

Linux et le System on Chip SoC

email : kadionik@enseirb.fr
http : <http://www.enseirb.fr/~kadionik>
<http://www.enseirb.fr/cosynux/>

Patrice KADIONIK, P. NOUEL

Groupe de recherche en COncption de SYStèmes NUmériques - ENSEIRB



COS

Linux et le System on Chip SoC



INTRODUCTION

- Cette présentation a pour objectif de préciser comment sont conçus actuellement les circuits numériques complexes avec l'approche système et le système sur silicium SoC (*System on Chip*).
- La méthodologie de conception appelée *codesign* sera expliquée. Elle autorise un développement conjoint du matériel (*hardware*) et du logiciel (*software*).
- Les principaux processeurs « logiciels » ou *softcore* seront passés en revue et mettant en avant leur adéquation à exécuter du logiciel libre.



COS

Linux et le System on Chip SoC



INTRODUCTION

- Linux pour l'embarqué est de plus en plus utilisé et sa mise en oeuvre au travers de μ Clinux sera décrite dans le cadre de l'offre de *codesign* d'Altera pour le *System on Programmable Chip* (SoPC).
- Un exemple couplant μ Clinux à un système SoPC sera donné pour montrer l'intérêt du codesign lors de la conception de systèmes numériques complexes.



COS(Δ)

Linux et le System on Chip SoC



PARTIE 1 : LA CONCEPTION DES SYSTÈMES NUMÉRIQUES COMPLEXES



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- Les systèmes numériques deviennent aujourd'hui de plus en plus complexes au niveau intégration et fonctionnalités et l'on est en mesure d'intégrer tout dans un même composant
- C'est le concept du *single chip*.
- Ceci est en fait lié à la loi empirique de Moore qui stipule que pour une surface de silicium donnée, on double le nombre de transistors intégrés tous les 18 mois !



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

	1998	1999	2001	2002	2004
Technologie µm	0,25 µm	0,18 µm	0,15 µm	0,13 µm	0,09
Complexité M en Millions de transistors	1 M	2-5 M	5-10 M	10-25 M	> 25
Loi de Moore					



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- On travaille maintenant au niveau système (ou fonctionnalité) et non au niveau porte logique (pour le grand bien des électroniciens).
- Les fonctionnalités peuvent être implantées dans des composants spécifiques de type ASIC (*Application Specific integrated Circuit*). On parle alors de Système sur Silicium SoC (*System on Chip*).
- Les fonctionnalités peuvent être implantées dans des composants logiques programmables de type FPGA (*Field Programmable Gate Array*). On parle alors de système SoPC (*System on Programmable Chip*).

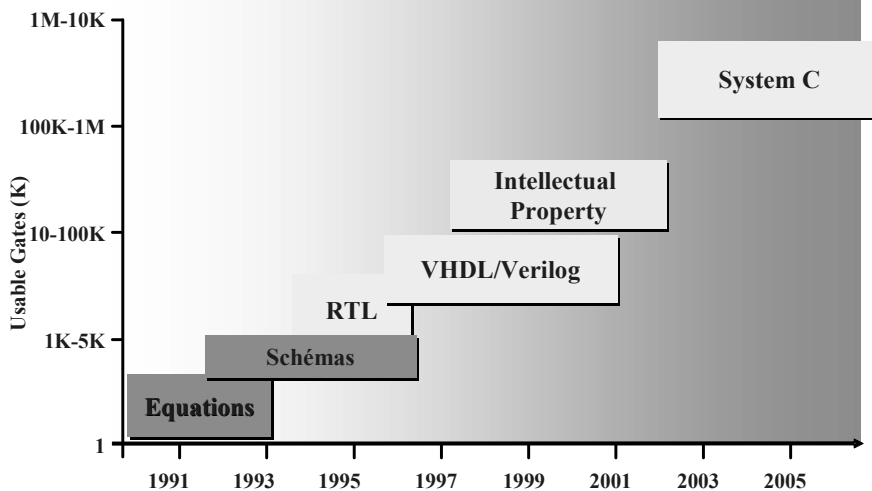


COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- L'approche « schématique » au niveau porte logique ou fonctionnalités de base RTL (*Register Transfer Logic*) est délaissée pour la conception des systèmes complexes au profit d'une approche « textuelle ».
- On utilise des langages de description de matériel comme VHDL (*Very high speed integrated circuit Hardware Description Language*) ou Verilog pour synthétiser une fonctionnalité numérique.
- Ces langages de description de matériel sont en fait de véritables langages de programmation informatiques, orientés objet. Ils sont utilisés conjointement avec un synthétiseur (compilateur) ou un simulateur.



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- Ces langages ont permis de travailler avec un niveau d'abstraction plus grand laissant les basses besognes au synthétiseur.
- On a pu rapidement développer des bibliothèques de fonctionnalités comme une interface USB, un contrôleur MAC Ethernet que l'on appelle blocs IP (*Intellectual Property*).
- On peut les acheter ou bien utiliser des blocs IP libres (comme du logiciel libre) dont le site phare de référence est <http://www.opencores.org>.



