

Systeme d'acquisition PXIe pour le telescope de faisceau d'EUDET: vers une programmation LabView des FPGA ?



Cayetano Santos

cayetano.santos@iphc.cnrs.fr

Jean Sébastien Pelle

jean-sebastien.pelle@IReS.in2p3.fr

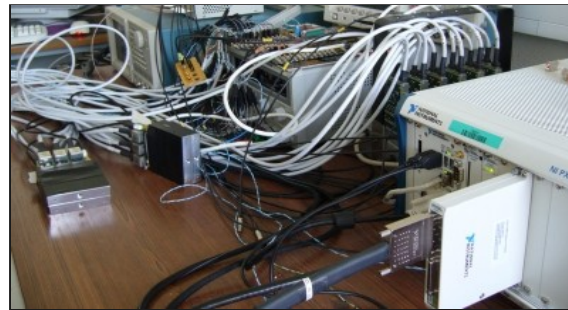
Gilles Claus

gilles.claus@ires.in2p3.fr

Plan de l'exposée



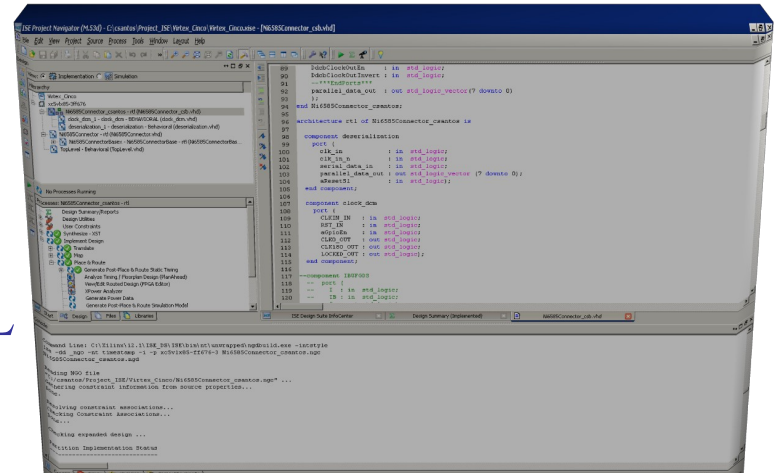
1. Motivation du projet



2. Fonctionnalités de base de LabVIEW / FPGA



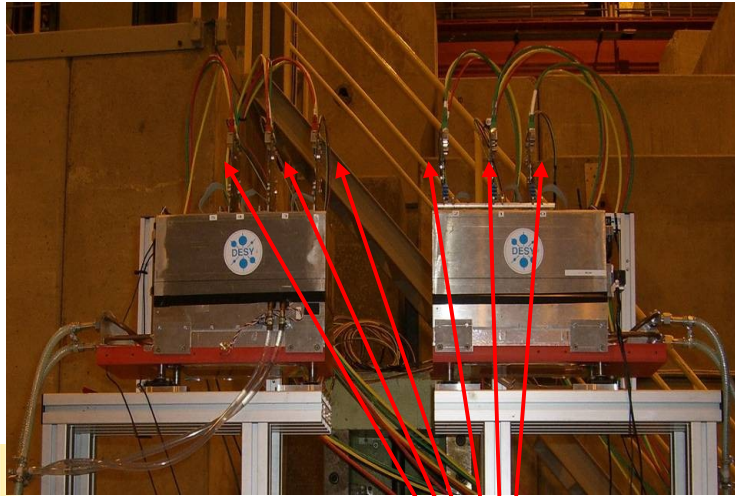
3. Flow de programmation standard Xilinx / ISE en VHDL (utilisateur expérimenté)



The project: DAQ for EUDET JRA1 Beam Telescope



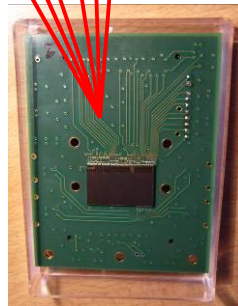
Beam Telescope with 6 planes of Mimosa 26 sensor



Mimosa 26

- 576 lines x 1152 columns
- Pixel pitch 18,4 μm
- Columns // readout
- One discriminator / column
- Integrated zero suppression logic
- Integration time 115,2 μs $\leftarrow \rightarrow$ 8680 frames / s

Module with one Mimosa 26



EUDET JRA1 Beam Telescope

EUDET – European project FP6

- ▶ Detector R&D toward ILC
- ▶ Beam Telescope \rightarrow 6 planes of Mimosa 26
 - ▶ Mimosa 26 = 576 lines x 1152 columns – Pixel pitch 18,4 μm
 - ▶ On chip
 - ▶ Discriminators \rightarrow Digital output
 - ▶ Zero suppression logic \rightarrow Data flow reduction
 - ▶ Telescope resolution $\sim 2 \mu\text{m}$ – Integration time 115,2 μs
 - ▶ 8680 frames / s \rightarrow 8680 events / s

▶ Requirements from DAQ point of view

- ▶ Data stream 20 MB/s / Plane
 - \rightarrow 120 MB/s for the Telescope
- ▶ Zero dead time readout
- ▶ Two serial data links @ 80 MHz / Plane
 - \rightarrow 12 serial links @ 80 MHz for the Telescope

The DAQ: Acquisition board development or try to use COTS



INFN DAQ : 6 x EUDRB Board in VME crate

For Telescope demonstrator --> EUDRB board

- ▶ Developed by collaboration for analogue Telescope (Mimotel)
 - ▶ VME – 4 inputs 12 bits @ 20 MHz – Zero suppression
 - ▶ Upgrade of board for digital Telescope (Mimosa 26)
 - ▶ → 2 digital links 80MHz / board
 - ▶ Requires 6 EUDRB boards + 2 VME crates
 - ▶ Well suited for telescope demonstrator
- ▶ Not the best solution to **duplicate & distribute** telescope DAQ
 - ▶ **Boards production & testing – Repair boards**
 - ▶ **Zero dead time acquisition difficult to reach**



IPHC-NI DAQ : 1 x Flex RIO board in PXI Express crate

For Telescope copies

- ▶ Development done by IPHC in **collaboration with NI**
 - ▶ Based on NI Flex RIO board
 - ▶ Requires **one single board** in a PXIe crate
 - ▶ **Main advantages**
 - ▶ Focus on FW & SW development → Few HW tasks
 - ▶ **No board production & testing**
 - ▶ **No need to repair boards → Done by NI**
 - ▶ System cost ↔ EUDRB system ... may be less



