Méthodologie de conception pour FPGA

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

ReDS
Reconfigurable & Embedded Digital Systems

Etienne Messerli
Professeur institut REDS, HEIG-VD
etienne.messerli@heig-vd.ch
http://www.reds.ch

17 septembre 2009

Contenu de la présentation

- Evolution des PLDs
- Evolution des designs
- Méthodologies de conception
- Décomposition des systèmes numériques
  - ✓Spatiale, temporelle, pipeline
- Amélioration des performances, optimisation
- Cahier méthodologique VHDL
- Vérification des designs

Méthodologie de conception pour FPGA

Evolution des PLDs

Evolution "Circuits logiques programmables"

- Depuis les années 1995 :

  Formidable évolution des circuits logiques programmables

- Caractéristiques principales :
  - ✓densité ↗ inclu RAM, jusqu'à 38Mbits
  - ✓fréquence ↗ jusqu'à 650 MHz
  - ✓prix par *gate* ↘ 1996 : ≈ 1ct/gate
    2004 : ≈ 0,01ct/gate
    2006 : ≈ 0,0001ct/gate

Evolution PLD : caractéristiques …

- technologie 90 nm, 65 nm et 40 nm!
  - ✓ plusieurs milliards de transistors sur une puce
- 11 couches d'interconnections en cuivre
- multiples arbres d'horloges et PLL
  - ✓ global up to 32, local up to 96, PLL up to 16 (with 8 output)
- fréquence jusqu'à 650 MHz
  - ✓ FPGA Achronix jusqu'à 1.5 GHz !
- nombre de *gates* jusqu'à ~15M
- nombre de flip-flops jusqu'à 950K
- plusieurs Mbits de RAM, jusqu'à 38 Mbits

… évolution PLD : caractéristiques

- LUT à 6 entrées au lieu de 4! (Virtex5)
  ALM (2 LEs) à 8 entrées => LUT à 7 entrées (StratixIII et IV)
- blocs pré-câblés (DSP, ALM, SERDES…)
  - ✓jusqu'à 130 transceivers high speed (max 11 GigaBit/s)
  - ✓jusqu'à 1400 multiplicateurs 18x18
- Hard Core (PCI, PCI express, Ethernet MAC, …)
- multiples standards I/O: LVTTL, PCI, LVDS, …
- Nbr I/O jusqu'à 1'200
  - ✓boitier FBGA1932 pins => 1'100 I/O

Evolution PLD : prix aout 2009 …

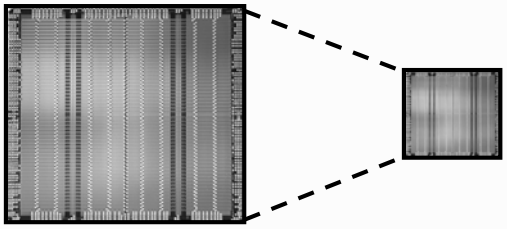
- PLDs de quelques francs à 11K francs !
- STRATIX-IV FPGA 530K, FBGA1152 : 11'400 $
  - ✓ 530K LEs, total RAM 27.3Mbits, 12 PLLs 10'600 Kgates
  - ✓ 98 transceiver, 1'024 Multiplier 18x18,904 I/O 0,001 $/gate
- *en mai 2008:*
  STRATIX-IV FPGA 150K, FBGA1152 : 11'000 $
  - ✓ *142K LEs, total RAM 7.3Mbits, 8 PLLs 2'850 Kgates*
  - ✓ *380 Multiplier 18x18,744 I/O 0,004 $/gate*
- CYCLONE-III FPGA 119K, FBGA780 : 502 $
  - ✓ 119K LEs, total RAM 3'888Kbits, 4 PLLs 2'380 Kgates
  - ✓ 288 Multiplier 18x18, 531 I/O 0,0002 $/gate
- CYCLONE-III FPGA 5K, EQF144 : 13 $
  - ✓ 5K LEs, total RAM 414Kbits, 2 PLLs 100 Kgates
  - ✓ 23 Multiplier 18x18, 94 I/O 0,0001 $/gate

… évolution PLD : prix aout 2009

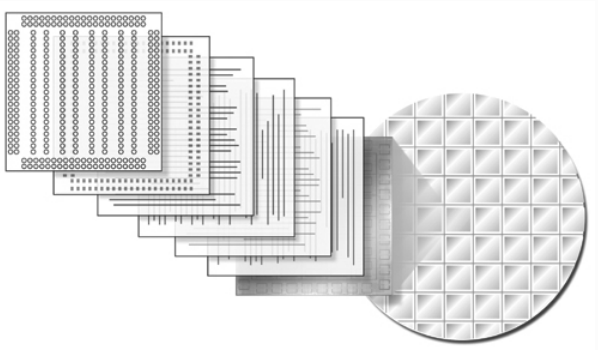
- MAX-II CPLD 2210 LE, FBGA256 32 $
  - ✓2210 LEs, Flash memory 8Kbits, 0 PLLs 44 Kgates
  - ✓0 Multiplier 18x18, 80 I/O 0,0007 $/gate
- MAX-II CPLD 240 LE, TQFP100 6 $
  - ✓240 LEs, total Flash memory 8Kbits, 0 PLLs 4.8 Kgates
  - ✓0 Multiplier 18x18, 212 I/O 0,001 $/gate

New Process Considerations

- **Benefits**
  - Higher Density
  - Higher Performance
  - Smaller Die Size=lower device cost

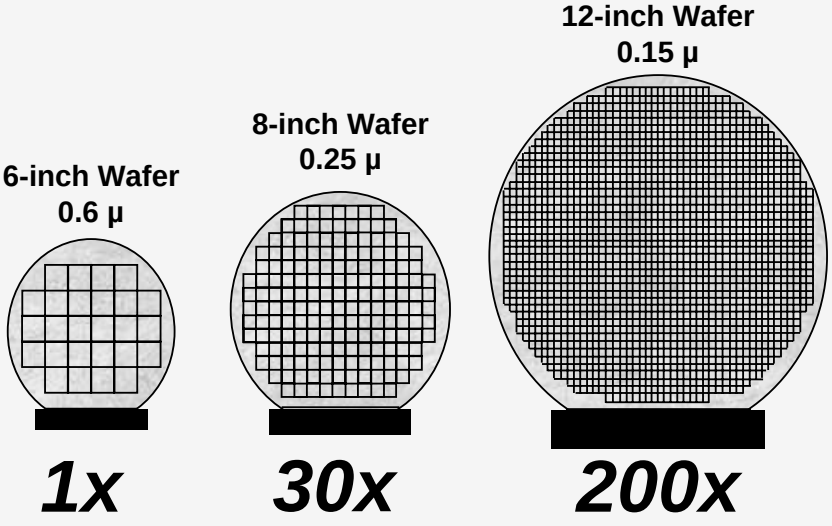


- **Drawbacks**
  - Higher Risks
  - Higher Invest Costs



Wafer/Die Sizes/Quantities

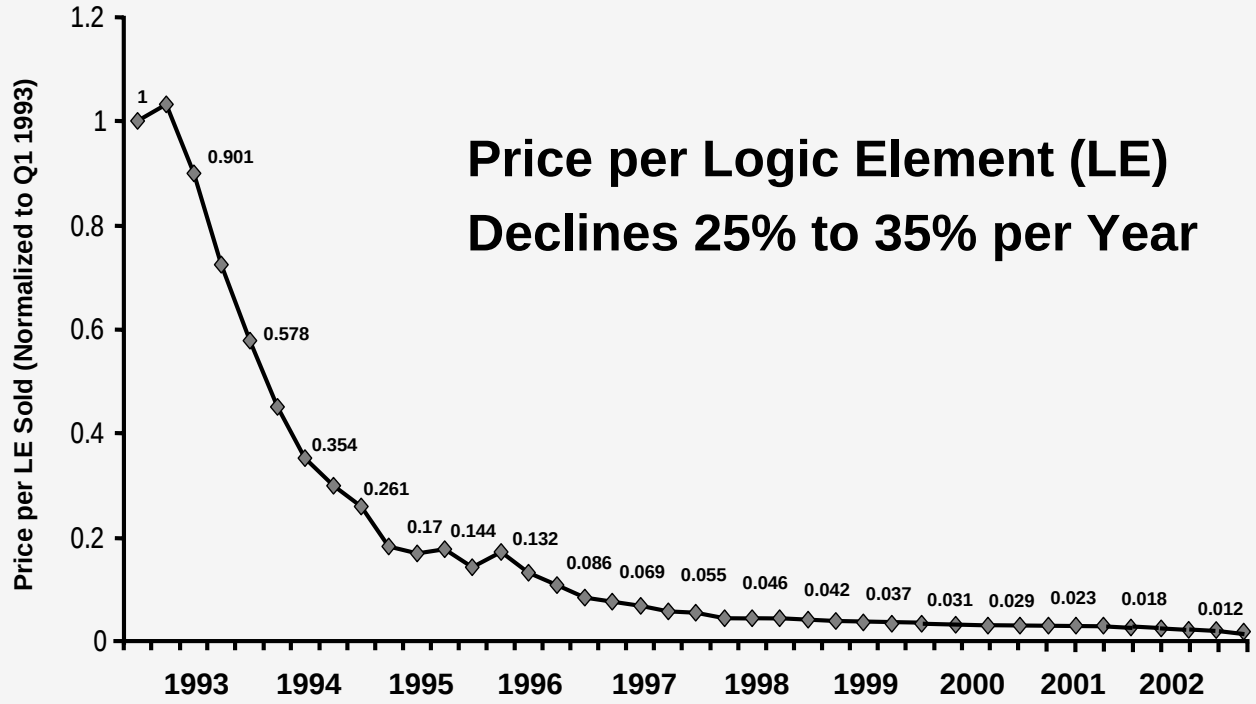
- **As Technology and Wafer Sizes Change,**
  We Get More Net Die Per Wafer "Yeald/Ausbeute"