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- On peut ainsi voir la conception d'un système numérique complexe comme un assemblage de blocs IP si bien que les langages de description de matériel sont un peu comme un langage assembleur vis à vis d'un langage plus évolué comme le langage C.
- Il est à noter que le nirvana serait de pouvoir générer un circuit numérique à partir d'un fichier source écrit en langage informatique comme le langage C : c'est ce que propose SystemC mais bien des progrès restent encore à faire dans ce domaine...



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- Les langages de description de matériel sont aussi intéressants pour la facilité de modification et de réutilisation d'un design précédent pour un nouveau design : c'est le *design reuse*.
- Cela permet de réduire aussi le *Time To Market* (TTM) !



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- Lorsque l'on conçoit un système numérique complexe, on met en oeuvre généralement un processeur embarqué.
- Ce processeur embarqué est :
 - Soit un bloc IP : on parle de processeur *softcore*.
 - Soit déjà implanté dans le circuit électronique en « dur » : on parle de processeur *hardcore*. Le processeur de ce type est généralement plus performant que le processeur du type précédent.



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- Le processeur embarqué allie la souplesse du logiciel à l'accélération du temps d'exécution du matériel.
- Une fonctionnalité particulière peut donc être composée d'une partie matérielle couplée à une fonctionnalité logicielle dédiée : on a donc une conception conjointe matérielle-logicielle ou *codesign*.
- Le *codesign* implique donc une conception en même temps du matériel et du logiciel, ce qui est une nouvelle méthodologie par rapport à la méthodologie de conception classique (conception matérielle puis conception logicielle)...



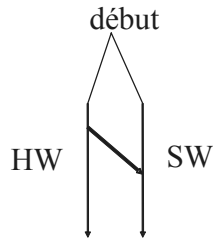
COS(Δ)

Linux et le System on Chip SoC



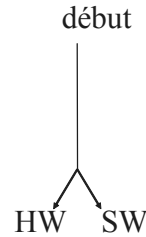
CONCEPTION DES SYSTÈMES NUMÉRIQUES

Conception traditionnelle



Réalisée par des groupes d'ingénieurs indépendants

Codesign



Réalisée par le même groupe d'ingénieurs en coopération



COS(Δ)

Linux et le System on Chip SoC



CONCEPTION DES SYSTÈMES NUMÉRIQUES

- Les processeurs embarqués comme leurs cousins du grand public sont de plus en plus puissants.
- La puissance CPU en MIPS double les 2 ans (loi empirique de Joy) et sont de plus communicants.
- On estime que l'on a besoin d'une bande passante réseau de 0,3 à 1 Mb/s par MIPS (loi empirique de Ruge).
- La course à la performance gagne aussi le monde de l'embarqué et des systèmes numériques complexes !



COS(Δ)

Linux et le System on Chip SoC



PARTIE 2 : LES PROCESSEURS POUR LE SOPC



COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC

- Le choix d'un processeur pour le SoPC peut se faire sur différents critères :
 - Processeur *hardcore* : pour ses performances au détriment de la flexibilité.
 - Processeur *softcore* : pour sa flexibilité de mise à jour au détriment de performances moindres que le précédent. La portabilité vers n'importe quel circuit FPGA est assurée en étant donc circuit FPGA indépendant. Il est aussi possible de migrer vers un circuit de type ASIC en cas d'une production en grande série.
- Généralement, on privilégie les processeurs *softcore* pour s'affranchir des problèmes d'obsolescence et pour pouvoir bénéficier facilement des évolutions apportées en refaisant une synthèse.



COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC

- Le processeur *softcore* peut être libre :
 - Il est décrit en langage de description de matériel (VHDL, Verilog).
 - Le code source peut être librement distribué et implanté dans n'importe quel circuit programmable FPGA.
 - On est alors indépendant du type de circuit FPGA.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 19 -



LES PROCESSEURS POUR LE SOPC

- Le processeur *softcore* peut être propriétaire :
 - Il est distribué par exemple sous forme d'une *netlist* pour être implantée dans un circuit FPGA.
 - Il est généralement lié à un fondeur de circuit FPGA particulier (comme Altera ou Xilinx).
 - On ne peut pas l'utiliser dans un circuit FPGA autre que celui pour lequel il est prévu. On a donc ici une boîte noire.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 20 -



LES PROCESSEURS POUR LE SOPC

- On trouvera principalement au niveau des processeurs *softcores* libres :
 - Le processeur Leon <http://www.gaisler.com/index.html>.
 - Le processeur OpenRisc <http://www.opencores.org/projects.cgi/web/or1k/overview>.
 - Le processeur F-CPU <http://www.f-cpu.org>.
 - Autres processeurs : clones de 6800, 68HC11, 68K, PIC : http://www.opencores.org/browse.cgi/filter/category_microprocessor