The DAQ: Proposal based on FlexRIO

6 x Mimosa 26

Clock out 80 MHz

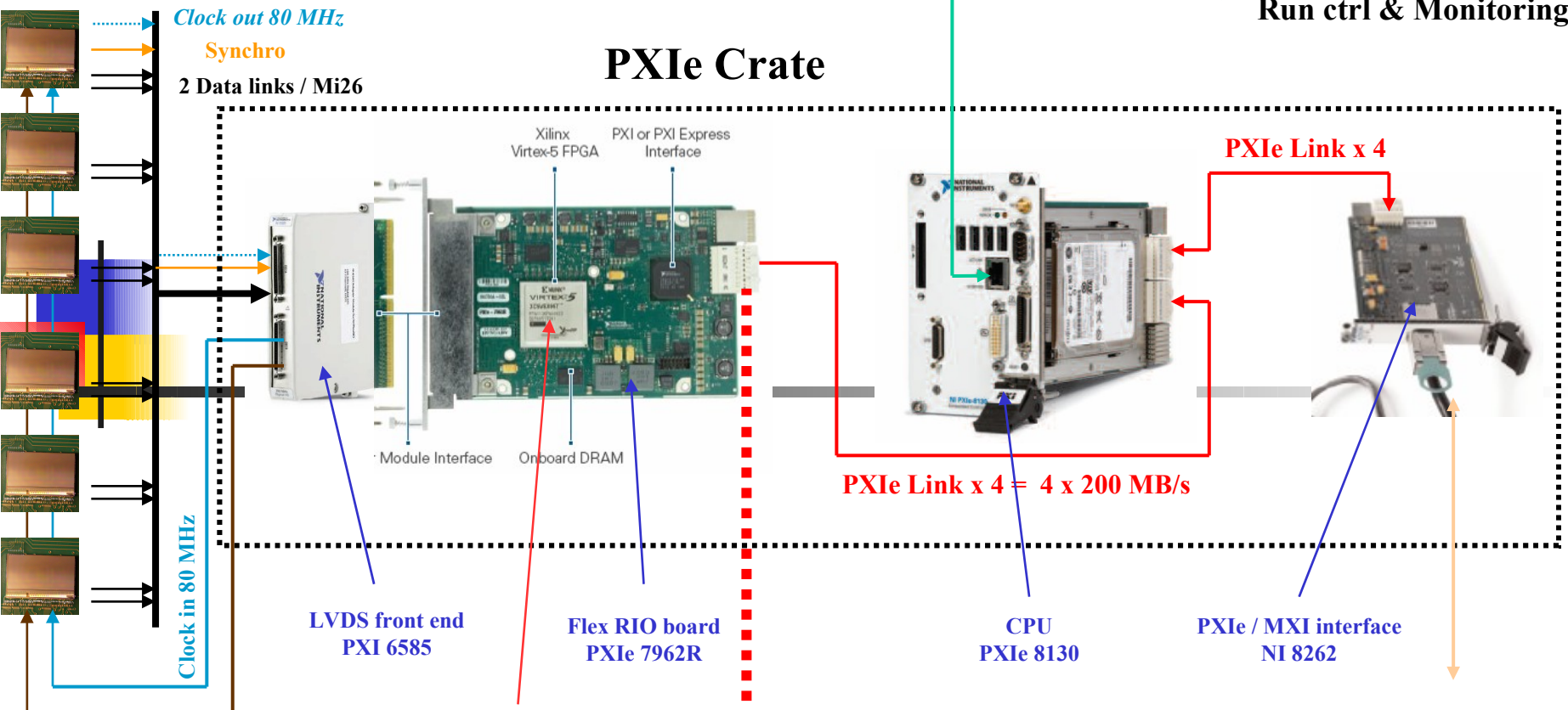
Synchro

2 Data links / Mi26

PXie Crate

Ethernet link
1 Gb/s → ~ 100 MB/s

EUDET JRA1 SW
Run ctrl & Monitoring



LVDS front end
PXI 6585

Flex RIO board
PXIe 7962R

CPU
PXIe 8130

PXIe / MXI interface
NI 8262

PXIe Link x 4 = 4 x 200 MB/s

PXIe Link x 4

MXI Link ~ 600 MB/s

Firmware (désérialisation)

Data bandwidth from board to disk
→ we need 120 MB/s



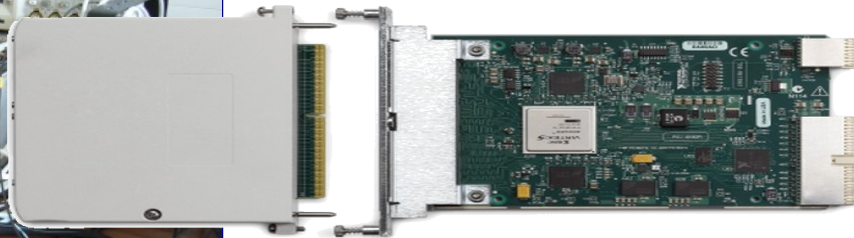
RAID Disk - 3 TB
HDD 8264



NI FlexRIO System Architecture

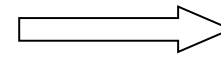


NI FlexRIO Adapter Module



NI FlexRIO FPGA Module

PXI



Observé 120 MB/s. (PXI) - 750 MB/s. (PXIe)



PXI Platform



- DSP-focused Virtex-5 SX50T FPGAm, direct access to FPGA I/O resources
- 594 KB embedded block RAM
- 66 differential pairs or 132 single ended
- Two global clock inputs
- 16 DMA channels for high-speed data streaming at more than 800 MB/s
- Peer-to-peer data streaming to and from other FPGA modules and select NI modular instruments
- Adapter module is required for I/O



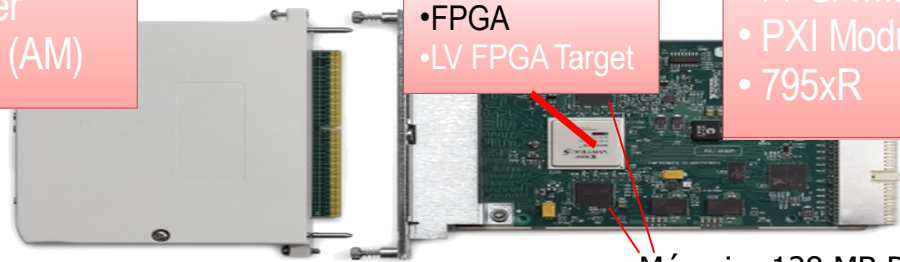
FPGA LX85 Virtex-5
DRAM DDR2 128 Mo



- IO Module
- Adapter Module (AM)

- FPGA
- LV FPGA Target

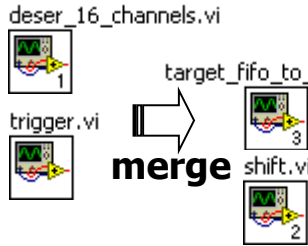
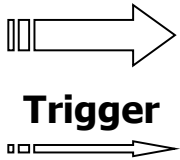
- FPGA Module
- PXI Module
- 795xR



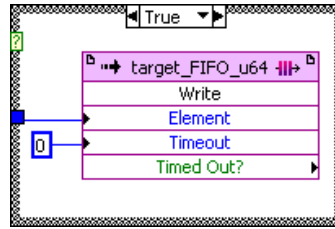
Mémoire 128 MB DRAM DDR2

Code Embarqué (désérialisation)

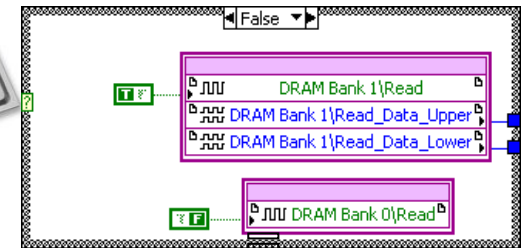
Données serie (2 links at 80 MHz.)