Typical Net Die Per Wafer:

1x ***30x*** ***200x***

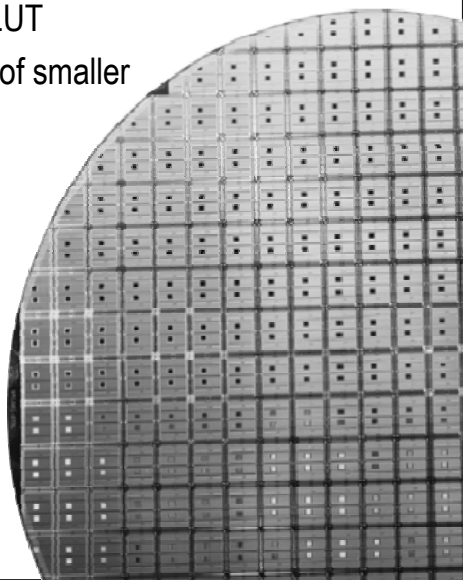
Price Trend



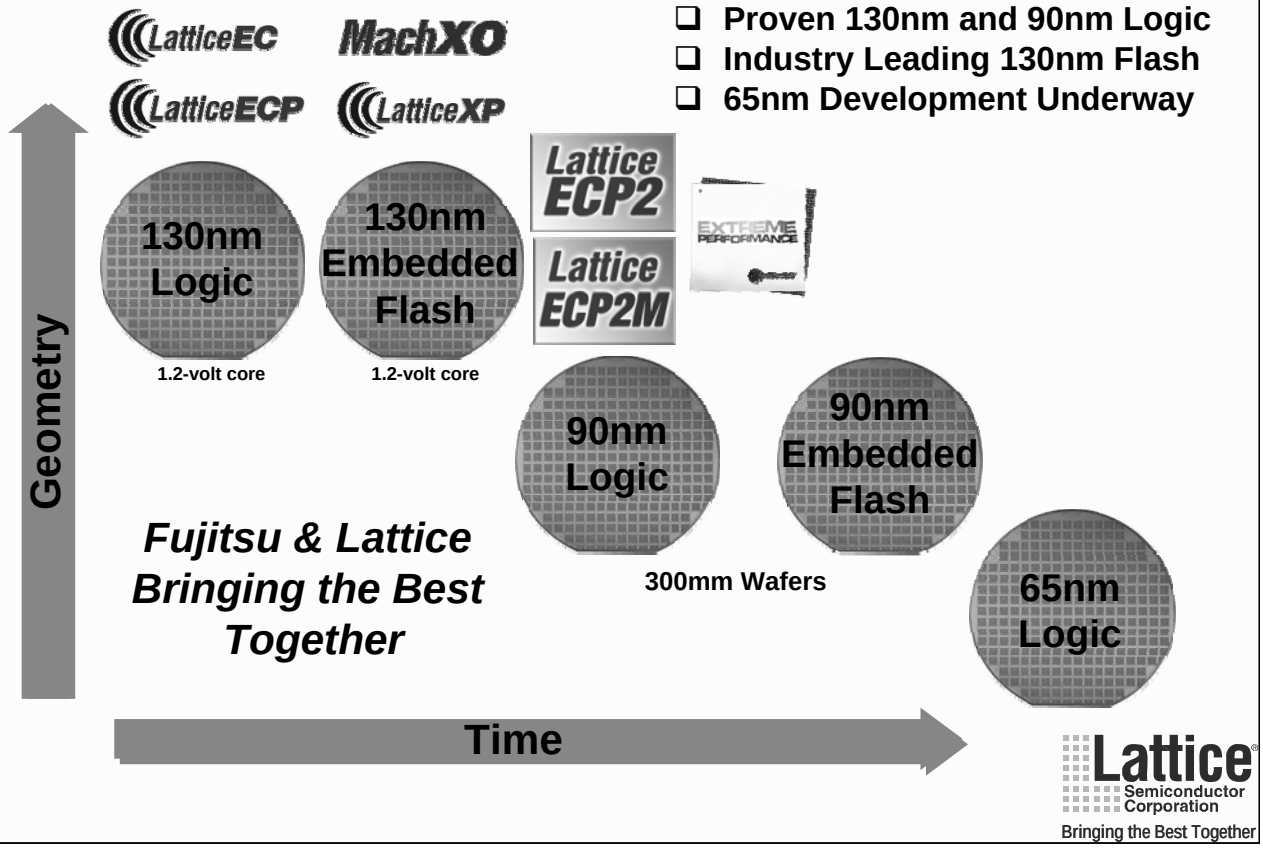
Price per Logic Element (LE)
Declines 25% to 35% per Year

New Process Technology

- Altera use 65-nm Process Technology for Stratix III FPGAs
  - 1.1 V or 0.9 V, 9-Layer-Metal, All Copper Process
- Stratix III device ALM has 8 inputs with a fracturable look-up table (LUT)
  - A full 6-input look-up table (LUT) or select 7-input LUT
  - Two independent outputs of multiple combinations of smaller LUT sizes for efficient logic packing
  - Implementing complex logic-arithmetic functions without additional resources
- Stratix III FPGA Performance (maximum)
  - Internal Clock Speed : 600 MHz
  - On-Chip RAM : 600 MHz
  - DSP Block : 550 MHz

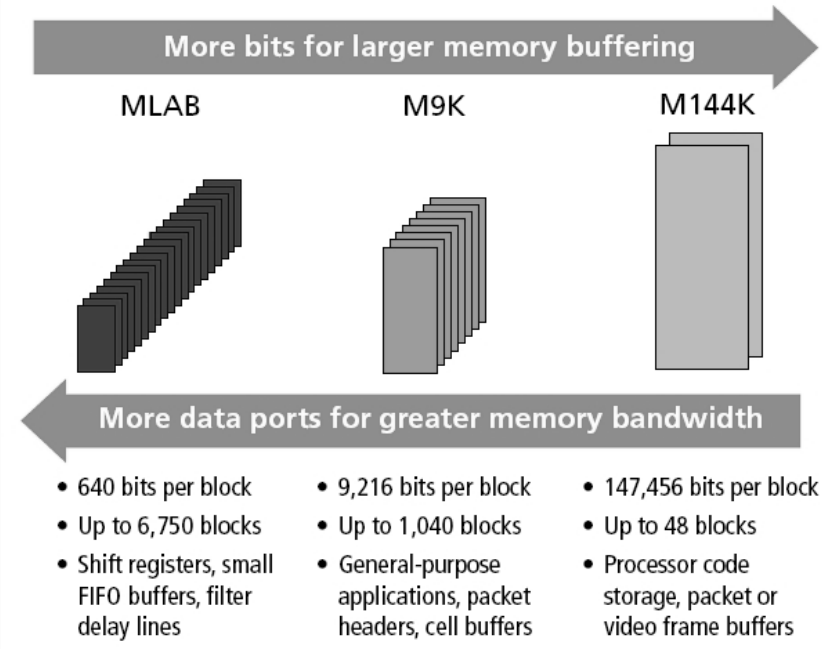


Lattice Technology Evolution



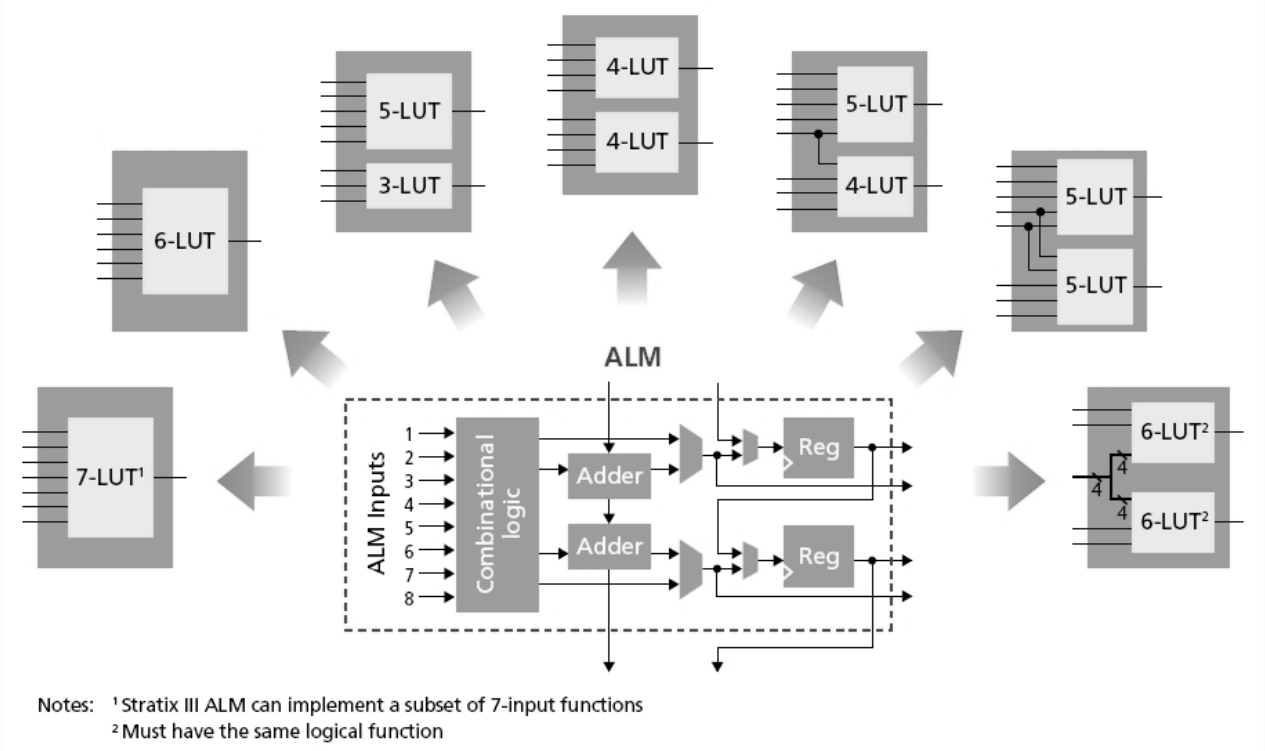
Lattice Semiconductor Corporation
Bringing the Best Together

