2019**
- **3 simulateurs libre :**
  - GHDL (bientôt IEEE 1076-2008)
    - http://ghdl.free.fr
  - NVC (IEEE 1076-2002)
    - https://github.com/nickg/nvc
  - FreeHDL (IEEE 1076-1993)
    - https://sourceforge.net/projects/qucs/

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity signed_adder is
6    port
7    (
8      aclr : in   std_logic;
9      clk  : in   std_logic;
10     a    : in   std_logic_vector;
11     b    : in   std_logic_vector;
12     q    : out  std_logic_vector
13   );
14 end signed_adder;
15
16 architecture signed_adder_arch of signed_adder is
17   signal q_s : signed(a'high+1 downto 0); -- extra bit wide
18
19 begin -- architecture
20   assert(a'length >= b'length)
21     report "Port A must be the longer vector if different sizes!
22     severity FAILURE;
23   q <= std_logic_vector(q_s);
24
25   adding_proc:
26   process (aclr, clk)
27     begin
28       if (aclr = '1') then
29         q_s <= (others => '0');
30       elsif rising_edge(clk) then
31         q_s <= ('0'&signed(a)) + ('0'&signed(b));
32       end if; -- clk'd
33     end process;
34
35 end signed_adder_arch;
```

https://fr.wikipedia.org/wiki/VHDL

Synthèse
Verilog – Exemple

- **Cadence Design Systems**
  - IEEE 1364-1995, 2001, 2005
- **SystemVerilog (Accellera):**
  - IEEE 1800-2005, 2009, 2012
- **Simulateurs libre**
  - Icarus Verilog

    http://iverilog.icarus.com/
  - Verilator

    https://www.veripool.org/verilator/
  - CVC – Tachyon

    http://www.tachyon-da.com/
  - VeriWell, cver : abandonnés ?

```
module pwmgen #(
`ifdef FORMAL
    parameter BPM_MAX = 20
`else
    parameter BPM_MAX = 250
`endif
)(
    /* clock and reset */
    input clk_i,
    input rst_i,
    /* timepulse */
    input tp_i,
    /* input value */
    input [($clog2(BPM_MAX+1)-1):0] bpm_i,
    input bpm_valid,
    /* output */
    output pwm_o);

reg [($clog2(BPM_MAX+1)-1):0] bpm_reg;
reg [($clog2(BPM_MAX+1)-1):0] pwmthreshold;
reg [($clog2(BPM_MAX+1)-1):0] count;
```

```
/* Latching bpm_i on bpm_valid */
always@(posedge clk_i or posedge rst_i)
begin
    if(rst_i)
    begin
        bpm_reg <= 0;
        pwmthreshold <= 0;
    end else begin
        if(bpm_valid)
            bpm_reg <= bpm_i;
        if(count == BPM_MAX)
            pwmthreshold <= bpm_reg;
    end
end

/* count */
always@(posedge clk_i or posedge rst_i)
begin
    if(rst_i)
        count <= BPM_MAX;
    else begin
        if(tp_i)
        begin
            if (count == 0)
                count <= BPM_MAX;
            else
                count <= count - 1'b1;
        end
    end
end

assign pwm_o = (count <= pwmthreshold);
```

Les langages de simulation synthétisable
Générateurs HDL

1) Introduction

2) Qu'est-ce qu'un FPGA ?

3) **Les langages de simulation et synthèse**

   1) Synthèse

   2) Générateurs HDL

      1) Chisel

      2) Migen/Litex

      3) Clash, SpinalHDL, MyHDL, bluespec

4) Synthèse, Placement routage et Bitstream

5) Conclusion

Générateur HDL
Chisel

- **C**onstructing **H**ardware **i**n **S**cala **E**mbedded **L**anguage
- **Scala**
  - DSL : Domain Specific Language
- **Créé en 2010 par l'équipe RISC-V (Berkeley)**
- **Language Synchrone**
  - Horloge et reset implicites
- **Chisel → FIRRTL → Verilog**

Générateur HDL
Chisel

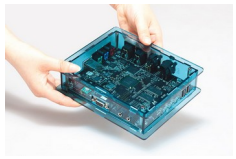
https://github.com/Martoni/GbVga

```scala
package gbvga

import chisel3._
import chisel3.util._
import chisel3.stage.{ChiselGeneratorAnnotation, ChiselStage}

class GbVga extends Module with GbConst {
  val io = IO(new Bundle {
    /* Game boy input signals */
    val gb = Input(new Gb())
    /* VGA output signals */
    val vga_hsync = Output(Bool())
    val vga_vsync = Output(Bool())
    val vga_color = Output(new VgaColors())
  })

  /* GameBoy write */
  val gbwrite = Module(new GbWrite(2, debug_simu=false, aformal=false))
  gbwrite.io.gb := io.gb