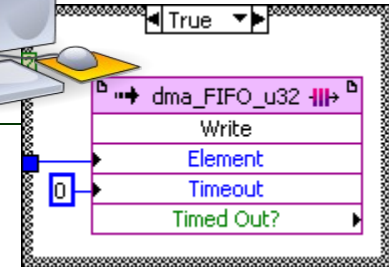
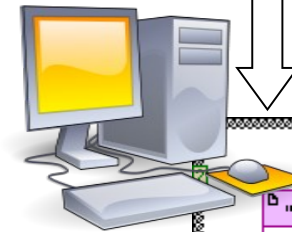
Vers FIFO interne



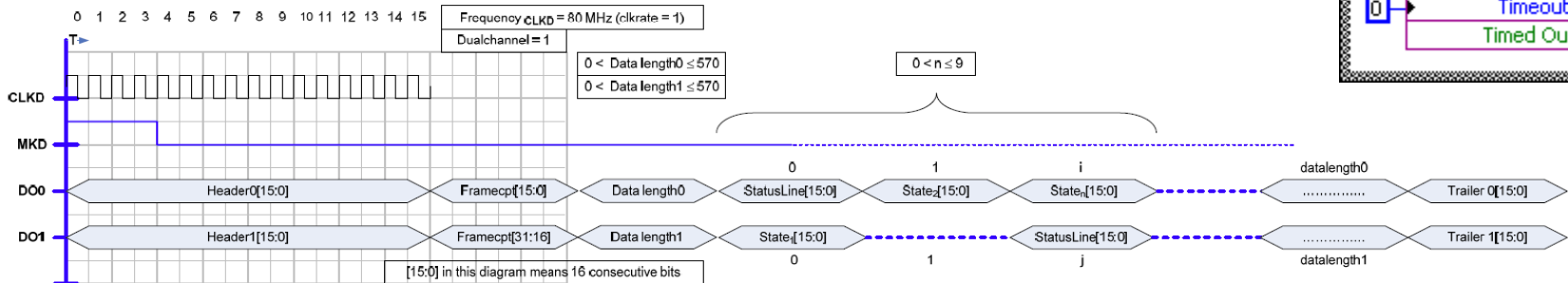
vers DRAM



vers PC (DMA)

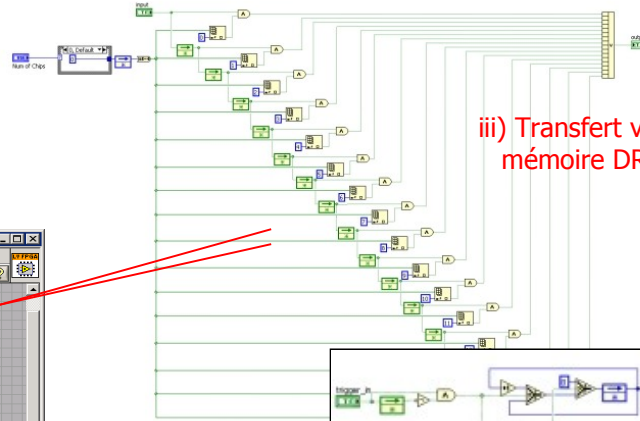
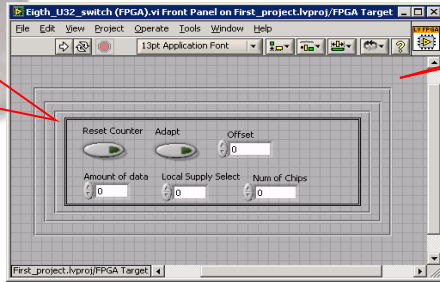


Maximum data stream length is 1140 W16 (word of 16 bits) – 570 W16 on each link (D00, D01)



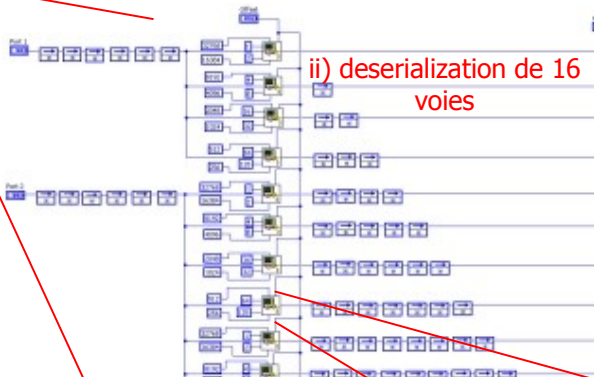


FPGA vi

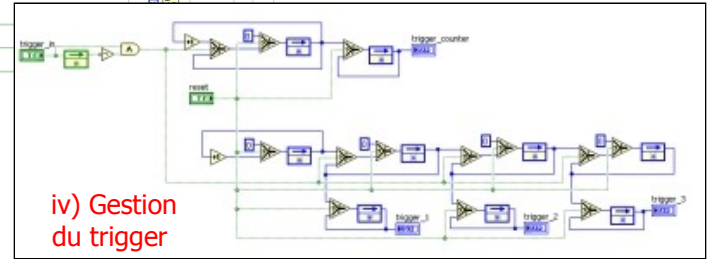


iii) Transfert vers la mémoire DRAM

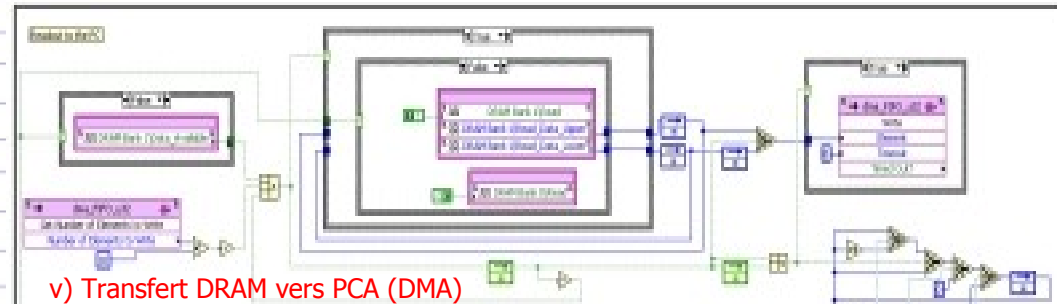
deser_16_channels.vi



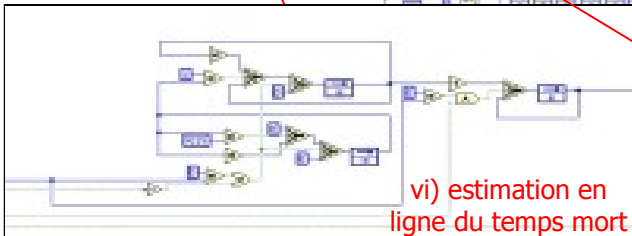
ii) deserialisation de 16 voies



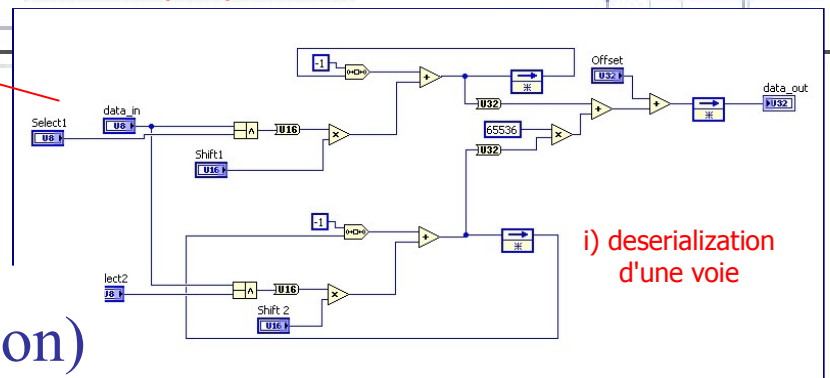
iv) Gestion du trigger



v) Transfert DRAM vers PCA (DMA)



vi) estimation en ligne du temps mort



i) deserialisation d'une voie

Code Embarqué (ii) (désérialisation)