TriMatrix memory



Stratix III, 2008

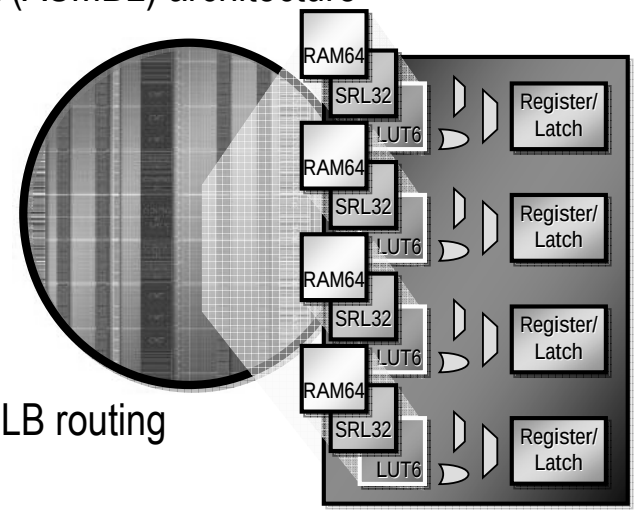
ALM block diagram example LUT configurations



Stratix III, 2008

Virtex-5 Logic Architecture

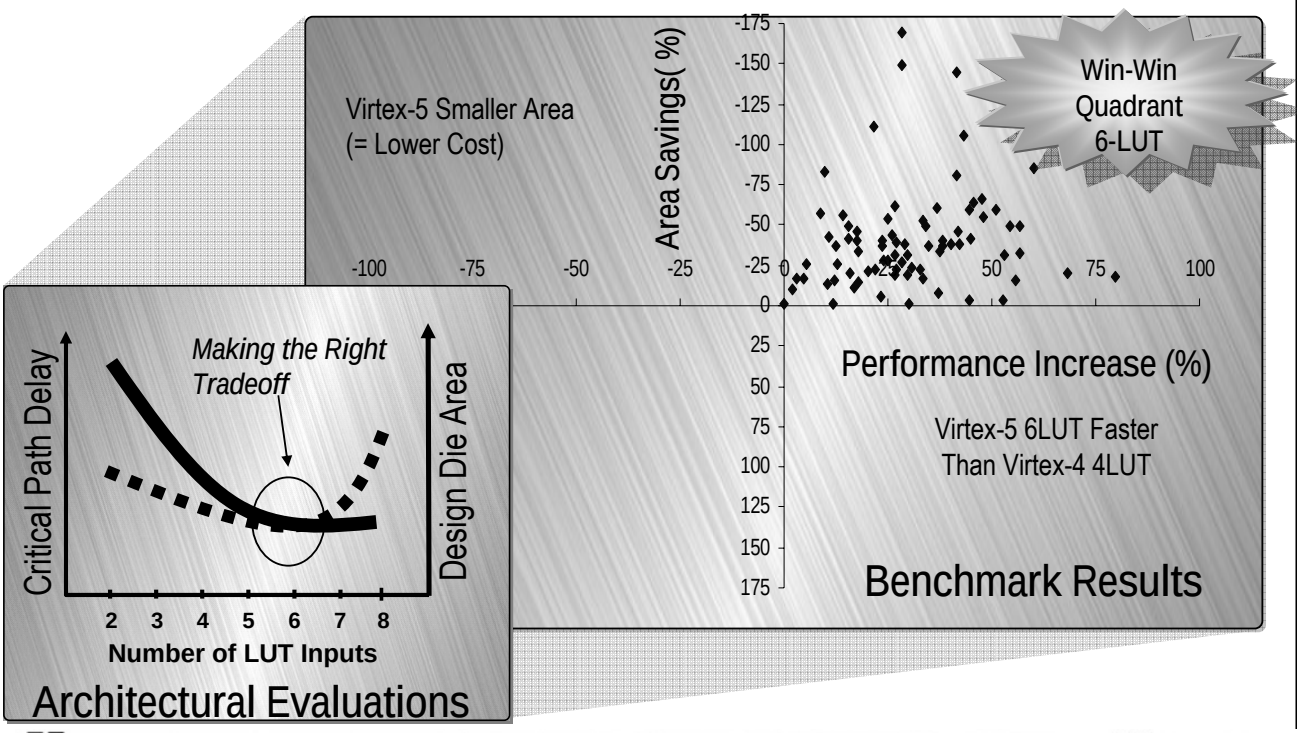
- Second-generation column-based Advanced Silicon Modular BLock (ASMBL) architecture
- Advanced logic structure
  - ✓ True 6-input LUTs
  - ✓ Exclusive 64-bit distributed RAM option per LUT
  - ✓ Exclusive 32-bit or 16-bit x 2 shift register
- More efficient and flexible inter-CLB routing
  - ✓ Increased performance



Virtex-5 is the flagship of the FPGA industry

Optimal Performance/Area Trade-off

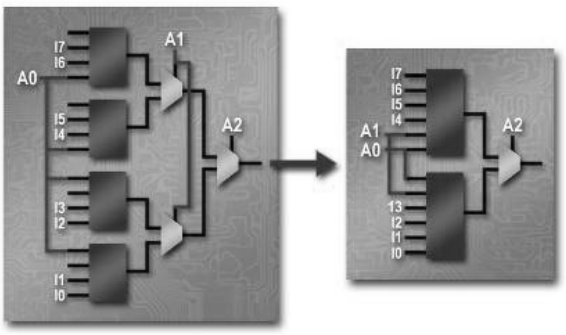
6-Input LUT Yields Best Results at 65nm



Logic Compaction with LUT6

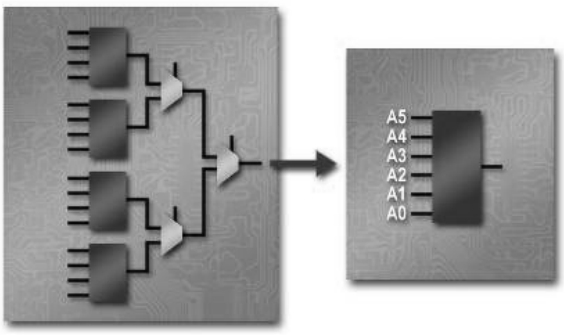
Use Fewer LUTs, Faster, Less Routing

8 to 1 Multiplexer



LUT4 LUT6

64 bit RAM



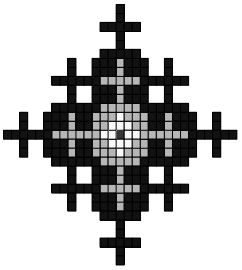
LUT4 LUT6

Interconnect Architecture

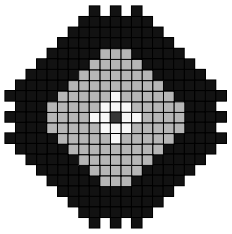
Reduce Routing Delay

- Diagonally symmetric interconnect pattern
- More logic reached per hop
  Fewer hops
  ↳ Shorter delay
  ↳ Faster design
- More symmetric pattern, same pattern for all outputs
  ✓ Better design tool Quality of Results (QoR)

Virtex-4 Interconnect Pattern



Virtex-5 Interconnect Pattern



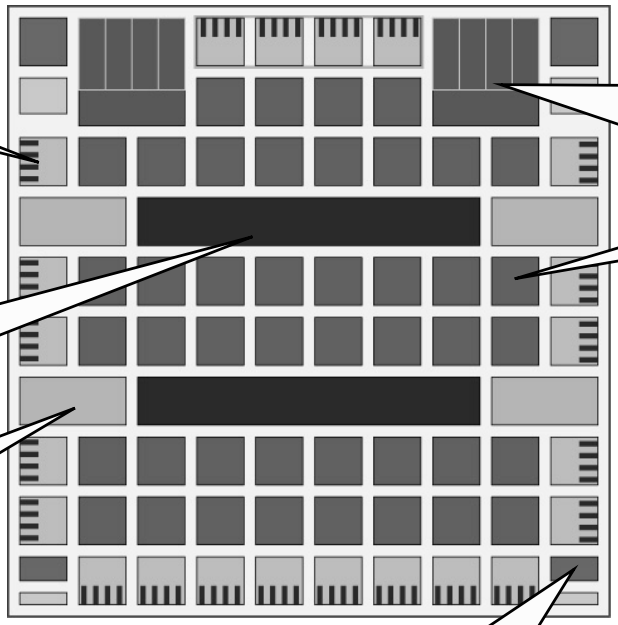
CLB

Direct
1 Hop
2 Hops
3 Hops

LatticeSC Architecture 500Mhz

AVNET Memec

High Performance FPGA Fabric



2Gbps PURESPEED I/O

4 to 32 SERDES (Up to 3.4Gbps) with Physical Coding Sublayer (PCS)

15K to 115K LUT4s

Up to 7.8 Mbits of Embedded Memory Blocks

System-Level Features: Embedded System Bus / Dedicated Microprocessor Interface / SPI Flash Configuration