  /* Mem Vga */
//  val memvga = Module(new MemVga())
  val memvga = Module(new MemVgaZoom())
  io.vga_hsync := memvga.io.vga_hsync
  io.vga_vsync := memvga.io.vga_vsync
  io.vga_color <> memvga.io.vga_color

  /* dual port ram connection */
  val mem = Mem(GBWIDTH*GBHEIGHT, UInt(2.W))
  when(gbwrite.io.Mwrite) {
    mem(gbwrite.io.Maddr) := gbwrite.io.Mdata
  }
  val last_read_value = RegInit(0.U(2.W))
  when(memvga.io.mem_read) {
    memvga.io.mem_data := RegNext(mem(memvga.io.mem_addr))
    last_read_value := memvga.io.mem_data
  }.otherwise {
    memvga.io.mem_data := last_read_value
  }

}

object GbVgaDriver extends App {
  (new ChiselStage).execute(args,
    Seq(ChiselGeneratorAnnotation(() => new GbVga())))
}
```

- RISCV : RocketChip
- SiFive : E310, U540, ...
- Google : Edge TPU

Générateur HDL
Migen

- **Python**
  - FHDL – Fragmented Hardware Description Language
- **Langage synchrone**
- **«Piles inclues» :**
  - Pilote les IDE pour la synthèse
  - Simulation
- **Générateur Verilog**
- **M-Lab**
  - Milkymist
  - ARTIQ



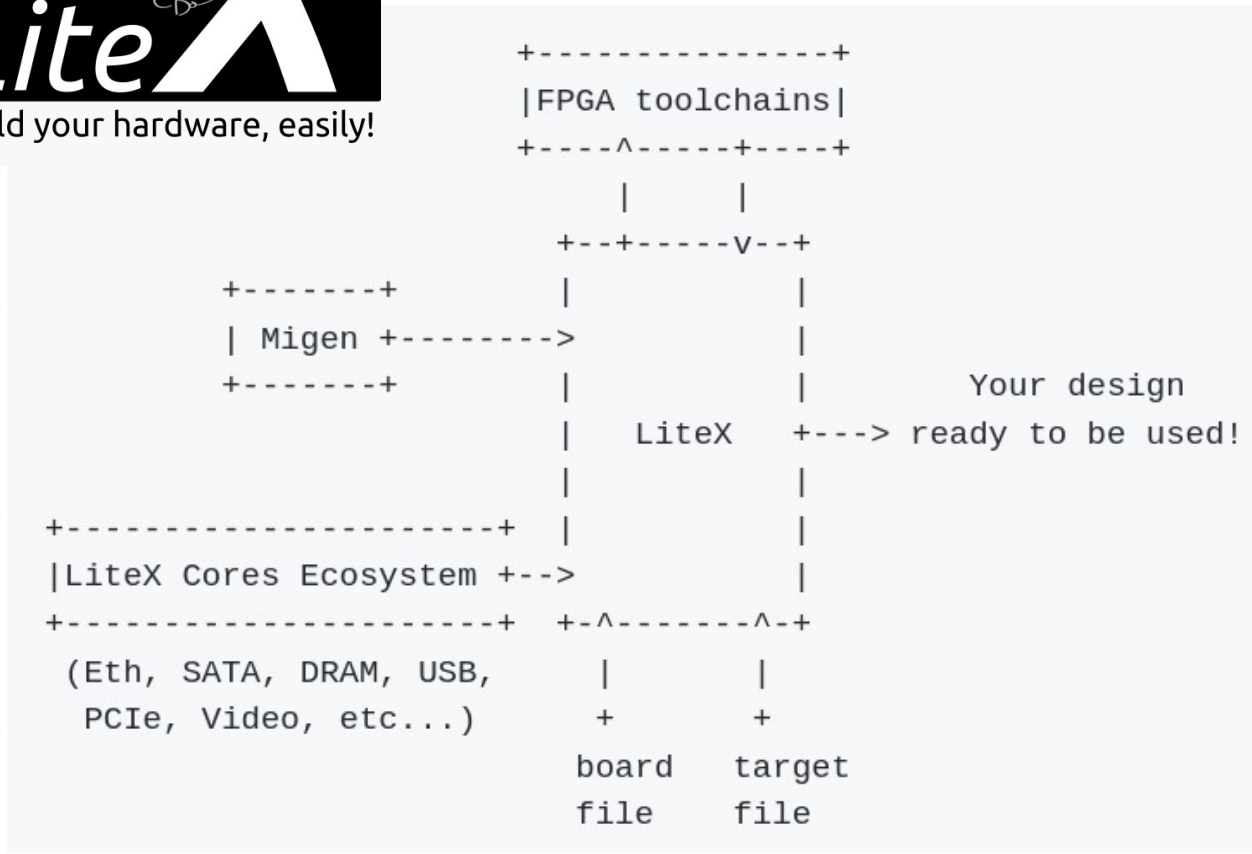