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 21 -



LES PROCESSEURS POUR LE SOPC

- On trouvera principalement au niveau des processeurs *softcores* propriétaires :
 - Le processeur NIOS et NIOS II d'Altera <http://www.altera.com>.
 - Le processeur Microblaze de Xilinx <http://www.xilinx.com>.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 22 -



LES PROCESSEURS POUR LE SOPC

PROCESSEUR SOFTCORE LEON



COS(Δ)

Linux et le System on Chip SoC

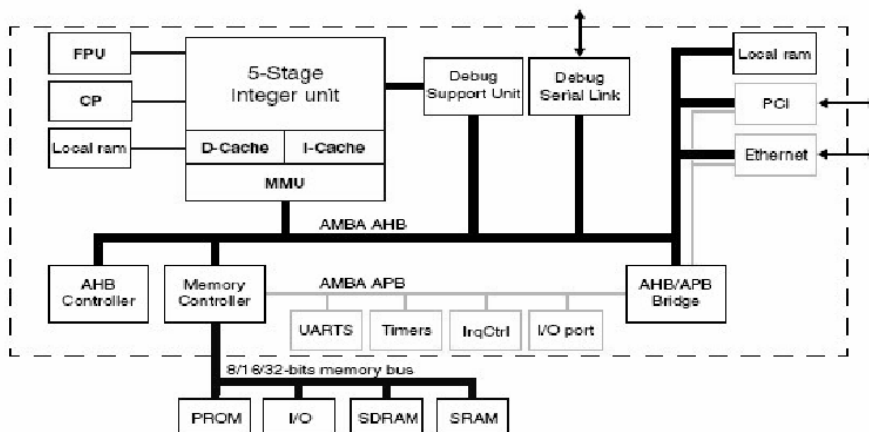


© pk/pn/2005 v 1.0

- 23 -

LES PROCESSEURS POUR LE SOPC

LEON : ARCHITECTURE



COS(Δ)

Linux et le System on Chip SoC



© pk/pn/2005 v 1.0

- 24 -

LES PROCESSEURS POUR LE SOPC

LEON : ARCHITECTURE

- Jeu d'instructions type SPARC V8 32 bits.
- 8 registres globaux, fenêtres de registres.
- Pipeline à 5 niveaux.
- Unité MAC et FPU (multiplication/division hardware).
- Cache d'instructions et de données configurable.
- Bus interne AMBA (ARM).
- MMU.

- Disponible en libre sous forme d'un module IP VHDL (*opencore*).



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 25 -



LES PROCESSEURS POUR LE SOPC

LEON : LOGICIELS

- Chaîne d'outils GNU (compilation croisée).
- Simulateur LEON TSIM.
- OS supportés :
 - μ Clinux.
 - Linux.
 - eCos.
 - RTEMS.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 26 -



LES PROCESSEURS POUR LE SOPC

PROCESSEUR SOFTCORE OPENRISC 1200



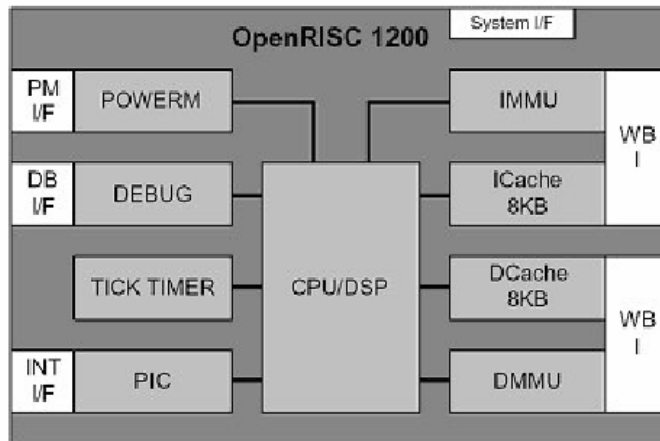
COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC

OPENRISC 1200 : ARCHITECTURE



COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC *OPENRISC 1200 : ARCHITECTURE*

- Jeu d'instructions 32 bits.
- Instruction *User Custom*.
- 32 registres de données au maximum.
- Pipeline à 5 niveaux.
- Unité MAC, FPU.
- Cache d'instructions et de données configurable.
- Bus interne Wishbone.
- MMU.

- Disponible en libre sous forme d'un module IP Verilog (*opencore*).



COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC *OPENRISC 1200 : ARCHITECTURE*

- Chaîne d'outils GNU (compilation croisée).
- Simulateur or1Ksim.
- OS supportés :
 - μ Clinux.
 - Linux.
 - eCos.
 - RTEMS.



COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC

PROCESSEUR SOFTCORE F-CPU



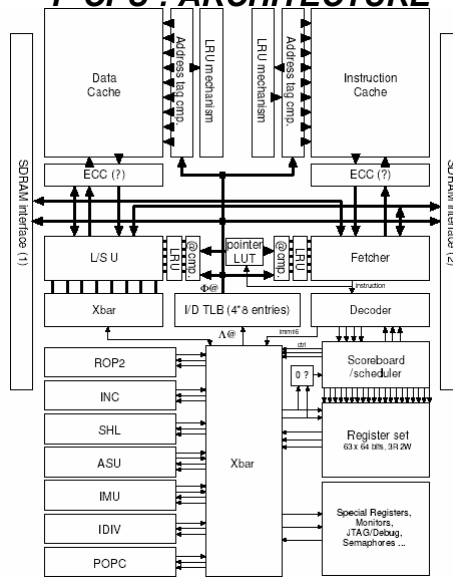
COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC

F-CPU : ARCHITECTURE



COS(Δ)



LES PROCESSEURS POUR LE SOPC

F-CPU : ARCHITECTURE

- Jeu d'instructions 32 bits.
- 64 registres 64 bits.
- Disponible en libre sous forme d'un module IP VHDL (*opencore*).
- En cours de développement...



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 33 -



LES PROCESSEURS POUR LE SOPC

F-CPU : ARCHITECTURE

- Chaîne d'outils GNU (compilation croisée).
- Pas d'OS.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 34 -



LES PROCESSEURS POUR LE SOPC

PROCESSEUR SOFTCORE MicroBlaze



COS

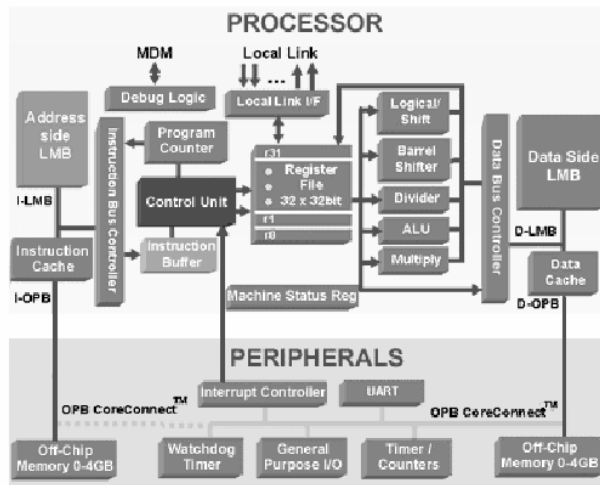
Linux et le System on Chip SoC



ENSEIRB
ÉCOLE NATIONALE SUPÉRIEURE
ÉLECTRONIQUE INFORMATIQUE & TELECOMMUNICATIONS
BORDEAUX

LES PROCESSEURS POUR LE SOPC

MicroBlaze : ARCHITECTURE



COS

Linux et le System on Chip SoC



ENSEIRB
ÉCOLE NATIONALE SUPÉRIEURE
ÉLECTRONIQUE INFORMATIQUE & TELECOMMUNICATIONS
BORDEAUX

LES PROCESSEURS POUR LE SOPC

MicroBlaze : ARCHITECTURE

- Jeu d'instructions 32 bits.
- 32 registres de données.
- Pipeline à 3 niveaux.
- Cache d'instructions et de données configurable.
- Bus interne LMB et CoreConnect IBM.
- Pas de MMU.

- Produit commercial de Xilinx.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 37 -



LES PROCESSEURS POUR LE SOPC

MicroBlaze : ARCHITECTURE

- Plateforme de développement Xilinx XPS.
- Chaîne d'outils GNU (compilation croisée).
- Simulateur XMD.
- OS supportés :
 - μ Clinux. Portage GPL
<http://www.itee.uq.edu.au/~jwilliams/mblaze-uclinux/>
 - microC/OS II. Produit commercial.
 - Noyau ATI. Produit commercial.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 38 -



LES PROCESSEURS POUR LE SOPC

PROCESSEUR SOFTCORE NIOS



COS(Δ)

Linux et le System on Chip SoC



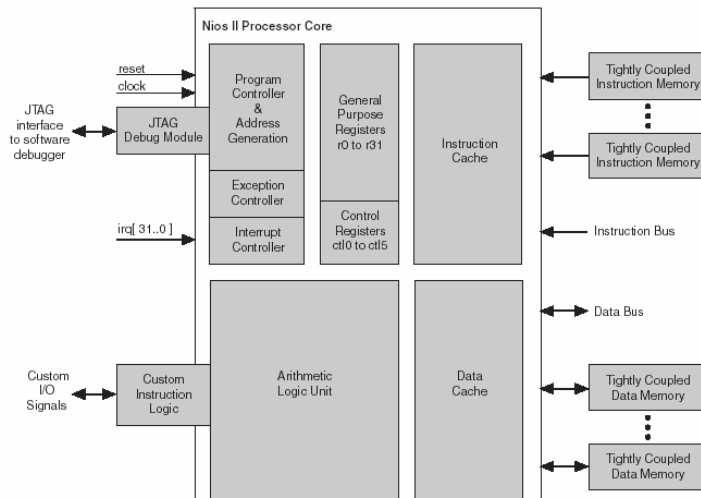
ENSEIRB
ÉCOLE NATIONALE SUPÉRIEURE
ÉLECTRONIQUE INFORMATIQUE & INFORMATIQUES
BORDEAUX