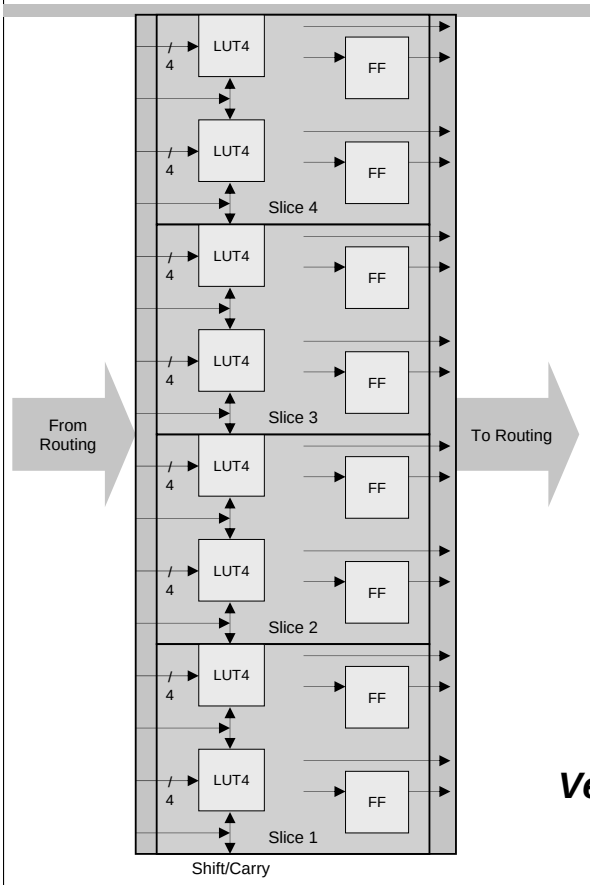
MACO: Embedded Structured ASIC Blocks (LatticeSCM Devices)

1.0V-1.2V Operating Voltage

8 Analog PLLs / 12 DLLs per Device

Lattice Semiconductor Corporation
Bringing the Best Together

Programmable Function Unit (PFU)



Features:

- Each slice contains two LUT4s and two FF/Latches
- Four slices per PFU
- Each slice is individually programmable
- Slices can be concatenated for longer functions
- PFUs can be concatenated for larger functions
- PFU Modes:
  - ✓Logic: LUT4, LUT5, LUT6, LUT7
  - ✓MUX: 2x1, 4x1, 8x1, 16x1, 32x1
  - ✓Ripple: 2-bit Adder, Subtracter, Counter, Comparator
  - ✓RAM/ROM: Single-port 32x1, 64x1, 128x1
  - ✓RAM/ROM: Dual-port 32x1, 64x1, 128x1
  - ✓Shift-Register: 8, 16, 32, or 64-bit

Versatile logic block running at 500MHz!

Lattice Semiconductor Corporation
Bringing the Best Together

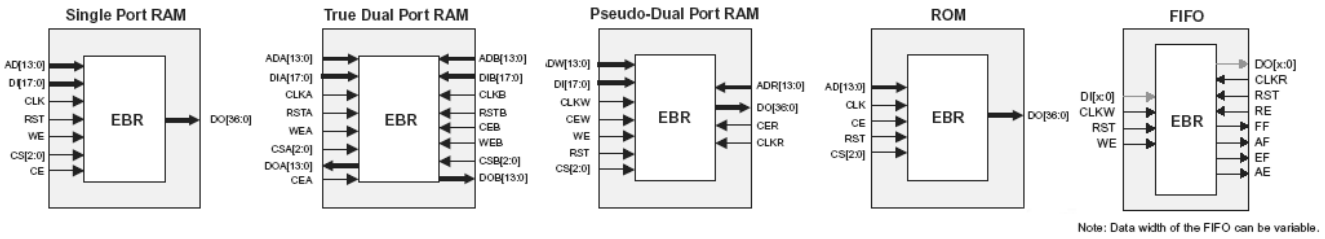
Programmable Function Unit (PFU)

Features:

- Each sysMEM block provides 18K bits
- 500 MHz Operation
- Registered inputs, registered outputs
- RAM can be pre-loaded during device initialization
- RAM may have different Read & Write widths

LatticeSC Embedded Memory					
Device	SC15	SC25	SC40	SC80	SC115
sysMEM Blocks (18Kb)	56	104	216	308	424
sysMEM (Mbits)	1.03	1.92	3.98	5.68	7.8

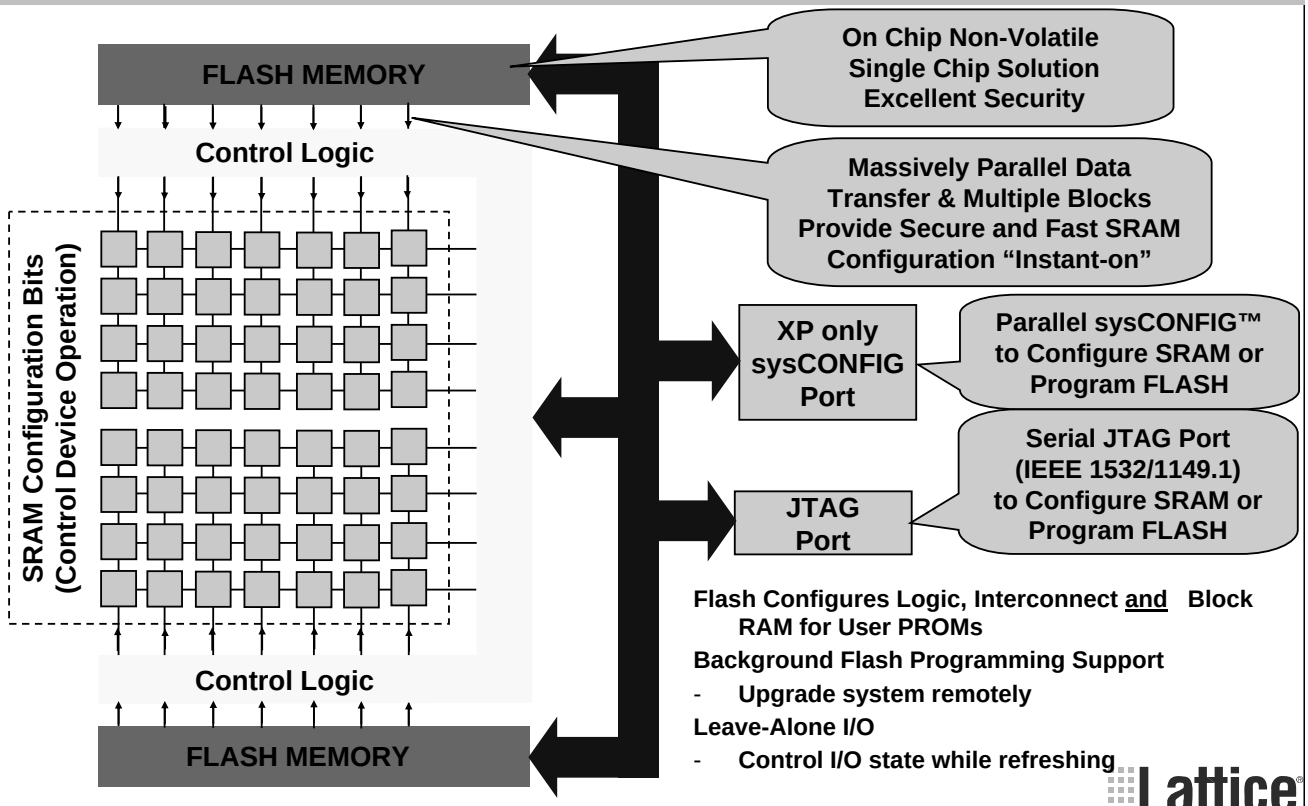
Flexible Configuration Capabilities:



Large Memory Capacity to Support Fast, Flexible RAM for Packet Buffering, Clock Domain Transfers, and Context Memory

Lattice Semiconductor Corporation
Bringing the Best Together

Programmable Function Unit (PFU)