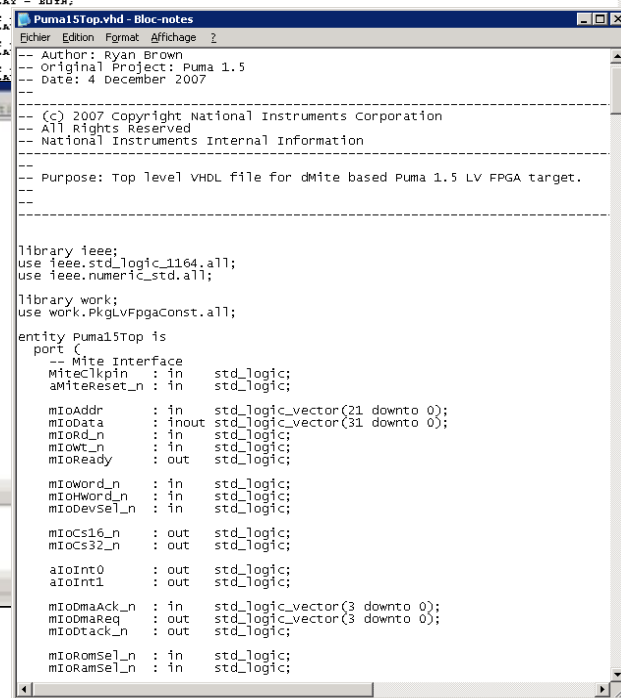
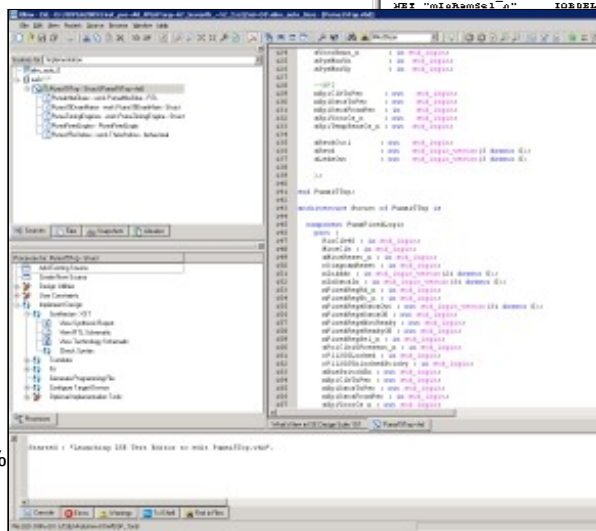
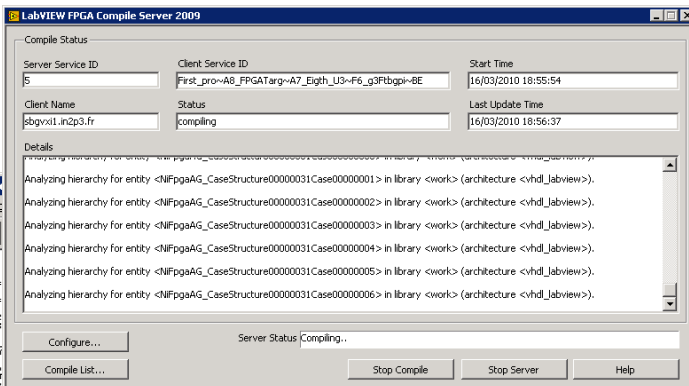
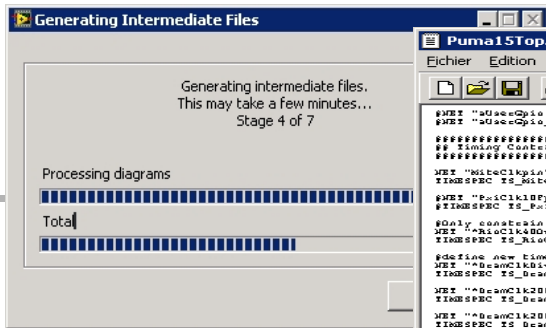
Generation du bit stream ...

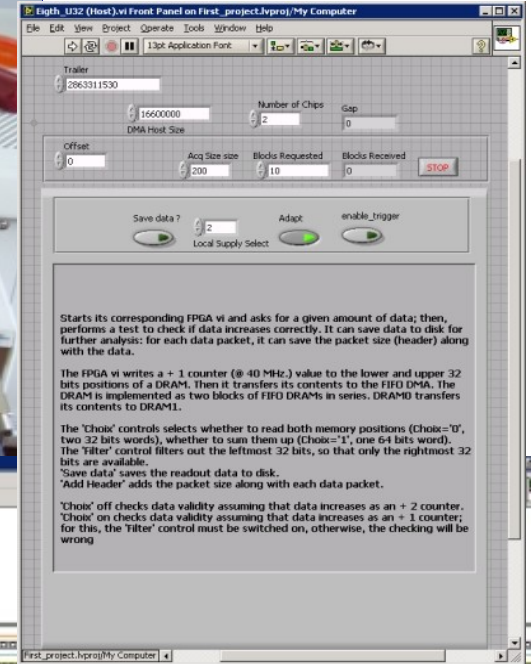


Device Utilization Summary:

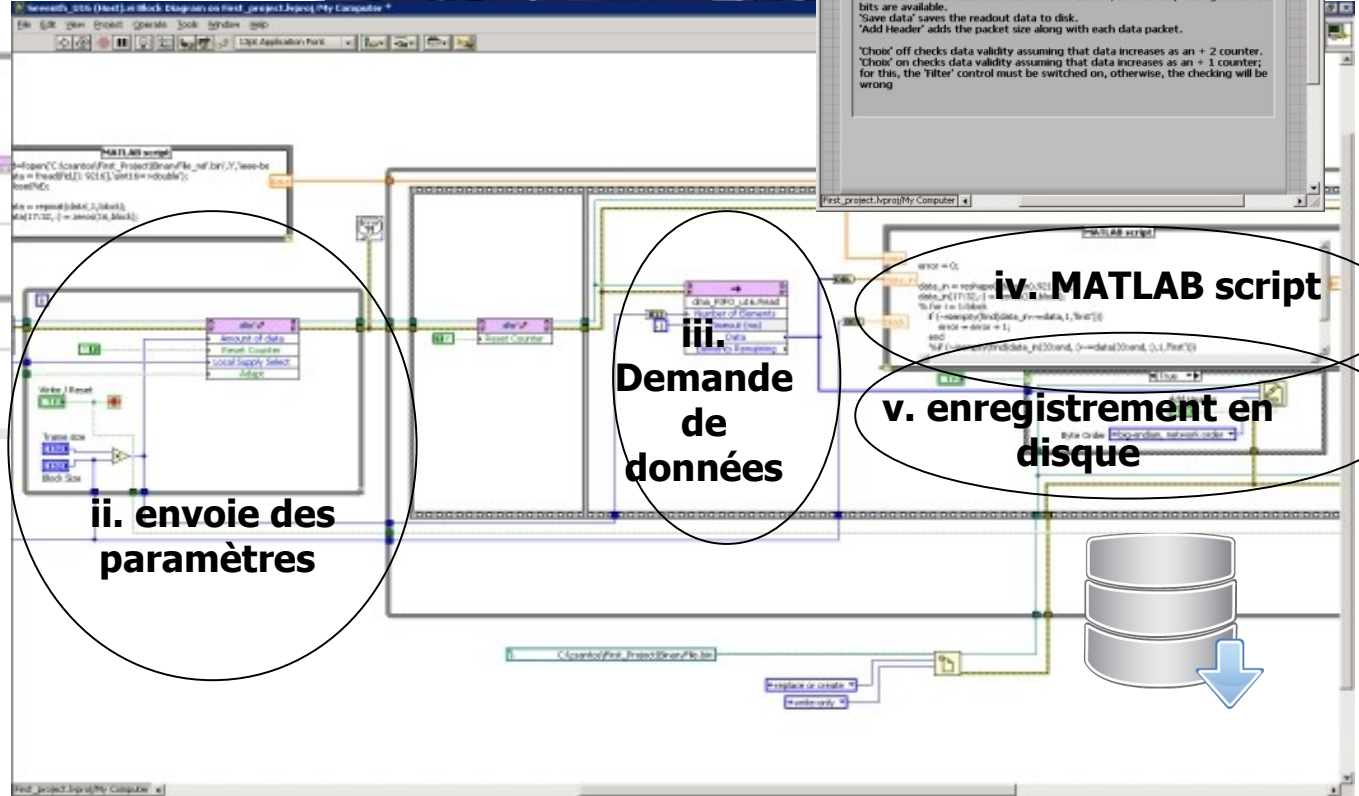
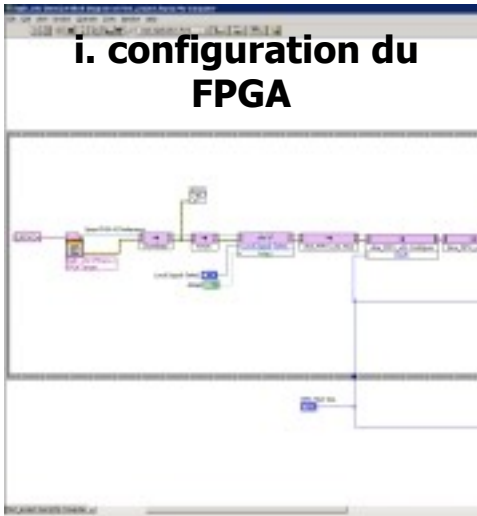
Selected Device :
5vlx85ff676-1