```
1  from migen import  *
2  from migen.fhdl import verilog
3
4  m = Module()
5  counter = Signal(24)
6  led1 = Signal()
7
8  m.comb += led1.eq(counter[23])
9  m.sync += counter.eq(counter+1)
10
11 print(verilog.convert(m, ios={led1}))
```

Générateur HDL
Litex

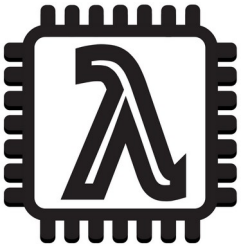
- Basé sur Migen
- «Linux» du FPGA

Build your hardware, easily!

Name	Build Status	Description
LiteX-Boards	ci passing	Boards support
LiteDRAM	ci failing	DRAM
LiteEth	ci passing	Ethernet
LitePCIe	ci passing	PCIe
LiteSATA	ci passing	SATA
LiteSDCard	ci passing	SD card
LiteICLink	ci passing	Inter-Chip communication
LiteJESD204B	ci passing	JESD204B
LiteVideo		VGA, DVI, HDMI
LiteScope	ci passing	Logic analyzer

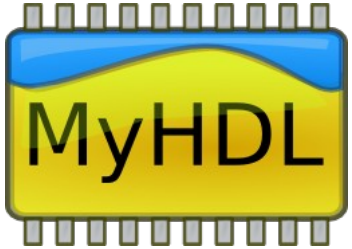


Générateur HDL autres



- **Clash**
  - Basé sur **Haskell**
  - QBayLogic

- **SpinalHDL**
  - Basé sur Scala, base du **VexRiscv**



- **MyHDL**
  - Basé sur Python
  - Langage à événements comme VHDL/Verilog
- **Bluespec**
  - Basé sur Haskell et/ou SystemVerilog
  - Utilisé pour les processeurs : Flute, Shakti, Piccolo
  - Récemment libéré

Les langages de simulation synthétisable
Générateurs HDL

1) Introduction

2) Qu'est-ce qu'un FPGA ?

3) Les langages de simulation et synthèse

4) Synthèse, Placement routage et Bitstream

- 1) Synthèse : Yosys
- 2) Placement routage
- 3) Bitstream

5) Conclusion

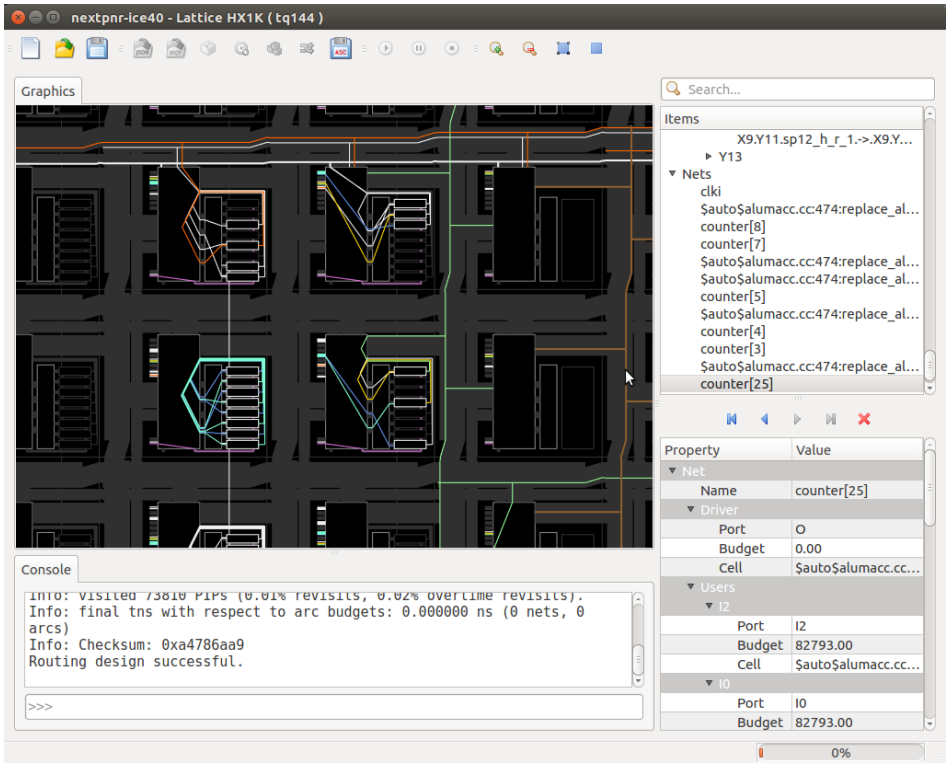
Synthèse
Yosys

- **Yosys Open SYnthesis Suite**
  - http://www.clifford.at/yosys/
- **Verilog-2005**
- **VHDL possible avec ghdl-yosys-plugin**
  - https://github.com/ghdl/ghdl-yosys-plugin
- **Preuve formelle yosys-smtbmc**
- **Grand nombre de formats de sortie :**
  - BLIF / EDIF/ BTOR / SMT-LIB / simple RTL Verilog / etc
- **Le «GCC» du FPGA**

Synthèse, Placement routage et Bitstream
Placement routage

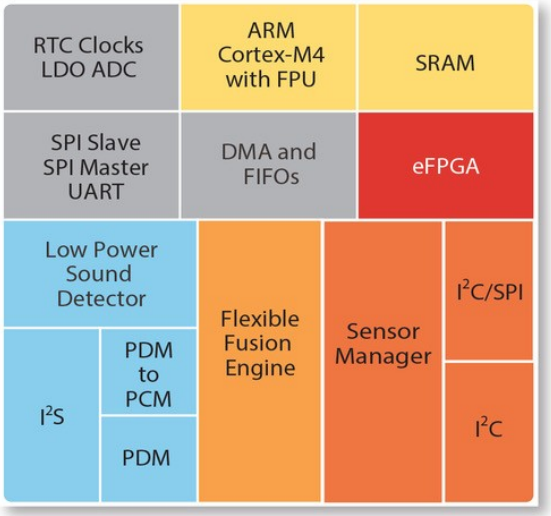
Placement routage
NextPnR

- https://github.com/YosysHQ/nextpnr
- **Héritier d'Arachne-pnr** https://github.com/YosysHQ/arachne-pnr
- **Nombreux FPGA**
  - Lattice iCE40
  - Lattice ECP5
  - Lattice Nexus
  - Gowin LittleBee
  - Intel CycloneV (beta)
  - Xilinx Série 7 (Work in progress)