Flash Configures Logic, Interconnect and Block RAM for User PROMs

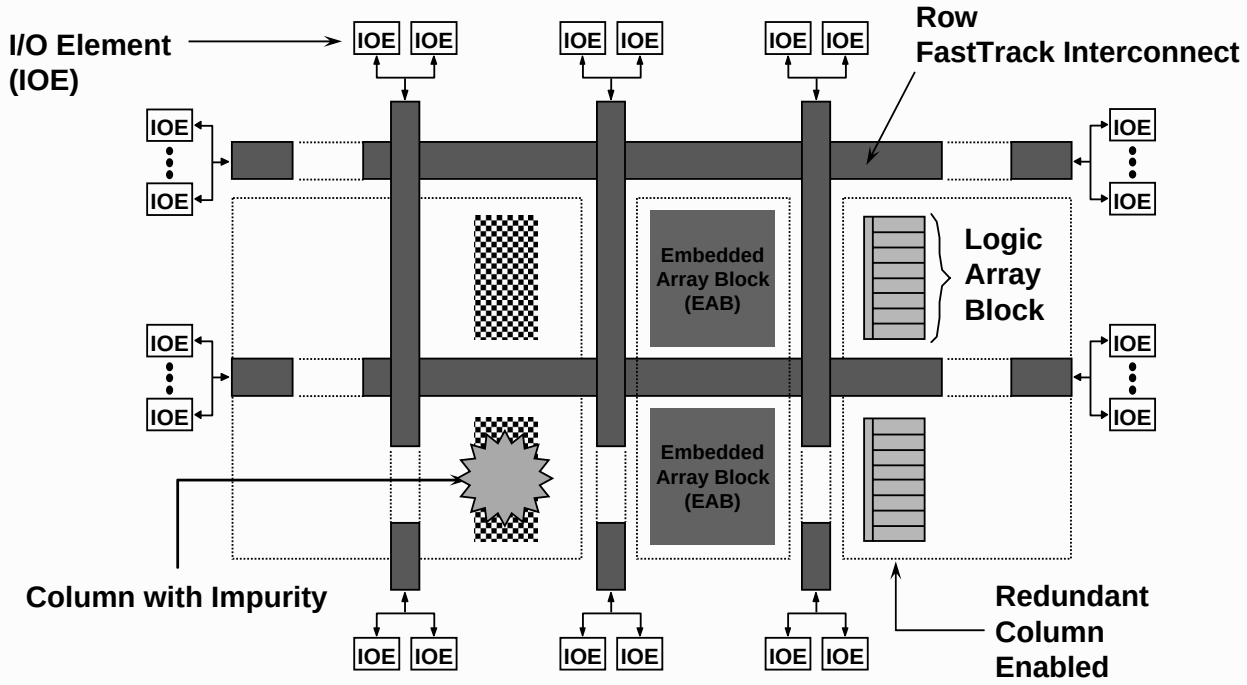
Background Flash Programming Support

- Upgrade system remotely

Leave-Alone I/O

- Control I/O state while refreshing

Redundancy: Repaired Die



FPGA Altera: Stratix IV

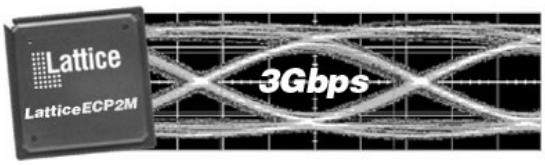
- Technologie 40 nm
- Stratix IV GX devices (FPGAs with transceivers):
  - ✓Up to 530K logic elements (LEs), 424'960 DFF,
    48 full-duplex CDR-based transceivers at up to 8.5 Gbps,
    98 High-Speed LVDS SERDES at up to 1.6 Gbps
- Stratix IV GT: 48 transceiver up to 11.3 Gbps
- Stratix IV E devices (enhanced FPGAs):
  - ✓Up to 680K LEs, 544'880 DFF, 31.5-Mbit RAM,
    1'360 18 x 18-bit multipliers, 1104 I/Os

FPGA Achronix

- Caractéristiques:
  - ✓fréquence: 1,5 GHz
  - ✓Synchronous frame surrounding a picoPIPE logic fabric
  - ✓Design is pipelined automatically
  - ✓40 lanes of 10 Gbps SerDes (family Speedster™)
- Applications *high speed:*
  - ✓Internal processing bandwidth
  - ✓Input/Output bandwidth
  - ✓External memory bandwidth

Info site "www.achronix.com", janvier 2009

FPGA avec SERDES block

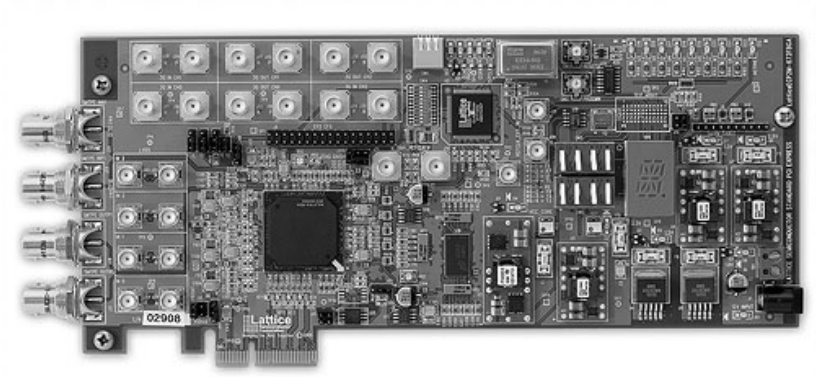


The LatticeECP2™ and LatticeECP2M™ families redefine the low-cost FPGA category, with more of the best FPGA features for less. By integrating features and capabilities previously only available on higher cost / high performance FPGAs, these families expand the range of applications that can take advantage of low cost FPGAs.

Info : site Lattice Semiconductor, décembre 2006

Carte de développement & FPGA

- LatticeECP2M PCI Express x4 Evaluation Board

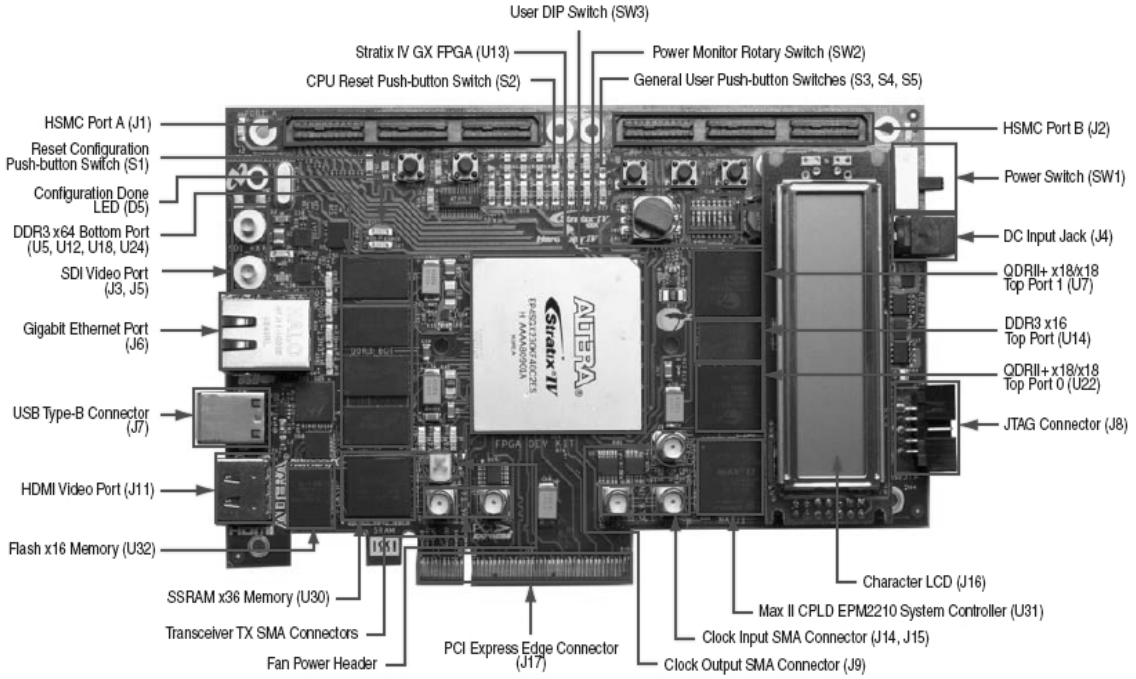


Décembre 2006

  - LFE2M35E-6F672C FPGA Device
  - x4 PCI Express edge connector
  - On-board DDR2 Memory
  - SMA connectors for SERDES I/O, LVDS evaluation

Carte de développement & FPGA

Figure 2–1. Overview of the Stratix IV GX FPGA Development Board Features



Technologie des PLDs

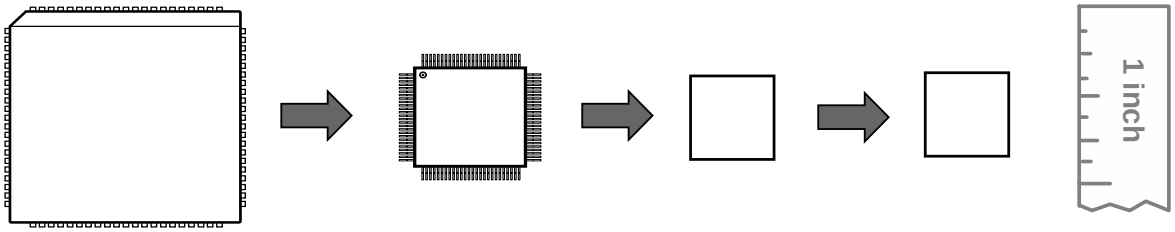
- Les PLDs utilisent de nombreuses technologies différentes :
  - ✓ Volatile ⇒ SRAM
  - ✓ Reprogrammable ⇒ EEPROM
    EEPROM Flash
    SRAM + EEPROM intégrée } nouveau
  - ✓ Programmable une seule fois ⇒ Anti-fuse, Via-link

Configuration des PLDs

- Majorité PLD programmables sur la carte
  - ✓ISP : In-System Programming
  - ✓via connexion JTAG
  - ✓Possibilité de reconfiguration dynamique
- Anti-fuse nécessite un programmateur
  - ✓très grande sécurité

