

Le FPGA est-il toujours la solution aux traitements temps-réels de signaux radio-fréquences?

méthodes de développement et écosystèmes

G. Goavec-Merou, P.-Y. Bourgeois, J.-M. Friedt

Institut FEMTO-ST, département Temps-Fréquence,
Univ. de Franche Comté, CNRS, ENSMM, Besançon
Contact : gwenhael.goavec@femto-st.fr

Financement : LabeX **FIRST-TF**



transparents disponibles sur
http://www.trabucayre.com/workshopFirstTF_prezoFPGA.pdf

FPGA or not FPGA ?

Le FPGA est-il toujours **LA** solution aux traitements temps-réels de signaux radio-fréquences ?

Oui ... et non

→ Ça dépend

Oui :

- gestion d'un flux rapide (>20 ME/s) ;
- synchronisation fine ;
- traitements exotiques.
- les traitements nécessitent du massivement parallèle ;

Non ou à discuter :

- pas ou peu de contraintes temporelles ;
- interfaces utilisateurs / communications ;
- utilisation de bibliothèques ou de *frameworks* (GNURadio).

Adapter les ressources aux besoins ⇒ choix du composant

Comparaison du développement

GCPU :

- 1 éditeur de texte
- 2 compilation/vérification du code

```
arm-none-eabi-gcc -mfloat-abi=hard \  
-Os [...] \  
-o main.elf -c main.c  
arm-none-eabi-objcopy -Obinary \  
main.elf main.bin
```

- 3 chargement

```
st-flash --reset write main.bin 0x8000000
```

- 4 debug

```
arm-none-eabi-gdb main.bin
```

Possibilité de se baser sur des bibliothèques (`libopencm3`) ou des exécutifs (`FreeRTOS`¹, `NuttX`², `RTEMS`³)

≤ 1min

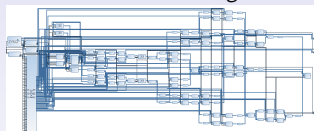
1. Q. Macé, J.-M Friedt `FreeRTOS` : application à la réalisation d'un analyseur de réseau numérique sur STM32, GLMF 207
2. G. Goavec, J.-M. Friedt : Un environnement exécutif visant la compatibilité POSIX : `NuttX` pour contrôler un analyseur de réseau à base de STM32, GLMF 210
3. J.-M Friedt, G. Goavec : Interfaces matérielles et OS libres pour Nintendo DS : `DSLlinux` et `RTEMS`, GLMF HS 43

FPGA :

simulation



Création du *design*



Debug : analyseur logique...

Méthodes de dev : voir seconde partie

Temps de génération du binaire :

≥ 1min → < 1h

Exemples

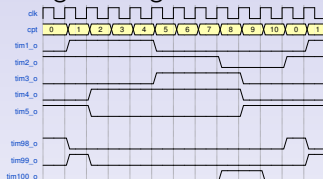
Génération de signaux selon un motif répétitif (*shutter* sur horloge optique)

3 signaux à générer



⇒ un microcontrôleur est suffisant

100 signaux à générer

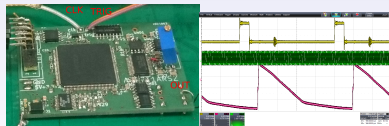


⇒ un FPGA est nécessaire

Monitoring : acquisition chaque seconde, affichage/stockage/transfert ⇒ GCPU.

Stabilisation de la base de temps d'un Radar de sol pour de l'interrogation de capteurs enterrés. Génération d'une rampe de $8\mu s$ sur 256 pas donc DAC@40 MHz, déclenchement sur trigger.

⇒ FPGA (iCE40)



Traitement d'un flux issu d'un double ADC 16 bits@1GE/s ⇒ FPGA

Alternatives

Selon les cas :

- un processeur généraliste exécutant un système d'exploitation (GNU/Linux) pour les tâches non temps-réelles ;
- un microcontrôleur monotâche pour les latences critiques ;
- Approche hétérogène processeur-microcontrôleur⁴. Quelques exemples :
 - TI Sitara am3355 : cortex A9@1 GHz + deux PRU @200 MHz ;
 - TI OMAP4460 : dual-core A9@1.2 GHz + dual Cortex-m3 + DSP (c6x)
 - Freescale/NXP iMX8 : dual-core cortex A72 + quad-core cortex A53 + deux cortex-M4F
 - Xilinx Zynq UltraScale+ MPSoC : dual-core cortex A53 @1.5 GHz + dual-core cortex R5 @600 MHz + FPGA

Communication entre cœurs par mémoire partagée et interruptions + infrastructures : IPC (*Inter Processor Communication*), AMP (*Asymmetric Multi Processor*), remoteproc et rpmsg.

4. P.-J. Texier, J. Charbrerie, i.MX7 : Communication interprocesseur, donnons vie au Cortex M4, GLMF 211

Un FPGA est :

- un outil reconfigurable comparé à la contrainte d'un jeu d'instructions figé.
- capable de réaliser des téra-opérations par seconde ;
- temps-réel.

Mais :

- le temps de développement est généralement plus long ;
- c'est un composant souvent cher ;
- n'est pas la seule option pour du traitement et le choix nécessite de bien évaluer le besoin par rapport à un objectif.