Placement routage
VPR

- **Versatile Place and Route**
- **Fait parti de VTR (Verilog To Routing)**
  - https://github.com/verilog-to-routing/vtr-verilog-to-routing
  - Utilisé dans **QORC** pour la **QuickFeather**
    - QuickLogic EOS S3



EOS S3

Synthèse, Placement routage et Bitstream
Placement routage

1) Introduction

2) Qu'est-ce qu'un FPGA ?

3) Les langages de simulation et synthèse

4) Synthèse, Placement routage et Bitstream

1) Synthèse

2) Placement routage

3) Bitstream

5) Conclusion

Bitstream
Ingénierie inverse

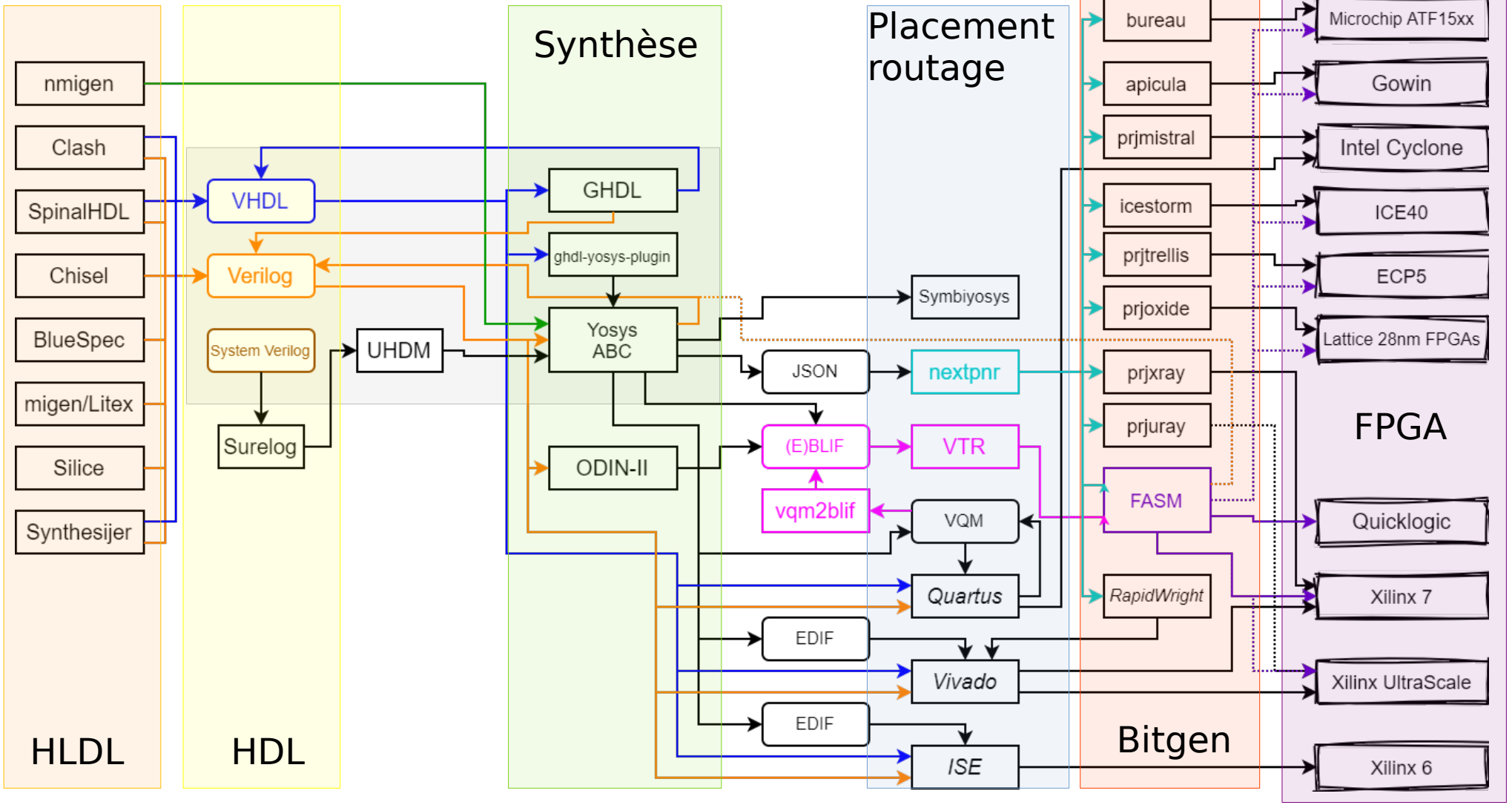
- **Debit (obsolète 2008) :**
  - Virtex 2 (xc2v2000)
  - *From the bitstream to the netlist* : Jean-Baptiste Note, Éric Rannaud
- **Icestorm (2015): http://www.clifford.at/icestorm/**
  - Lattice iCE40
- **Apicula : https://github.com/YosysHQ/apicula**
  - Gowin LittleBee
- **X-Ray : https://github.com/SymbiFlow/prjxray**
  - Xilinx serie7
- **Trellis : https://github.com/SymbiFlow/prjtrellis**
  - Lattice ECP5
- **QuickLogic-FASM : https://github.com/QuickLogic-Corp/quicklogic-fasm**
  - Standard Ouvert !
  - QuickLogic EOS S3

Conclusion

- **Il n'y a pas que le VHDL**
- **Le développement sur FPGA**
  - Possible en open source
  - En pleine progression
- **La partie Simulation**
  - Mature
  - Peu connue dans le monde industriel
- **Il reste du chemin à parcourir pour**
  - Placement&routage
  - bitstream
- **Révolution en cours dans les ASIC**
  - SKY130, OpenLane, OpenSTA, ...
  - Caravel

La chaîne de développement FPGA

Open Source



https://github.com/hdl/awesome

Synthèse
ABC, OdinII, Alliance

- **ABC :**
  - Optimisation logique
  - Universitaire (Berkeley)
  - Format AIGER et/ou BLIF
- **OdinII :**
  - Inclu dans VTR (Verilog To Routing)
  - Synthèse Verilog vers BLIF (Berkeley Logic Interchange Format)
  - Universitaire (Berkeley)
- **Alliance/coriollis :**
  - Universitaire (Université Pierre et Marie Curie – Paris)
  - VHDL
  - ASIC

Les langages de simulation synthétisable
Générateurs HDL

Simulation
SystemC

- Standard Accellera
- Classe C++
- Pas de synthèse libre
- «Simulation» sous forme d'exécutable compilé avec GCC
- Utilisable avec Verilator