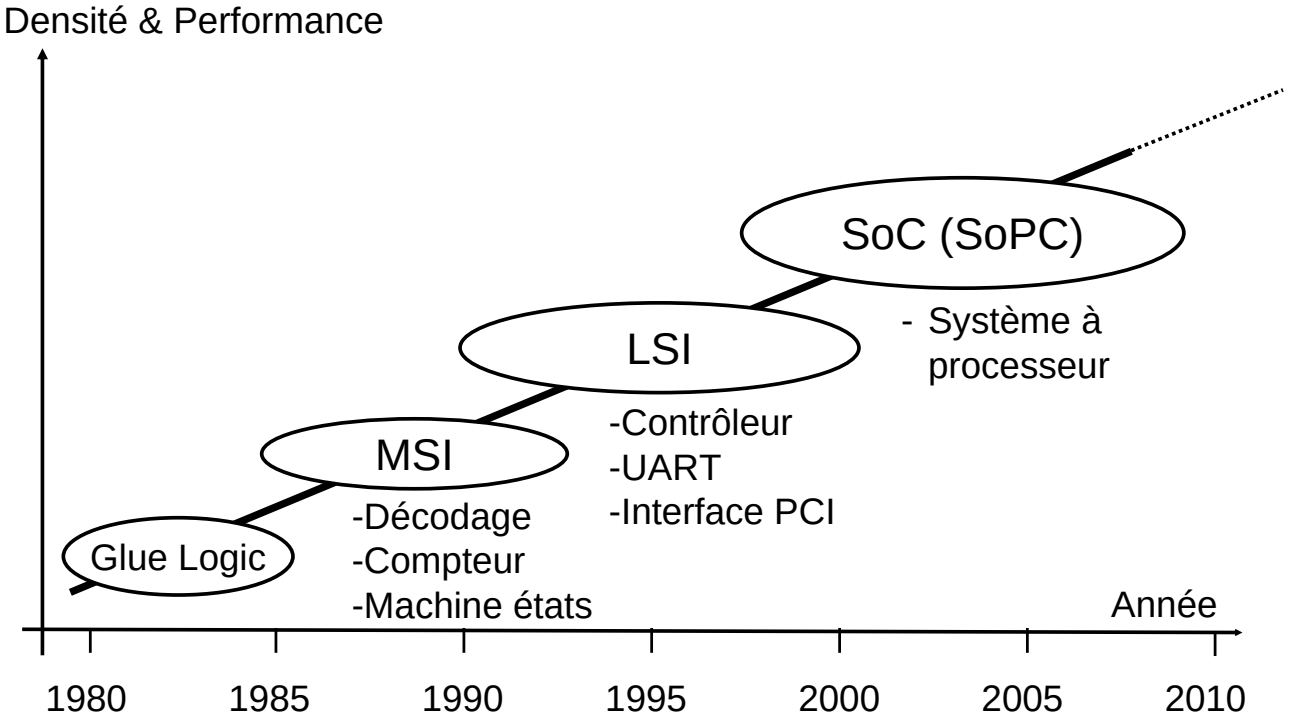
Evolution des boitiers

SOIC	Small Outline Integrated Package	8 à 24 pins (doc Altera)
PLCC	Plastic J-Lead Chip Carrier	20 à 84 pins
PQFP	Plastic Thin Quad Flat Pack	100 à 144 pins
BGA	Ball-Grid Array	env 100 pins à .. (†)
FBGA	FineLine Ball-Grid Array	100 à 1760 pins
UBGA	Ultra FineLine Ball-Grid Array	49 à 169 pins

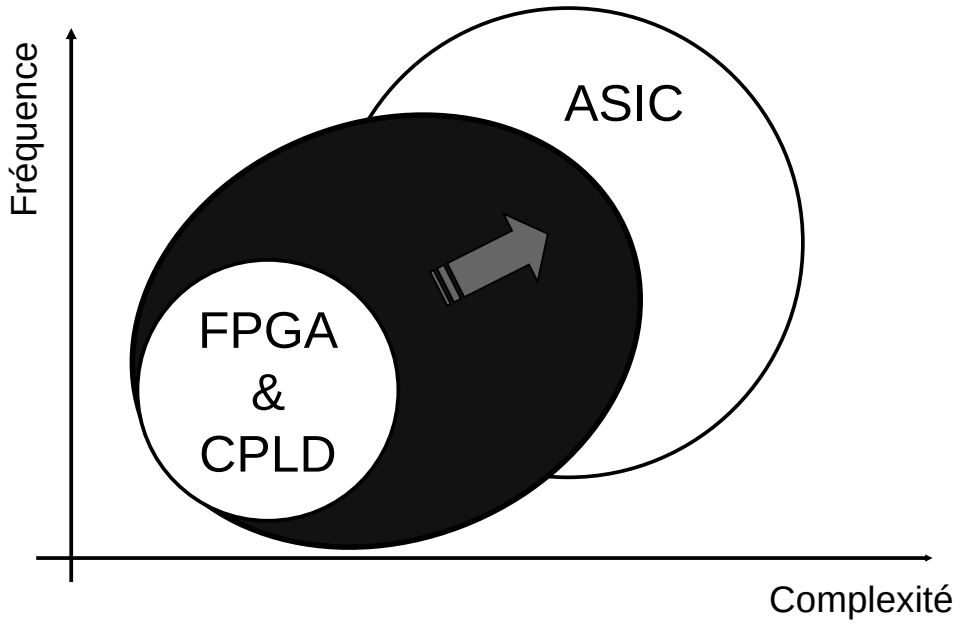


	84-Pin PLCC	100-PQFP	100-FBGA	169-UBGA
Surface relative	100%	23%	12%	5%

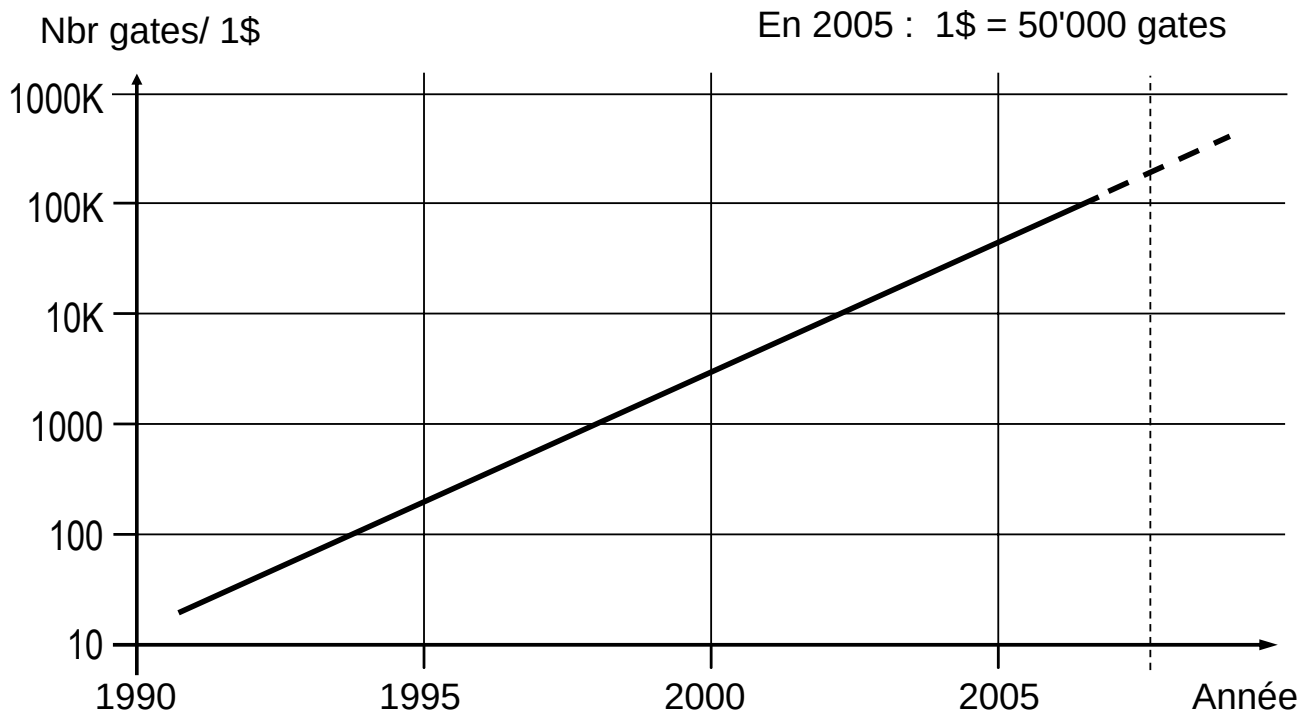
Domaine d'utilisations des PLDs



Comparaison : FPGA - ASIC



Prix des PLDs low cost (volume price!)



Ancien CPLD : MAX 7000S

- Technologie EEPROM
- EPM7128SLC84-10
  - ✓ $ 30 (-15 $19)
  - ✓ 68 IOs
  - ✓ 128 MacroCells
  - ✓ 128 DFF
  - ✓ env. 2'500 gates
  - ✓ 240 MHZ
  - ✓ prix ~ 0,01 $/gate

Nouveau PLD : MAX II

- Technologie Flash
- FPGA vendu comme CPLD !
- EPM1270T144C4
  - ✓ $33 (C5 $25)
  - ✓ 116 IOs
  - ✓ 1270 LEs
  - ✓ 1270 DFF
  - ✓ env. 20'000 gates
  - ✓ 8Kbits memory
  - ✓ 300 MHz
  - ✓ prix ~ 0,0015 $/gate

Nouveau PLD : MAX II

- Technologie Flash
- FPGA vendu comme CPLD !
- EPM240GT100C5
  - ✓ 6.6 $
  - ✓ 80 IOs
  - ✓ 240 LEs
  - ✓ 240 DFF
  - ✓ env. 3'800 gates
  - ✓ 8Kbits flash memory
  - ✓ 300 MHz
  - ✓ prix ~ 0,0017 $/gate

Prix Cyclone II, ALTERA

- Info de février 2006
- Cyclone II 400K gates EP2C35
  - ✓Volume price :
    $22 pièce
    prix ~ 0.00005 $/gate
  - ✓EP2C35F484C
    1 pièce $91 (oct 2007)
    soit 4 x "volume price"

Table 1. Low-Cost ASIC Replacement Solution Overview

	Cyclone	Cyclone II	HardCopy Stratix
ASIC Gates *(1)*	35K to 240K	55K to 820K	308K to 950K
Total RAM Bits	59,904 to 294,912	119,808 to 1,152,000	1,944,576 to 5,658,048
User I/O	104 to 301	142 to 622	473 to 773
Number of Family Members	5	6	5
Core Voltage	1.5 V	1.2 V	1.5 V
Process Technology	130-nm	90-nm	130-nm
Embedded 18 x 18 Multipliers	None	13 to 150	10 to 22 *(3)*
Volume Price *(2)*	$12 for 150K Gates	$22 for 400K Gates	Contact Sales Representative

Notes:

1. The ASIC gate count does not include embedded RAM or multipliers. Adding these will increase the total ASIC gate count.
2. Cyclone series FPGA volume price based on 250,000-unit quantities today. Contact your local sales representative or distributor for additional pricing information.
3. HardCopy Stratix implements multipliers in digital signal processing (DSP) blocks.