© pk/pn/2005 v 1.0

- 39 -

LES PROCESSEURS POUR LE SOPC

NIOS II : ARCHITECTURE



COS(Δ)

Linux et le System on Chip SoC



ENSEIRB
ÉCOLE NATIONALE SUPÉRIEURE
ÉLECTRONIQUE INFORMATIQUE & INFORMATIQUES
BORDEAUX

© pk/pn/2005 v 1.0

- 40 -

LES PROCESSEURS POUR LE SOPC

NIOS II : ARCHITECTURE

- 2 versions : NIOS I et NIOS II. Seule NIOS II présentée.
- 3 cores possibles : *fast, economy, standard*.
- Jeu d'instructions 32 bits.
- 32 registres dont 6 de contrôle.
- Pipeline à 6 niveaux (*fast*).
- Cache d'instructions et de données configurable.
- Bus interne Avalon.
- Pas de MMU.
- Produit commercial d'Altera.



COS(Δ)

Linux et le System on Chip SoC



LES PROCESSEURS POUR LE SOPC

NIOS II : ARCHITECTURE

- Plateforme de développement Quartus II.
- Chaîne d'outils GNU (compilation croisée).
- IDE Eclipse
- Simulateur ModelSim.
- OS supportés :
 - μ Clinux. Portage GPL
<http://www.niosforum.com/>
 - microC/OS II. Produit commercial.
 - Noyau Nucleus. Produit commercial.
 - eCos.



COS(Δ)

Linux et le System on Chip SoC



PARTIE 3 : PERFORMANCES DES PROCESSEURS SOFTCORE



COS(Δ)

Linux et le System on Chip SoC



PERFORMANCES PROCESSEURS SOFTCORE

- Les performances dépendent :
 - Du circuit FPGA utilisé.
 - Du type de la mémoire utilisé.
- Les mesures ont été faites sur des cartes de développement Altera ou Xilinx.
- Mesures faites par le « De Nayer Instituut ».

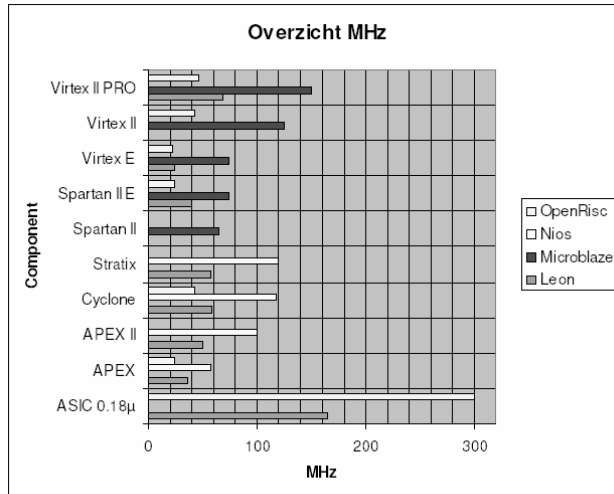


COS(Δ)

Linux et le System on Chip SoC



PERFORMANCES PROCESSEURS SOFTCORE



HOGESCHOOL VOOR WETENSCHAP & KUNST | DE NAYER INSTITUUT



COS

Linux et le System on Chip SoC



PERFORMANCES PROCESSEURS SOFTCORE

MicroBlaze™

Device Family	Timing	D-MIPS	MUL.	RAM	Cache [kB]	
					i	d
Virtex II 1000-4	100 MHz	62	HW	blockRAM	NA	NA
Virtex II 1000-4	100 MHz	45	SW	blockRAM	NA	NA
Virtex II 1000-4	50 MHz	4	HW	SDRAM	NA	NA
Virtex II 1000-4	50 MHz	6	HW	SDRAM	2	2
Virtex II 1000-4	50 MHz	13	HW	SDRAM	8	8

Nios™

Device Family	Timing	D-MIPS	RAM	Cache [kB]	
				i	d
Altera APEX20k200-2X	33 MHz	13	SRAM	0	0
Altera APEX20k1000E-2X	40 MHz	9	SDRAM	0	0
Altera Cyclone EP1C20	50 MHz	17	SRAM	4	4
Altera Cyclone EP1C20	50 MHz	15	SDRAM	4	4
Altera Stratix EP1S10	50 MHz	17	SRAM	4	4
Altera Stratix EP1S10	50 MHz	15	SDRAM	4	4

OpenRisc 1200

Device Family	Timing	D-MIPS	RAM
Virtex-E 1600	25 MHz	20	SRAM
Altera Cyclone	32 MHz	0.6	SDRAM