```
#include "systemc.h"

SC_MODULE(counter)
{
  sc_in<bool> clk;
  sc_in<bool> nrst;
  sc_out<sc_uint<8> > q;
  void count()
  {
    if(nrst == false)
      q = 0;
    else
      q = q + 1;
  };
  SC_CTOR(and3)
  {
    SC_METHOD(count);
    sensitive << clk.pos();
  }
}
```

https://fr.wikipedia.org/wiki/SystemC

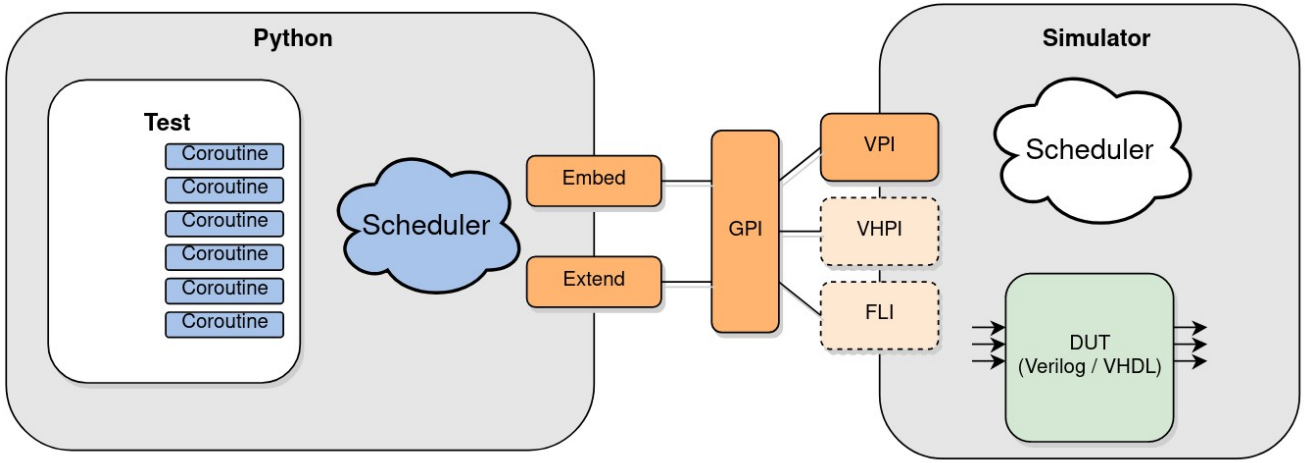
Simulation
CocoTB

- **COsimulation COroutine TestBench**
- **Cosimulation**
  - Utilisation du simulateur habituel
  - Verilog, VHDL, ...
- **Coroutines**
  - Librairie d'extensions
    - Cocotbext-wishbone
    - Cocotbext-spi
    - Cocotbext-uart
    - Avalon
    - cocotb_usb