Number of BUFGRs	6 out of 32	18%
Number of BUFGRCTRLs	1 out of 32	3%
Number of BUFGRIOs	4 out of 56	7%
Number of DCIRESETs	1 out of 1	100%
Number of FIFO36_72_EXPs	2 out of 96	2%
Number of FIFO36_EXPs	2 out of 96	2%
Number of IDELAYCTRLs	3 out of 16	18%
Number of LOCed IDELAYCTRLs	3 out of 3	100%
Number of ILOGICs	103 out of 560	18%
Number of LOCed ILOGICs	4 out of 103	3%
Number of External IOBs	328 out of 440	74%
Number of LOCed IOBs	328 out of 328	100%
Number of IDELAYs	94 out of 560	16%
Number of LOCed IDELAYs	4 out of 94	4%
Number of External IPADs	6 out of 442	1%
Number of LOCed IPADs	6 out of 6	100%
Number of OLOGICs	168 out of 560	30%
Number of PLL_ADVs	1 out of 6	16%
Number of RAMB18X2s	3 out of 96	3%
Number of RAMB36SDP_EXPs	4 out of 96	4%
Number of STARTUPS	1 out of 1	100%
Number of SYMONs	1 out of 1	100%
Number of Slice Registers	11273 out of 51840	21%
Number used as Flip Flops	11273	
Number used as Latches	0	
Number used as LatchThrus	0	
Number of Slice LUTs	6005 out of 51840	11%
Number of Slice LUT-Flip Flop pairs	13001 out of 51840	25%





i. configuration du FPGA



iv. MATLAB script


v. enregistrement en disque


Interface de contrôle coté PC




Est-il possible d'aller encore plus loin ? ...

Il est possible de contourner le codage en LabView / FPGA

• LabView FPGA ne fournit l'accès qu'à un certain nombre, limité, des ressources matérielles dans le FPGA (inversion d'horloge ??) 


• Le codage en Labview simplifie le travail ...
• mais seulement SI on n'a pas d'expérience en codage HDL 

• Possibilité d'avoir différents domaines d'horloge ? 

• Instanciation de DCM, SRL16, DSP48, etc. ? 

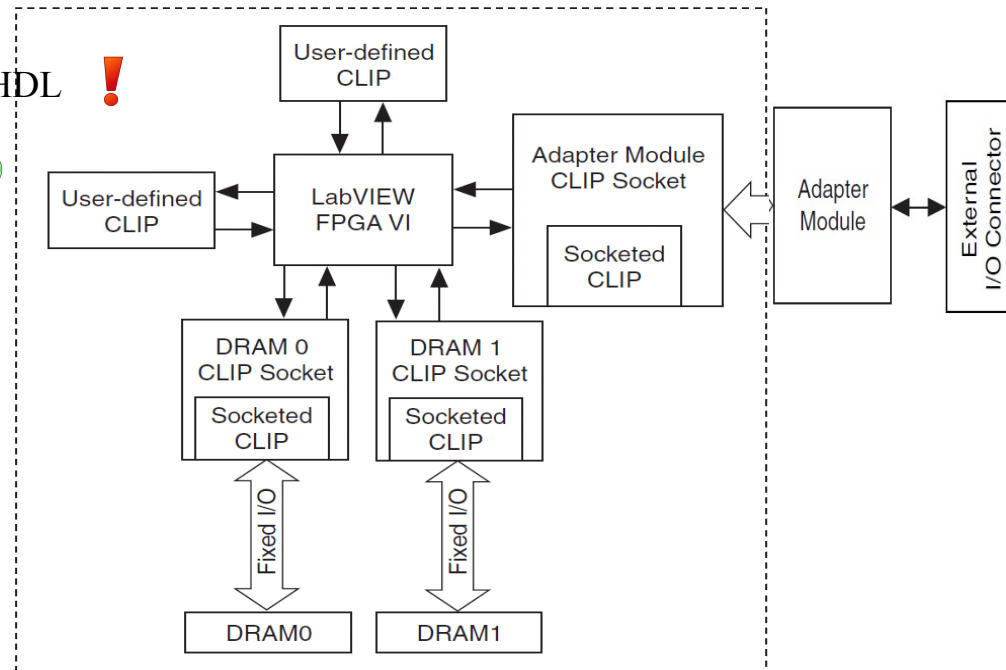
• Prévu pour intégrer IP de chez Xilinx ? 

• Permet l'utilisation de ressources FPGA I/O 
• spécifiques (IODELAY) ?

• Optimisation des ressources (vitesse, surface) ? 

• Possibilité de réutiliser du code existant ? 