Leon SPARC™

Device Family	Timing	D-MIPS	RAM	Cache [kB]	
				i	d
Xilinx Virtex-E 1000E-6	25 MHz	21	SRAM	16	8
Altera APEX20k200-2X	26 MHz	21	SRAM	2	2
Altera APEX20k1000E-2X	20 MHz	4	SDRAM	8	8
Xilinx Virtex II 6000-5	40 MHz	27	SRAM	4	4
Altera Cyclone EP1C20	50 MHz	33	SRAM	4	4



COS

Linux et le System on Chip SoC



PARTIE 4 :

EXEMPLE DE MISE EN ŒUVRE :

PROCESSEURS SOFTCORE NIOS II SOUS μ Clinux



COS(Δ)

Linux et le System on Chip SoC



EXEMPLE : NIOS II/ μ Clinux

- Un exemple de *codesign* est donné en mettant en œuvre le processeur *softcore* NIOS II sous μ Clinux (Linux embarqué).
- La carte cible est la carte Altera Stratix 1S10.
- Exemple logiciel : serveur web embarqué.



COS(Δ)

Linux et le System on Chip SoC



EXEMPLE : NIOS II/μClinux IDE SYNTHÈSE

- L'IDE mis en œuvre pour la synthèse matérielle du processeur softcore NIOS II est Quartus II d'Altera.
- On est tributaire des outils de fondeurs de circuits FPGA.

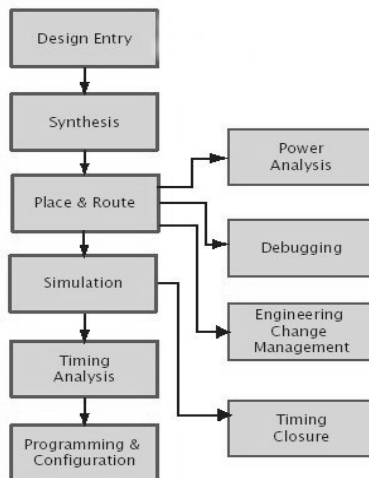


COS(Δ)

Linux et le System on Chip SoC



EXEMPLE : NIOS II/μClinux IDE SYNTHÈSE



COS(Δ)

Linux et le System on Chip SoC



EXEMPLE : NIOS II/μClinux

IDE SYNTHÈSE

Altera SOPC Builder - std - 1s10

File System Module View Tools Help

System Contents More "cpu" Settings System Generation

Altera SOPC Builder

- Interface to User ...
- Avion Modules
 - Nios II Processor - Altera C
- Bridges
- Communication
 - EP4K10K Nios Development
 - EP1K10K Nios Development I
 - EP2K10K Nios Development I
 - Ethernet
 - Extra Utilities
 - Lancelot
 - Legacy Components
 - Memory
 - Other
- Alto Modules
 - Bridges


Target: Nios Development Board, Stratix (EP1S10) Target Device Family: Stratix System Clock Frequency: 50 MHz

Use	Module Name	Description	Base	End	IO
<input checked="" type="checkbox"/>	cpu	Nios II Processor - Altera Corporation	0x00000000	0x00000000	
<input checked="" type="checkbox"/>	ext_ram_bus	Avalon Tri-State Bridge			
<input checked="" type="checkbox"/>	ext_flash	Flash Memory (Common Flash Interface)	0x00000000	0x00000000	
<input checked="" type="checkbox"/>	ext_ram	IDT71V416 SRAM	0x00000000	0x00000000	
<input checked="" type="checkbox"/>	onchip_ram_64_kbytes	On-Chip Memory (RAM or ROM)	0x00000000	0x00000000	
<input checked="" type="checkbox"/>	lan1c111	LAN91C111 Interface (Ethernet)	0x00010000	0x00010000	6
<input checked="" type="checkbox"/>	sys_clk_timer	Interval timer	0x00020000	0x00020000	0
<input checked="" type="checkbox"/>	jtag_uart	JTAG UART	0x00020020	0x00020020	1
<input checked="" type="checkbox"/>	button_pio	PIO (Parallel IO)	0x00020030	0x00020030	2
<input checked="" type="checkbox"/>	led_pio	PIO (Parallel IO)	0x00020040	0x00020040	
<input checked="" type="checkbox"/>	lcd_display	Character LCD (16x2, Optrix 1C207)	0x00020050	0x00020050	
<input checked="" type="checkbox"/>	high_res_timer	Interval timer	0x00020060	0x00020060	3
<input checked="" type="checkbox"/>	seven_seg_pio	PIO (Parallel IO)	0x00020080	0x00020080	
<input checked="" type="checkbox"/>	reconfig_request_pio	PIO (Parallel IO)	0x00020090	0x00020090	
<input checked="" type="checkbox"/>	uart1	UART (RS-232 serial port)	0x000200A0	0x000200A0	4
<input checked="" type="checkbox"/>	sysad	System ID Peripheral	0x000200B0	0x000200B0	
<input checked="" type="checkbox"/>	sdram	SDRAM Controller	0x01000000	0x01000000	
<input checked="" type="checkbox"/>	mouse	Interface to User Logic	0x000200C0	0x000200C0	5
<input checked="" type="checkbox"/>	keyboard	Interface to User Logic	0x000200D0	0x000200D0	7
<input checked="" type="checkbox"/>	vga	Lancelot VGA	0x000200E0	0x000200E0	

All Available Components

cpu was generated as a time-limited OpenCore Plus module and will time-out unless compiled in Quartus II with a valid license.
 Problem checking for web-updates (Unable to download component catalog, try again later.)