```python
@cocotb.test()
async def one_frame(dut):
    fname = "one_frame"
    tgw = TestGbWrite(dut)
    tgw.display_config()
    await Timer(1)
    tgw.log.info("Running test {}".format(fname))
    await tgw.reset()
```

```python
async def reset(self):
    self.rst <= 1
    await Timer(100, units="ns")
    self.rst <= 0
    self._mem_writer = cocotb.fork(self.memory_writer())
```

```python
pixelcount = dut.io_countcol.value.integer
```



Les langages de simulation synthétisable
Générateurs HDL

1) Introduction

2) Qu'est-ce qu'un FPGA ?

3) Les langages de simulation et synthèse

4) Synthèse, Placement routage et Bitstream

5) Outils de construction (build)

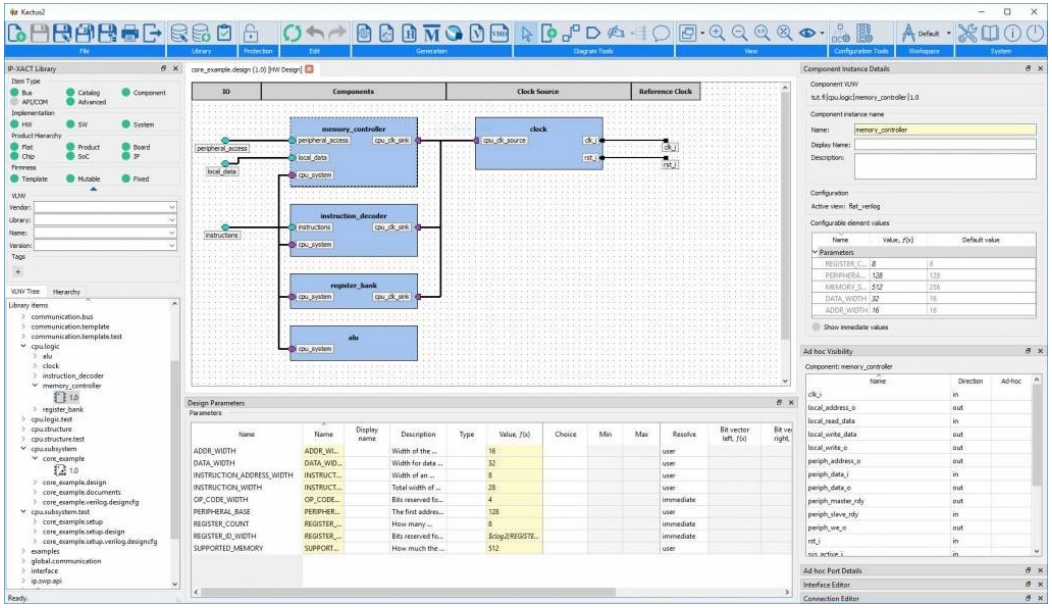
1) IP-XACT, CactusII, POD

2) Edalize, FuseSoc

6) Conclusion

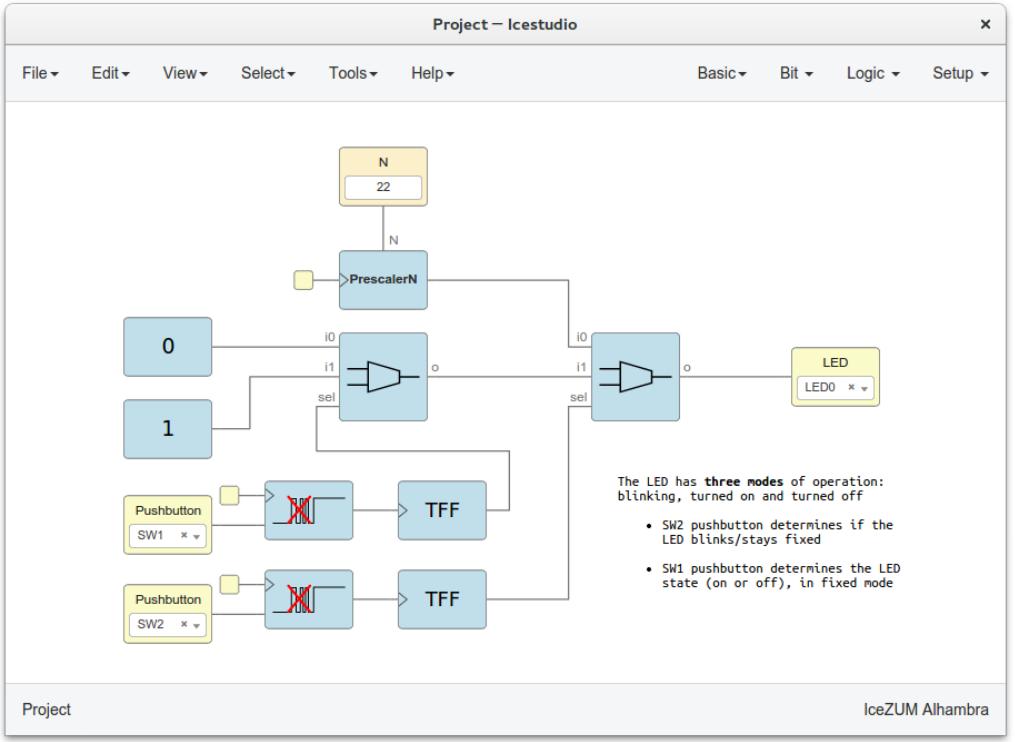
Outils de construction
standards ?

- **IP-XACT :**
  - Standard de description de composants
  - XML – IEEE 1685-2009
  - Accellera
- **KactusII :**
  - IDE graphique Qt
  - Basé sur IP-XACT
  - Université de tampere (Finlande)