*HDL Interface Node
User Defined CLIP
Socketed CLIP
IP Integration Node*



HDL Interface Node → Permet d'écrire en VHDL des dizaines de lignes de code

Parameters

Name	Direction	Type	Length
data	out	u64	--
enable	in	TF	--
rst	in	TF	--
reset_to	in	u64	--

Code

```
entity up_counter is
    generic(
        ClockFrequency : Integer := 40000000;
        InSingleCycle : boolean := true
    );
end up_counter;

architecture implementation of up_counter is
    signal data_tmp : unsigned(63 downto 0) := (others => '0');
    signal enable_tmp, rst_tmp : std_logic := '0';
begin
    process
    begin
        if (reset = '1' or rst_tmp = '0') then
            data_tmp <= unsigned(reset_to);
        elsif (enable_tmp = '1') then
            data_tmp <= data_tmp + 1;
        end if;
        wait until clk'event and clk = '1';
    end process;
end implementation;
```

- La limitation en nombre de lignes vient du fait d'avoir à rentrer le code dans l'espace prévu à cet effet
- GUI / Template LabView / FPGA
- Possibilité d'instancier au top niveau des composants fournis dans des fichiers externes
- Il faut fournir les signaux '*clk*' et '*aReset*'
- Pas pratique pour développer des projets plus larges

User Defined CLIP (i)

→ Méthode privilégiée pour la création de projets en VHDL

FPGA Module Data Type	VHDL Type
Boolean	std_logic
U8 and I8	std_logic_vector(7 downto 0)
U16 and I16	std_logic_vector(15 downto 0)
U32 and I32	std_logic_vector(31 downto 0)

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity DemoClipAdder is
port (
    clk      : in std_logic;
    aReset  : in std_logic;
    reset    : in std_logic;
    enable   : in std_logic;
    reset_to : in std_logic;
    data     : out std_logic_vector(7 downto 0);
end DemoClipAdder;
    
```

```

architecture rtl of DemoClipAdder is
    signal data_tmp : unsigned(7 downto 0);

    process (clk, aReset, reset)
    begin
        if (aReset = '1' or reset = '0') then
            data_tmp <= unsigned(reset_to);
        elsif (clk'event and clk = '1') then
            if (enable = '1') then
                data_tmp <= data_tmp + 1;
            end if;
        end if;
    end process;
    
```



Avantages par rapport à LV / FPGA:

- Exécution du code dans des multiples domaines d'horloge
- Possibilité d'inclure des contraintes dans la compilation
- Permet de communiquer directement avec pins d'entrée / sortie

```

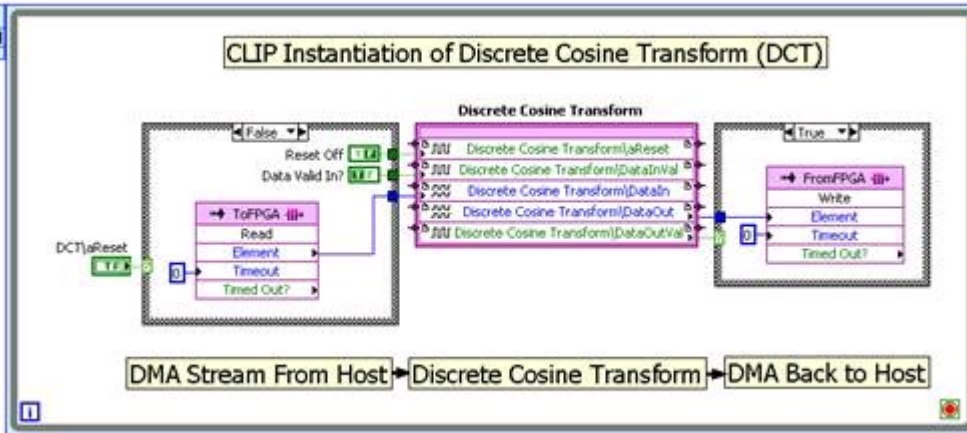
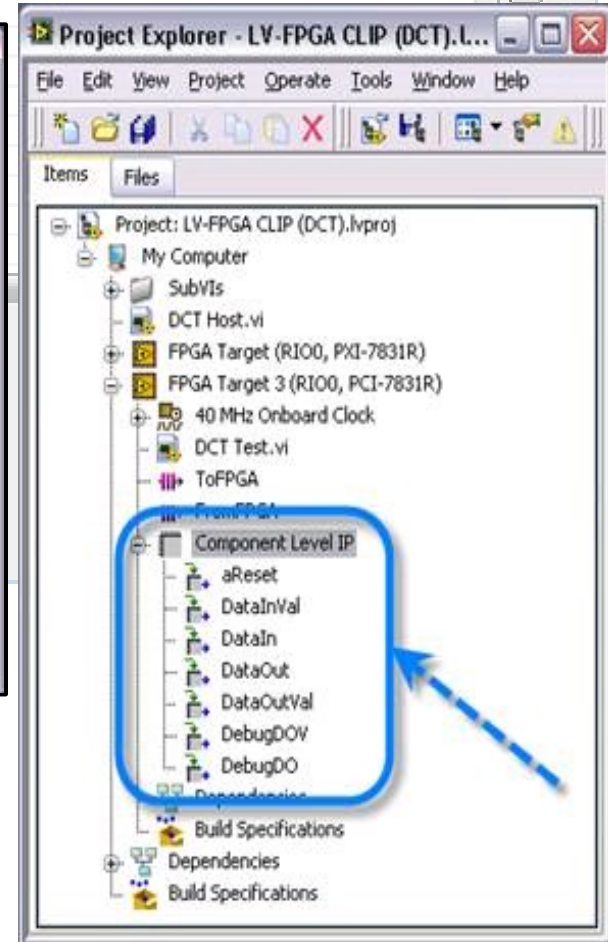
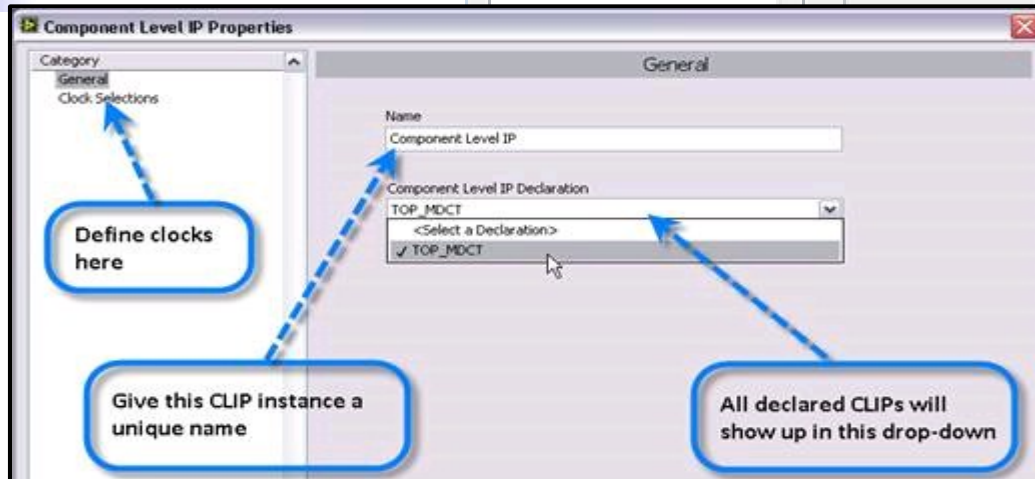
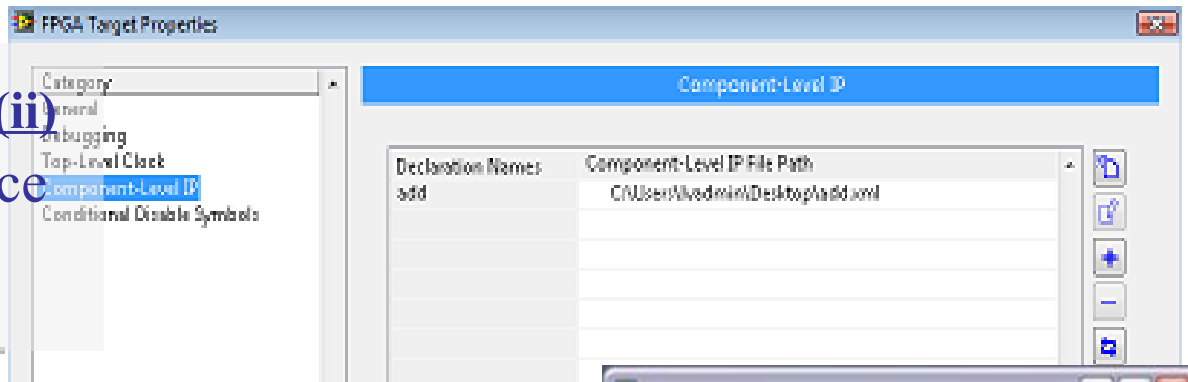
<InterfaceList>
  <Interface Name="DemoClipAdderIO">
    <InterfaceType>LabVIEW</InterfaceType>