Prix Cyclone III, ALTERA

- Info octobre 2007
- Cyclone III, EP3C25
  - ✓ 24,624 LEs
  - ✓ 66 multiplicateurs 18x18
  - ✓ 594Kbits de RAM
  - ✓ 4 x PLLs
- EP3C25F324C8 (FBGA)
  1 pièce $49
  prix ~ 0.0001 $/gate

Table 1. Cyclone III FPGA Overview

Device	EP3C5	EP3C25	EP3C40
Logic Elements	5,136	24,624	39,600
M9K Embedded Memory Blocks *(1)*	46	66	126
Total RAM (Kbits)	414	594	1,134
Embedded 18-bit x 18-bit Multipliers	23	66	126
PLLs	2	4	4
Maximum User I/O Pins	182	215	535
Differential Channels	70	83	227
Availability (Commercial Grade)	Buy Now	Buy Now	Q4 2007

Prix Arria GX, ALTERA

- Arria GX, EP1AGX35
  - 33,520 LEs
  - 56 multiplicateurs
  - 1'317Kbits de RAM
  - 4 x PLLs
  - 4 x transceiver high speed
- EP1AGX35CF484-C6 FBGA
  - 1 pièce $122 prix octobre 2007
  - 1 pièce $230 prix janvier 2009
  - prix ~ 0.0003 $/gate

Table 2. Arria GX Device Family

Device	EP1AGX20	EP1AGX35	
Transceiver Channels	4	4	8
Equivalent LEs (1)	21,580	33,520	
Total Memory Bits	1,229,148	1,348,416	
18 x 18 Multipliers	40	56	
PLLs (2)	4	4	
Availability (3)	Available Buy Now	Available Buy Now	Available Buy Now

Prix Arria GX, ALTERA

- Info octobre 2007
- Arria GX
  - EP1AGX35
    - 33,520 LEs
    - 56 multiplicateurs
    - 1'317Kbits de RAM
    - 4 x PLLs
    - 4 x transceiver high speed
  - EP1AGX35CF484-C6 FBGA
    - 1 pièce $122

Table 2. Arria GX Device Family

Device	EP1AGX20	EP1AGX35	
Transceiver Channels	4	4	8
Equivalent LEs (1)	21,580	33,520	
Total Memory Bits	1,229,148	1,348,416	
18 x 18 Multipliers	40	56	
PLLs (2)	4	4	
Availability (3)	Available Buy Now	Available Buy Now	Available Buy Now

Prix Arria II GX, ALTERA

- Info septembre 2009
- Arria II GX, 40 nm
  - EP2AGX45
    - 33,520 LEs
    - 56 multiplicateurs
    - 1'317Kbits de RAM
    - 4 x PLLs
    - 4 x transceiver high speed
  - EP2AGX45….
    - pas de prix sur site Altera !

Table 1–1. Arria II GX Device Features (Part 1 of 2)

Feature	EP2AGX30	EP2AGX45	EP2AGX65
ALMs	10,800	18,050	25,300
LEs	27,000	45,125	63,250
PCI Express hard IP Blocks	1	1	1
M9K Blocks	144	319	495
Total Embedded Memory in M9K Blocks (Kbits)	1296	2871	4455
Total On-Chip Memory (M9K + MLABs) (Kbits)	1,634	3,435	5,246
Embedded Multipliers (18 × 18)	128	232	312
General Purpose PLLs	4	4	4
Transceiver Tx PLLs *(2)*	2	2 or 4 *(1)*	2 or 4 *(1)*

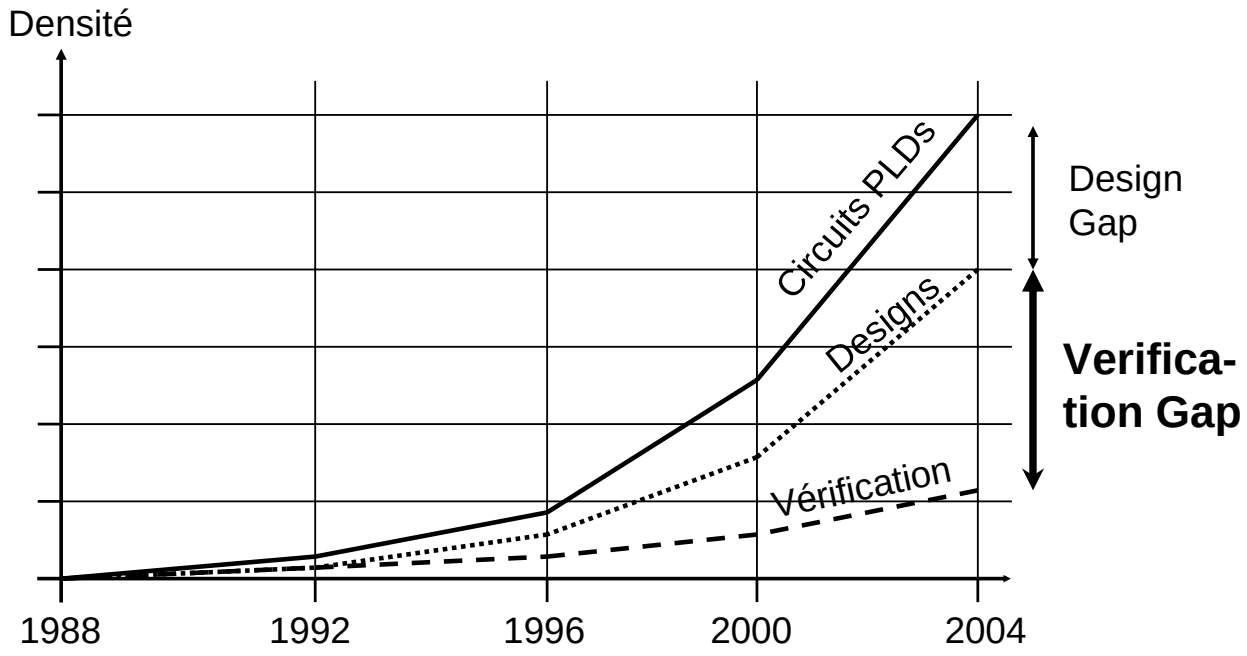
Méthodologie de conception pour FPGA

Evolution des designs

Complexité et coûts...

- Nouveau développement:
  - ✓PLD indispensable (CPLD, FPGA, HardCopy, ASIC)
- Niveau de complexité augmente rapidement :
  - ✓dizaines vers centaines de milliers de portes, voir millions de portes
- Développement "à la main" trop difficile et trop coûteux :
  outils informatiques performants, méthodologies et bibliothèques de "pièces" sont indispensables

Défi des PLDs ...



… défi des PLDs

- La capacité disponible des circuits progresse plus vite que les designs réalisés.
- Raccourcir les temps de développement :
  - ✓langage de haut niveau : description fiable et efficace
  - ✓outils moderne et performant
  - ✓réutiliser les descriptions
  - ✓Utilisation de bloc IP (Intellectual Property)
- Garantir la vérification des designs

Vérification : le nouveau défi …

- Dépannage d'un PLD programmé :

  très coûteux voir impossible
- Simulation indispensable,

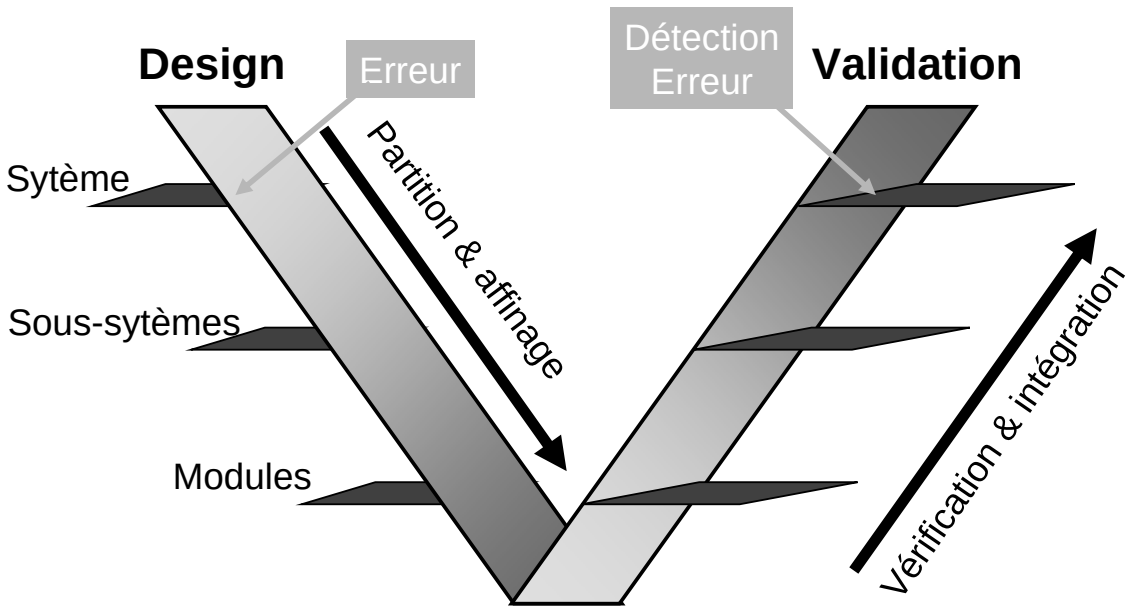
  nécessite : outils de vérification performants
  langages adaptés