Méthodes de développement

- bas niveau/historique : développement à partir de 0 ou utilisation de l'existant. Approche IP ou blocs⁵ ;
- intermédiaire : génération de code à partir de classes écrites dans un langage de haut-niveau^{6 7} ;
- haut-niveau : analyse et conversion de code entre un langage de haut niveau et du code HDL^{8 9}.

5. K. Benkrid, D. Crookes, A. Benkrid : Towards a general framework for FPGA based image processing using hardware skeletons, Parallel Computing 28, Issues 7-8, Aug. 2002

6. S. Bourdeauducq : Migen, une "boîte à outils" en python, GLMF 149

7. J.I. Villar, J. Juan, M.J. Bellido, J. Viejo, D. Guerrero and J. Decaluwe, Python as a hardware description language : A case study, 2011 VII Southern Conference on Programmable Logic (SPL), Cordoba, 2011

8. R. A. Walker, R. Camposano : A survey of high-level synthesis systems. Kluwer Academic Publishers, 1991.

9. J. Cong, B. Lui, S. Neuendorffer, J. Noguera, K. Vissers et Z. Zhang : High-level synthesis for fpgas : From prototyping to deployment. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, 30(4) :473491, March 2011.

Bas-niveau/historiques

Le FPGA pour
les traitements
temps-réels de
signaux
radio-fréquences

G. GOAVEC-
MEROU

FPGA or not
FPGA ?

Méthodes de
développement

Écosystèmes

Utilisation d'IPs (*Intellectual Property*) libres ou propriétaires et prêtes à l'emploi :

⇒ accélération du temps de mise en œuvre, mais **boîtes noires** ;

Développement HDL (*Hardware Description Language*) :

⇒ **contrôle total**... Mais temps de développements allongés, simulations obligatoires et parfois résultats moins performants que l'existant.

Migen

```
from migen import *

from migen.fhdl import *
from migen.build.generic_platform import →
    ↳ Pins, IOStandard
from migen.build.platforms import de0nano
from migen.fhdl.specials import Special

class MyPresc(Module):
    def __init__(self, max_value, tick):
        counter = Signal(max=max_value)
        self.sync += If(counter == max_value-1,
            tick.eq(1),
            counter.eq(0)
        ).Else(counter.eq(counter+1), tick.eq→
            ↳ (0))

plat = de0nano.Platform()
led0 = [plat.request("user_led", x) for x →
    ↳ in range(8)]
m = Module()
tick = Signal()
m.submodules += MyPresc(pow(2,23), tick)
led_cpt = Signal(8)
m.sync += If(tick, led_cpt.eq(led_cpt + 1))
m.comb += [led0[x].eq(led_cpt[x]) for x in →
    ↳ range(8)]
print(verilog.convert(MyPresc(pow(2,23), →
    ↳ tick)))
plat.build(m, toolchain_path="/opt/altera→
    ↳ /15.0/quartus/bin")
```

Utilisé sur des projets tels que Artiq¹⁰

10. <https://m-labs.hk/artiq>

Intermédiaires

MyHDL

```
import myhdl
from myhdl import *

@block
def Chenillard(tick, leds, clock, reset, →
    ↳ leds_size):
    leds_s = Signal(intbv(0)[leds_size:0]);
    sens_s = Signal(intbv(1, min=-1, max=2))
    maxcnt = int(pow(2,leds_size))-1

    @always_seq(clock.posedge, reset=reset)
    def chenillard():
        if tick == 1:
            if leds_s == maxcnt-1:
                sens_s.next = -1
            elif leds_s == 1 and sens_s == -1:
                sens_s.next = 1
            leds_s.next = leds_s + sens_s;

    @always_comb
    def assign_leds():
        leds.next = leds_s
    return chenillard, assign_leds
```

Chisel (langage Scala)

utilisé pour implémentation
Sifive (cœur *Open Source*
riscV).

- HLS : code C/C++ transformé en code HDL. Des directives `#pragma` peuvent être nécessaires pour guider l'outil ;
- SDSoC : code C/C++ distribué entre le GCPU et le FPGA (conversion par HLS ou utilisation de blocs existants). Ajout de glues pour la communication. Quid du traitements de flux rapides et de la consommation de ressources induite par l'infrastructure ?
- Matlab HDLCoder

- Artiq¹¹(m-labs/NIST) : génération du bistream à partir de python + code python sur le processeur et communications par RPC ;
- RFNoC¹²(EttusResearch) : ajout de traitements HDL dans le firmware d'une USRP, routage du flux entre les blocs au travers de GNURadio \Rightarrow SDR (*Software Define Radio*) ;
- Pyrpl¹³(LKB) : couche d'abstraction logicielle sur plate-formes à base de FPGA avec frontends numériques-analogiques + GUI ;
- écosystème DTF(FEMTO-ST) : normalisation interfaces, pilotes + bibliothèques pour abstraction logicielle, wrapper python, GUI Qt, outils divers ;
- autres : discussions/présentations dans cette session.

11. https://m-labs.hk/artiq/slides_timing.pdf

12. <https://www.ettus.com/sdr-software/detail/rf-network-on-chip>

13. <http://pyrpl.readthedocs.io/en/latest>