- **POD :**
  - Peripheral On Demand
  - Outils python «maison» Armadeus



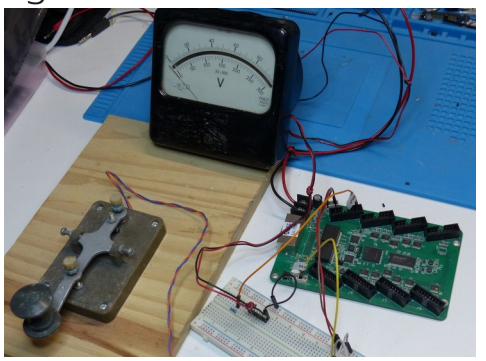
Outils de construction
Edalize et Fusesoc

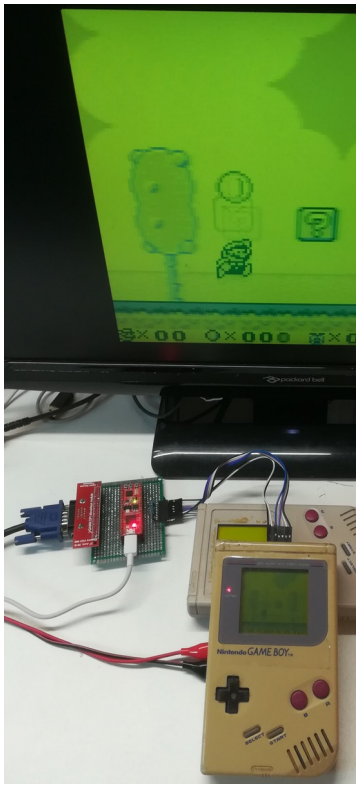
- **Edalize**
  - Module Python pour piloter tous les IDE
    - ISE, Quartus, NextPnR, Modelsim, Verilator, yosys, xsim, VCS, OpenLane, Libero, Vivado, ...
- **FuseSoC**
  - Description «concurrente» de IP-XACT
  - YAML
- **APIO**
  - Pilotage des écosystèmes Open Source
  - ECP5, iCE40
- **IceStudio**
  - Interface graphique



Qui suis-je ?
Activité open-source

- Mainteneur de modules Chisel et Cocotb
  - Cocotb :
    - **Cocotbext-spi** : https://github.com/Martoni/cocotbext-spi
    - **Cocotbext-imxeim** : https://github.com/Martoni/cocotbext-imxeim
  - Chisel :
    - **WbPlumbing** : https://github.com/Martoni/WbPlumbing
    - **WbGPIO** : https://github.com/Martoni/wbGPIO
    - **MDIO** : https://github.com/Martoni/mdio
    - **Spi2Wb** : https://github.com/Martoni/spi2wb
  - Bricolages autour des FPGA
    - **GbVGA** : https://github.com/Martoni/GbVga