Exit < Prev Next > Generate



pk/pn/2005 v 1.0

Linux et le System on Chip SoC


- 51 -



EXEMPLE : NIOS II/μClinux

CARTE CIBLE


- Circuit FPGA Stratix II EP2S30F672C5.
- 1 Mo de SRAM 16 bits, 16 Mo de SDRAM 32 bits, 16 Mo de mémoire Flash.
- 1 support CompactFlash type I.
- 1 interface Ethernet 10/100 Mb/s.
- 2 ports série (RS-232 DB9).
- 1 connecteur JTAG.
- 4 boutons poussoirs, 8 leds utilisateurs.
- 2 afficheurs 7 segments.
- 1 afficheur LCD 2x16.



pk/pn/2005 v 1.0

Linux et le System on Chip SoC

- 52 -



EXEMPLE : NIOS II/ μ Clinux DEVELOPPEMENT LOGICIEL

- On utilise l'environnement de développement Eclipse.
- Le portage de μ Clinux est disponible pour NIOS II et est intégré à Eclipse.
- Les principales étapes sont :
 - Etape 1 : design de référence
 - Etape 2 : génération du noyau μ Clinux
 - Etape 3 : construction du système de fichiers root
 - Etape 4 : téléchargement et boot



COS

Linux et le System on Chip SoC



EXEMPLE : NIOS II/ μ Clinux DEVELOPPEMENT LOGICIEL

The screenshot shows the Quartus II Programmer window with the following details:

- Hardware Setup:** Mode: JTAG, Progress: 100%
- Table:**

File	Device	Checksum	Usercode	Program/Configure	Verify	Blank-Check	Examine
C:/altesa/klu/niios2/soa...	EP1K10K10	00711296	FFFFFFFF	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

OpenCore Plus Status: Click Cancel to stop using OpenCore Plus IP. Time remaining: unlimited.

Console:

- Info: Device 1 contains JTAG ID code 0x02001000
- Info: Configuration succeeded - 1 device(s) configured
- Info: Successfully performed operation(s)
- Info: Ended Programmer operation at Wed Nov 17 10:47:57 2004



COS

Linux et le System on Chip SoC



EXEMPLE : NIOS II/μClinux

```

SOPC Builder 4.1.0
uClinux/Nios 1
Altera Nios II support for 2501 Microprocessor: L.A.A. on '1'.
On node 0 totalpages: 4096
  DM zone: 0 pages, LIFO batch:1
  Normal zone: 4096 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
Built 1 zoneLists
Kernel command line: root=/dev/mtdblock0 ro
PID hash table entries: 16 (order 4: 128 bytes)
Memory available: 14512k/4096k RAM, 0k/0k ROM (1365k kernel code, 303k data)
Calibrating delay loop... 37.27 BogoMIPS
Dentry cache hash table entries: 2048 (order: 1, 8192 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
POSIX conformance testing by UNIFIX
NET: Registered protocol family 16
Serial: JTAG UART driver $Revision: 1.3 $
ttyJ0 at MMIO 0x80920020 (irq = 1) is a jtag_uart
RAMDISK driver initialized: 16 RAM disks of 4096k size 1024 blocksize
smc_probe: 75000 KHz Nios
SMSC LAN91C11 Driver (v2.1). (Linux Kernel 2.6)
eth0: SMSC LAN91C11FD (rev:1) at 0x80910300 IRQ:6 MEMSIZE:0192h NOWAIT:0 ADDR: 00:07:
ed:8a:07:2f
smc_probe: 75000 KHz Nios
Uniform Multi-Platform E-IDE driver Revision: 7.00alpha2
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
CF: cfi=1
Unable to initialize compact flash card. Please re-insert
Using anticipatory io scheduler
microtronixintcl: RAM probe address=0x200000 size=0x1ee000
Creating 1 MTD partitions on "RAM":
0x00000000-0x001ee000 : "ROMfs"
microtronixintcl: set ROMfs to be root filesystem
NET: Registered protocol family 2
IP: routing cache hash table of 512 buckets, 4kbytes
TCP: Hash tables configured (established 1024 bind 2048)
NET: Registered protocol family 1
NET: Registered protocol family 17
UFS: Mounted root (romfs filesystem) readonly.
Freeing unused kernel memory: 48k freed (0x1186000 - 0x1191000)

expand: from=/ramfs.ing to=/dev/ram0
expand: from=/ramfs.ing to=/dev/ram1

/etc/issue          www.microtronix.com          July 2004

Welcome to Linux on the Nios II
Nios2 login:
```



COS

© pk/pn/2005 v1.0

Welcome to Linux on the Nios II



- 55 -

EXEMPLE : NIOS II/μClinux SERVEUR WEB EMBARQUE

- Le but est d'embarquer un serveur web sur la carte cible pour piloter à distance des E/S de la carte cible.
- On embarque le serveur web *boa* (<http://www.boa.org/>).
- On développe des scripts CGI pour piloter les E/S. Les E/S considérées sont ici les 8 leds connectées au port IO *led_pio* du processeur softcore NIOS II.
- Un script CGI (application μClinux) est intégrée dans le système de fichiers root sous */home/httpd/cgi-bin* du serveur web *boa*.
- Une page HTML correspondant est aussi créée.