    <SignalList>
      <Signal Name="Clk">
        <HDLName>clk</HDLName>
        <DataType>
          <Boolean />
        </DataType>
        <Direction>ToCLIP</Direction>
        <SignalType>clock</SignalType>
        <FreqInHertz>
          <Max>200M</Max>
          <Min>1M</Min>
        </FreqInHertz>
      </Signal>

      <Signal Name="reset_to">
        <HDLName>reset_to</HDLName>
        <DataType>
          <U64 />
        </DataType>
        <Direction>ToCLIP</Direction>
        <SignalType>data</SignalType>
    
```

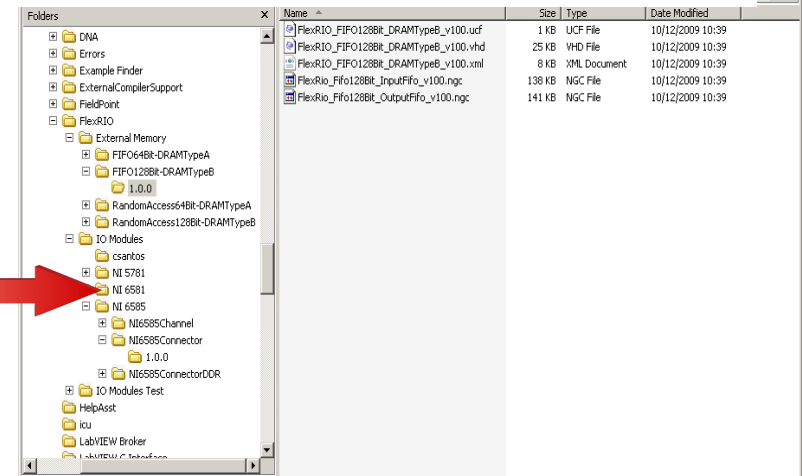
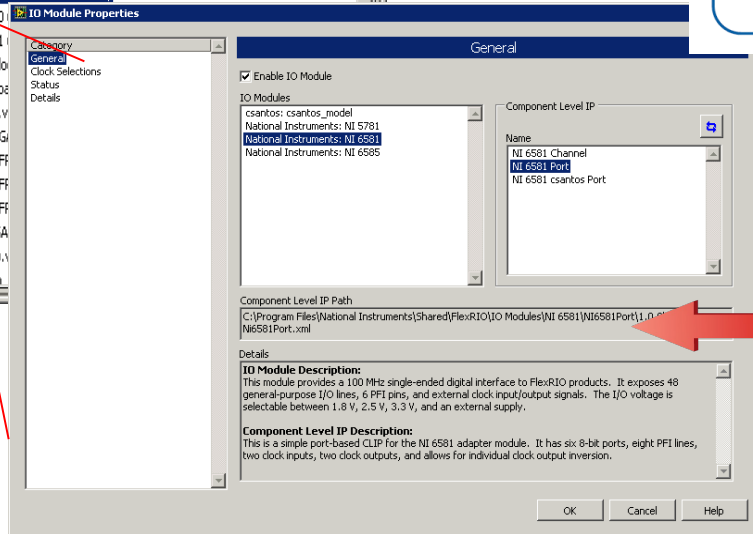
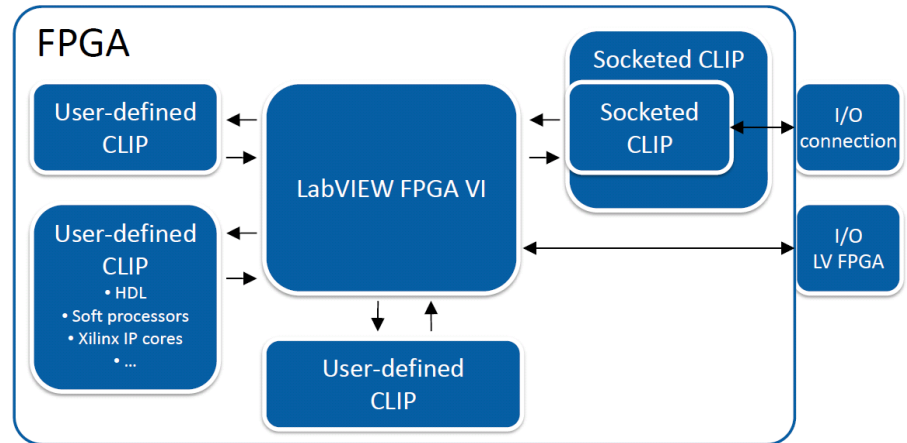
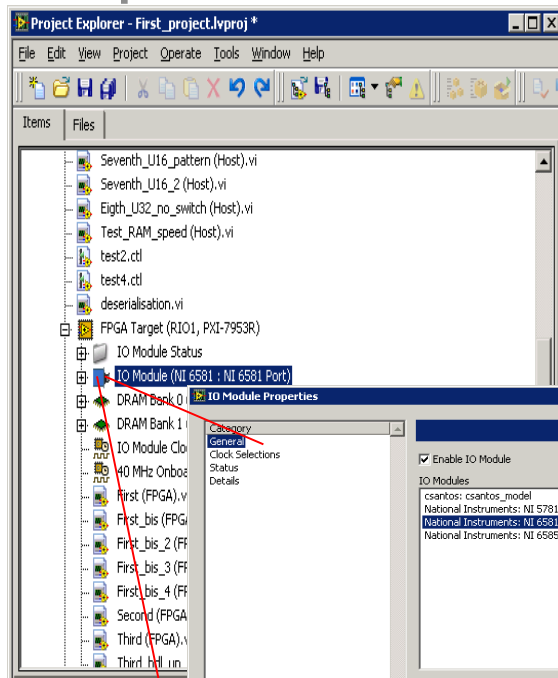
User Defined CLIP (ii)

→ Intégration via l'interface
LV / FPGA



Socketed CLIP (i) → Interface HDL avec périphériques externes

- Code HDL fournis par NI: passerelle pcb → vi LabView
- Fortement couplé avec l'interface LV / FPGA
- Liste de signaux / fonctionnalité fournie non modifiable
- Possibilité de rajouter des signaux (données) coté vi LV & custom netlist
- MIG Xilinx sous forme de fichier .ngc et enveloppe VHDL



Socketed CLIP (ii)

→ Instantiation de ressources Virtex V

- DCM, PLL, DSP48, IODELAY ...
- BUFGCE, BUFG ...