    - **TapTempo (LinuxFR)** : https://github.com/Martoni/TapTempoASIC



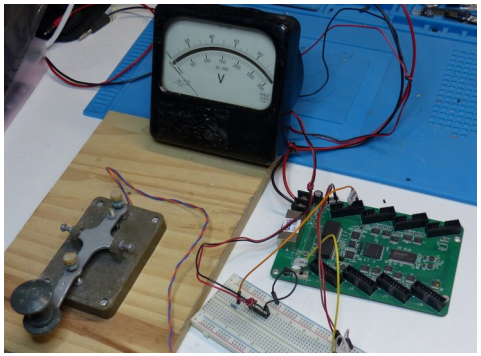


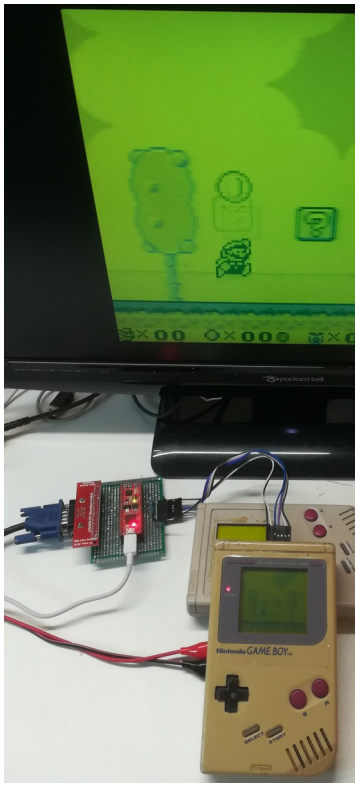
Qui suis-je ?
Publications

- Blog «Front de libération des FPGA»
  - https://www.fabienm.eu/flf/

- Écriture d'articles aux éditions Diamonds :
  - OpenSilicium, Linux Magazine, Hackable

- Bricolage autour des FPGA
  - TapTempo
  - GbVGA



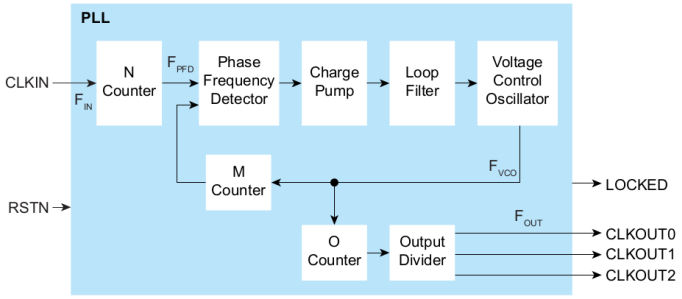


Qu’est-ce qu’un FPGA ?

Briques de bases

- **Différents types de cellules logique «hard»**
  - Multiplieurs/DSP
  - IO
  - SerDes
  - PLL
  - BRam

Figure 8: T8 PLL Block Diagram



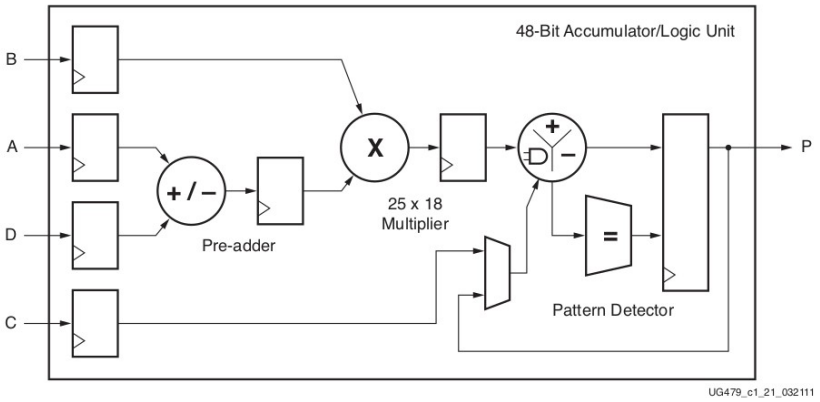
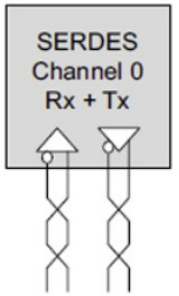


Figure 1-1: **Basic DSP48E1 Slice Functionality**



Simulation
Librairies UVM, UVVM

- **UVM : Universal Verification Methodology**
  - SystemVerilog
  - Accelera
  - Encore difficile à simuler avec des logiciels libres
- **UVVM : Universal VHDL Verification Methodology**
  - VHDL
  - Projet open source basé sur UVM