COS

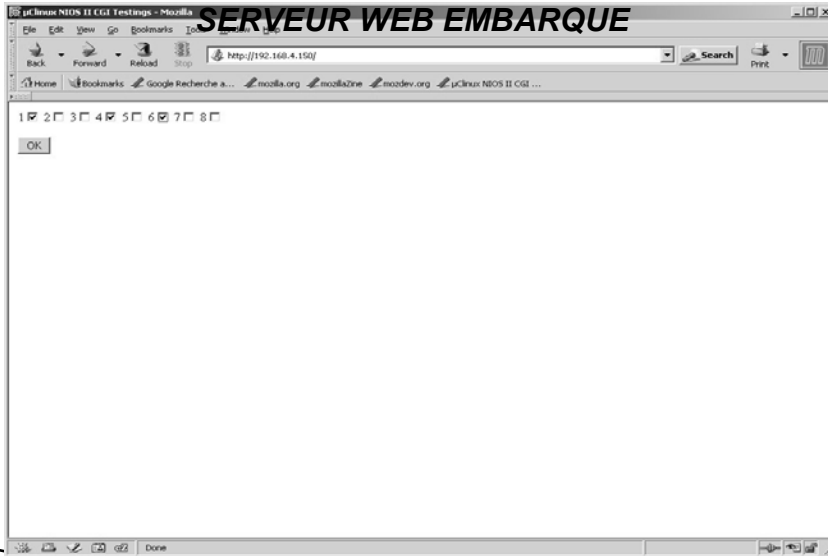
Linux et le System on Chip SoC

© pk/pn/2005 v1.0



- 56 -

EXEMPLE : NIOS II/μClinux



COS(Δ)

Linux et le System on Chip SoC



PARTIE 5 : CONCLUSION



COS(Δ)

Linux et le System on Chip SoC



CONCLUSION

- La conception des systèmes numériques complexes nécessite maintenant de mettre en oeuvre une nouvelle méthodologie de conception : le *codesign*.
- L'intégration grandissante sur le silicium a permis d'avoir une approche de conception orientée système et l'on développe maintenant des systèmes sur silicium SoC et SoPC.
- Les systèmes SoC et SoPC intègrent généralement un processeur embarqué si bien qu'il semble naturel d'embarquer Linux dans ces systèmes !



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 59 -



CONCLUSION

- Le *codesign* apparaît dans le développement conjoint matériel-logiciel pour la conception des systèmes numériques complexes :
 - On développera une partie de l'application sous forme d'un bloc IP (en VHDL par exemple) pour bénéficier d'une accélération matérielle du traitement. Cela revient en fait à développer un coprocesseur matériel spécifique.
 - On développera l'autre partie de l'application sous forme logicielle pour bénéficier de la souplesse de la « logique programmée » exécutée par le processeur softcore embarqué dans le système cible.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 60 -



CONCLUSION

- L'apport de Linux embarqué (μ Clinux) est indéniable pour différentes raisons :
 - On a un système de fichiers à disposition.
 - On a un système d'exploitation multitâche.
 - On peut réutiliser des briques logicielles issues du logiciel libre.
 - On peut intégrer ses modifications pour introduire des accélérations de traitement par matériel.



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 61 -



CONCLUSION

- Le lien entre le coprocesseur matériel spécifique et le processeur softcore exécutant le noyau μ Clinux sera établi en développant un driver Linux.
- Il est à noter que dans le cas de μ Clinux et en absence de MMU, on peut développer :
 - Un driver ou un module Linux.
 - Un driver dans l'espace utilisateur (*user space driver*). En absence de MMU, un processus Linux accède directement au matériel : on n'a donc pas de driver Linux à écrire mais on peut planter tout aussi facilement son système...



COS(Δ)

Linux et le System on Chip SoC

© pk/pn/2005 v 1.0

- 62 -



CONCLUSION

- La mise en oeuvre de Linux embarqué (μ Clinux) couplé à un processeur *softcore* (NIOS II d'Altera) a été présentée.
- Il est clair que le couple Linux embarqué-processeur *softcore* est une solution à explorer et à exploiter dès que l'on a un système numérique complexe à construire, ce qui offre de belles perspectives à notre système d'exploitation préféré...



COS()

Linux et le System on Chip SoC