```
emacs@PXIE_DIGITAL
File Edit Options Buffers Tools VHDL Help
[Icons]
VCLKIN_IBUFG_INST : IBUFG
port map (I => CLKIN_IN,
          O => CLKIN_IBUFG);

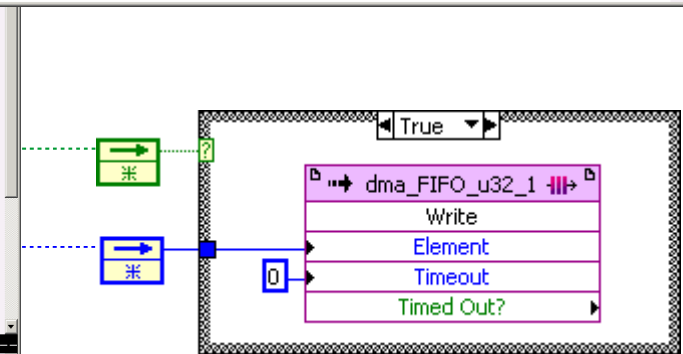
CLK0_BUFG_INST : BUFGCE
port map (I => CLK0_BUF,
          CE => aGpioEn,
          O => CLKFB_IN);

CLK180_BUFG_INST : BUFG
port map (I => CLK180_BUF,
          O => CLK180_OUT);

DCM_ADV_INST : DCM_ADV
generic map (CLK_FEEDBACK => "1X",
             CLKDV_DIVIDE => 2.0,
             CLKFX_DIVIDE => 1,
             CLKFX_MULTIPLY => 4,
             CLKIN_DIVIDE_BY_2 => false,
             CLKIN_PERIOD => 12.500,
             CLKOUT_PHASE_SHIFT => "NONE",
             DCM_AUTOCALIBRATION => true,
             DCM_PERFORMANCE_MODE => "MAX_SPEED",
             DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS",
             DFS_FREQUENCY_MODE => "LOW",
             );
--\--- clock_dcm.vhd 42% L59 (VHDL/es)---
```

```
emacs@PXIE_DIGITAL
File Edit Options Buffers Tools VHDL Help
[Icons]
BITSLLIP => '1', -- 1-bit Bitslip enable input
CE1 => '1', -- 1-bit clock enable input
CE2 => '1', -- 1-bit clock enable input
CLK => clk_in, -- 1-bit master clock input
CLKB => clk_in.n, -- clk input for DATA_RATE=DDR
CLKDIV => '0', -- 1-bit divided clock input
D => serial_data_in, -- connects to IODELAY or input b
OCLK => '0', -- 1-bit fast output clock input
RST => aResetSl, -- 1-bit asynchronous reset input
SHIFTIN1 => '0', -- 1-bit cascade Master/Slave inp
SHIFTIN2 => '0', -- 1-bit cascade Master/Slave inp
);

ISERDES_NODELAY_inst2 : ISERDES_NODELAY
generic map (
-- TRUE/FALSE to enable bitslip controller, must be "FALSE" in inte
BITSLLIP_ENABLE => true,
-- Specify data rate of "DDR" or "SDR"
DATA_RATE => "SDR",
-- Specify data width
-- NETWORKING SDR: 2, 3, 4, 5, 6, 7, 8 : DDR 4, 6, 8, 10
-- MEMORY SDR N/A : DDR 4
DATA_WIDTH => 8,
-- Use model - "MEMORY" or "NETWORKING"
INTERFACE_TYPE => "NETWORKING",
-- Define number or clock enables to an integer of 1 or 2
NUM_CE => 1,
--Set SERDES mode to "MASTER" or "SLAVE"
SERDES_MODE => "SLAVE")
port map (
Q1 => open, -- 1-bit registered SERDES output
Q2 => open, -- 1-bit registered SERDES output
Q3 => open, -- 1-bit registered SERDES output
Q4 => open, -- 1-bit registered SERDES output
Q5 => parallel_data_out(6), -- 1-bit registered SERDES output
Q6 => parallel_data_out(7), -- 1-bit registered SERDES output
);
--\--- deserialization.vhd 45% L70 (VHDL/es)---
```



IP Integration Node

→ Outil d'importation des IP générés par Xilinx Coregen (fichiers .xco)

- Gadget pratique à télécharger sur <http://decibel.ni.com/content/docs/DOC-5907>

- Donne accès à toutes les IP fournies par Xilinx (Core Generator)

1) Générer les fichiers .xco, .vhd, etc.

2) Importer ces fichiers à l'aide du node

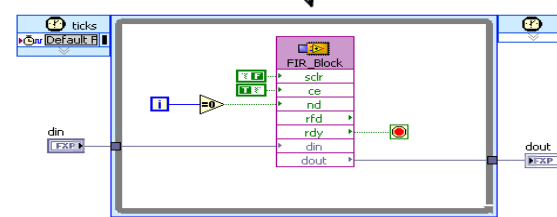
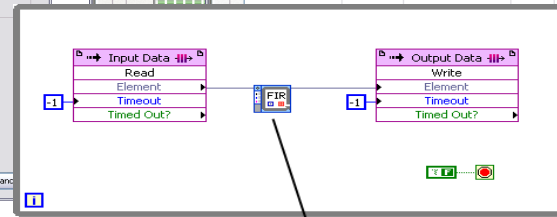
The screenshot shows the FIR Compiler interface. On the left, there is a plot of the frequency response. On the right, the 'Component Name' is 'FIR_Block'. The 'Coefficients File' is set to 'D:\projects\NIPI App Note\Fir.coe'. The 'Filter Specification' section shows 'Filter Type: Single Rate', 'Rate Change Type: Integer', 'Interpolation Rate Value: 1', 'Decimation Rate Value: 1', 'Zero Pack Factor: 1', and 'Number of Channels: 1'. The 'Frequency Specification' shows 'Input Sampling Frequency: 2'.

The screenshot shows the Xilinx CORE Generator interface. The 'IP Catalog' is open, showing various IP blocks. The 'FIR Compiler' IP is selected. The 'Information' section shows 'Core type: FIR Compiler', 'Version: 5.0', and a 'Core Summary' describing the tool's capabilities. The 'Actions' section shows 'Welcome to Xilinx CORE Generator. Help system initialized.'

The screenshot shows the 'IP Integration Node Properties' dialog. The 'IP Name' is 'FIR_Block'. The 'IP Source' is 'IP'. The 'IP Terminals' table is shown below:

Name	Direction	Data Type	Hide?	Default Value
scr	IN	Boolean	No	N/A
ce	IN	Boolean	No	N/A
rd	IN	Boolean	No	N/A
rfd	OUT	Boolean	No	N/A
rfdy	OUT	Boolean	No	N/A
din	IN	FXP <+/-,16,1>	No	N/A
dout	OUT	FXP <+/-,38,7>	No	N/A

The 'Terminal Order' section shows 'Move Up' and 'Move Down' buttons. The 'Terminal Properties' section shows 'Data type' set to 'FXP', 'Fixed-point encoding' set to 'Signed', and 'Integer word length' set to 7.






Conclusions

Avantages dans l'approche LabView / FPGA

- Permet de se concentrer sur le firmware / software
- Le matériel est prêt à l'emploi → pas de test, bugs et autres problèmes
- Système ouvert et flexible: facilement extensible et customisable
- Pas limité à une seule application (rentabilité de l'investissement):
basé sur un FPGA Virtex V
- Modules d'entrée / sortie échangeables
- Système utilisé en faisceau de test au CERN avec des résultats satisfaisants

Il est possible de contourner le codage en LabView ? ...

- Fort couplage entre le code et la GUI de développement / LabView FPGA 
- NI fournis les instruments de base pour intégrer du code VHDL 
- Il est possible d'aller au delà et customiser le projet ... jusqu'à un certain point
 - Pas d'accès aux options d'un projet ISE / Xilinx
 - Pas de connections JTAG → pas de Chipscope (débugage matériel) 
 - NI rajoute une couche supplémentaire ... dommage

Questions ?

 Cayetano Santos
cayetano.santos@iphc.cnrs.fr

Efficiency vs Threshold