Nouveau défi: Vérification des designs pour PLDs

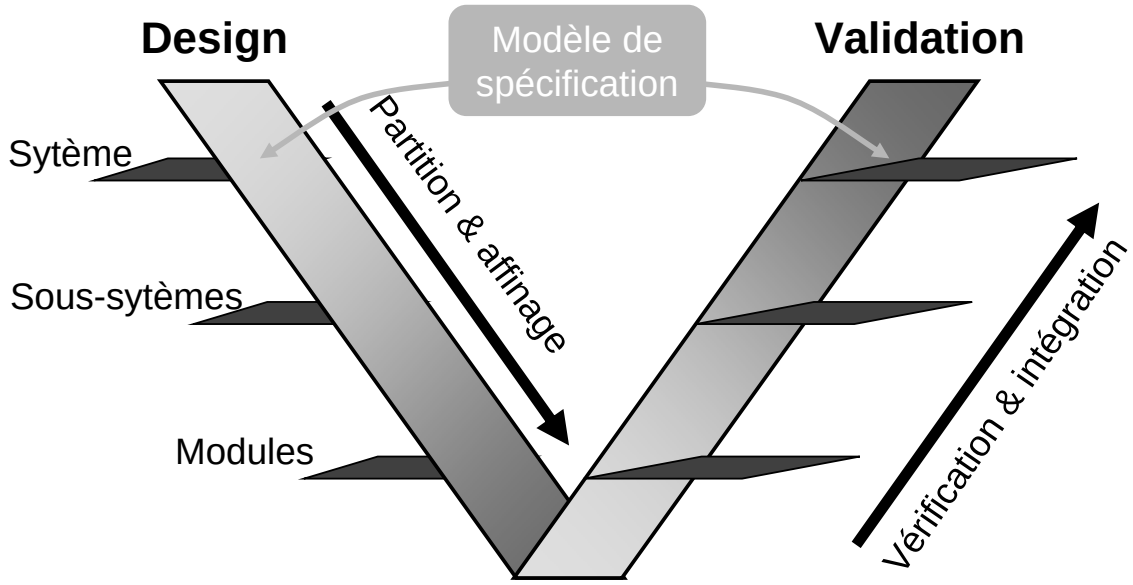
… vérification : le nouveau défi …

- Vérifier conformité avec le cahier des charges
- Maîtriser les coût (durée) de la vérification
  - ✓représente env. 70% du temps !
- Détecter les erreurs le plus rapidement possible durant le développement
- Assurer fonctionnement après intégration

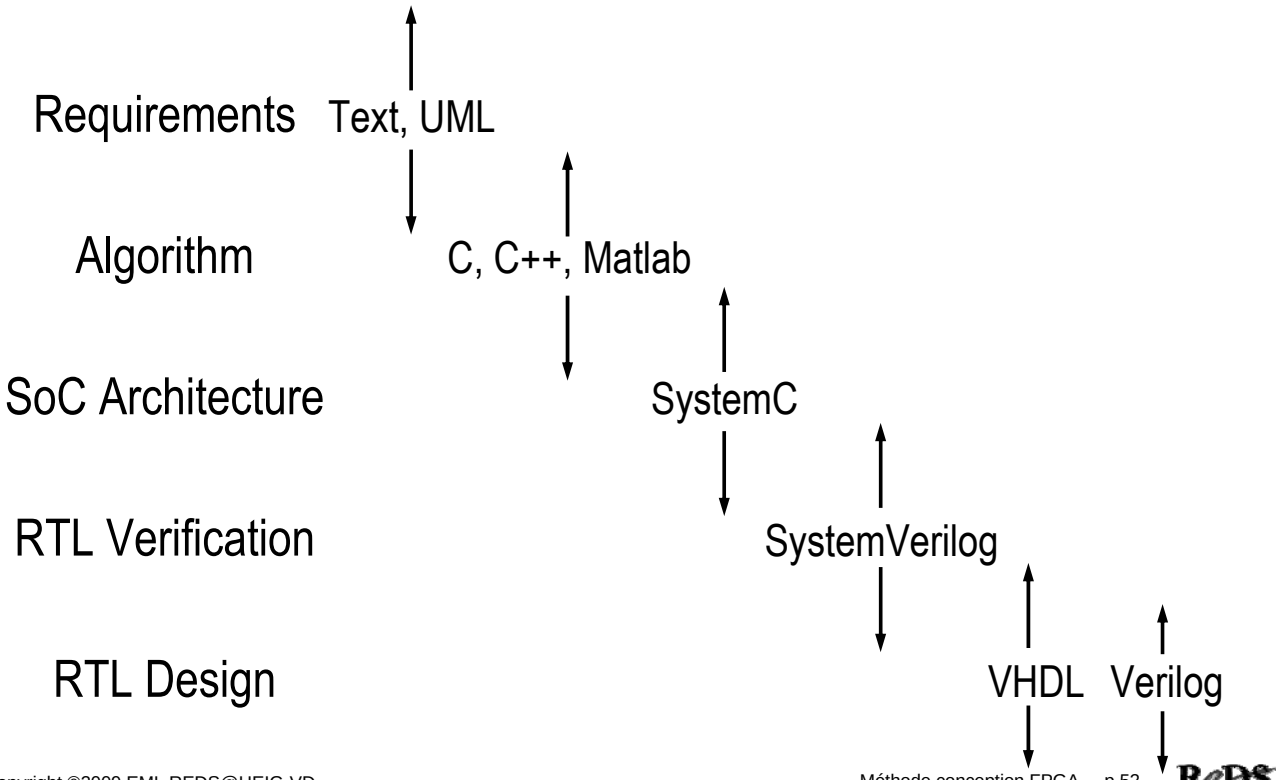
… vérification : le nouveau défi …



… vérification : le nouveau défi



Design and verification languages



Langages pour la synthèse (Design)

Pas de changement de langage pour la synthèse.

Indispensable: méthodologie et design re-use

- Le langage VHDL est très performant
  - ✓ hiérarchique, paquetage, générique, attributs, …
- Utilisation parfois nécessaire de Verilog pour les IPs
- Dans le cas de l'utilisation du langage Verilog
  - ✓ nécessité d'utiliser des fonctionnalités du SystemVerilog

Langages pour la vérification

Projets de petites à moyennes tailles:

- ✓Utilisation du langage VHDL

Projets de tailles moyennes à élevés, nécessité :

- ✓modèles décrits à très haut niveau (transaction level)
- ✓génération de stimuli aléatoires
- ✓ vérifications par assertions

Solution actuelle: SystemC, PSL (assertions), ..

Nouveau langage: SystemVerilog

- ✓programmation OO (classes), random, assertions, …

Méthodologie avec SystemVerilog

- OVM, Open Verification Methodology
  - ✓http://www.ovmworld.org/
  - ✓Mise en commun des méthodologies de Cadence et Mentor Graphic
  - ✓Cadence: URM, Universal Reuse Methodology
  - ✓Mentor Graphics: AVM, Advanced Verification Methodology
- VMM, Verification Methodology Manual
  - ✓http://www.vmm-sv.org/
  - ✓ARM & Synopsys

Simulateur SystemVerilog

Simulateurs: VHDL, Verilog, SystemVerilog, SytemC

- ✓Incisive, Cadence
- ✓QuestaSim, Mentor Graphics (évolution de ModelSim)
- ✓VCS, Synopsys
- ✓Active HDL & Riviera, Aldec
- ✓MPSim, AXIOM Design Automation

  →support pas VHDL !

Vérification: nouvelle voie

- Nusym promises high coverage with "intelligent verification"
  - ✓http://www.nusym.com/
  - ✓http://en.wikipedia.org/wiki/Nusym_Technology
  - ✓Petite société qui développe de nouvelle solution pour réaliser des vérifications "intelligentes"

Méthodologie de conception pour FPGA

Méthodologie de conception

Préambule

- Présentation orientée vers les FPGAs et CPLDs
- Les méthodologies, les règles, ... sont valables pour des designs intégrés dans un PLD.
- Règle principale : conception **Full** synchrone

- Dans le cas de réalisation avec un ASIC, de design asynchrone, etc.. les méthodologies, les règles, .. à utiliser peuvent être différentes

Complexité ?

Qu'est-ce que la complexité d'un design ?

- Complexité ≠ Densité

Quelques facteurs aggravants :

- Asynchronisme
- Spécification incomplète
- Proximité des limites technologiques
- Traitement numériques complexes

Réussir la conception des FPGA

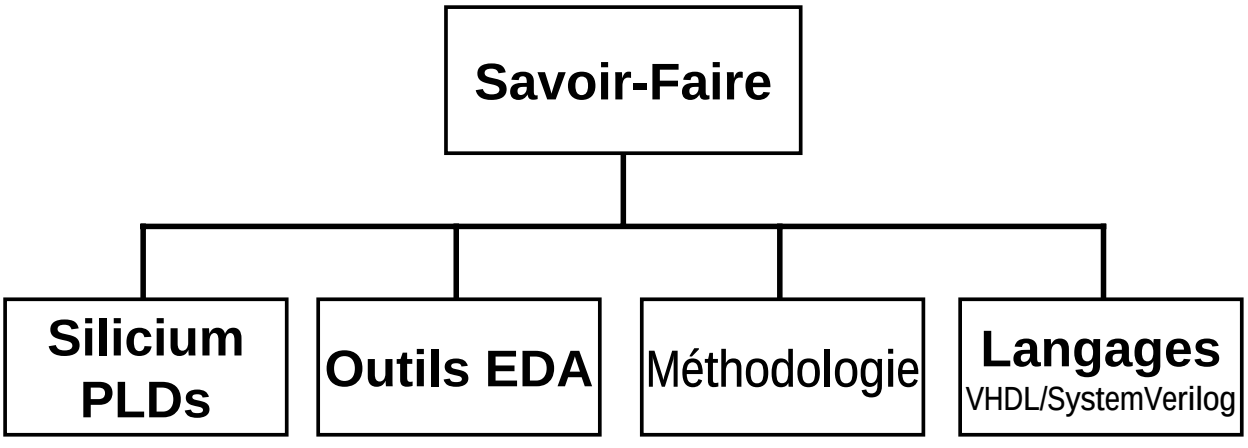
Conseil de M. Bertrand Cuzeau :

Réussir la conception des FPGA complexes

Do it right, the first time !

Bertrand Cuzeau
Forum CPLD-FPGA 2000
à Yverdon-les-Bains

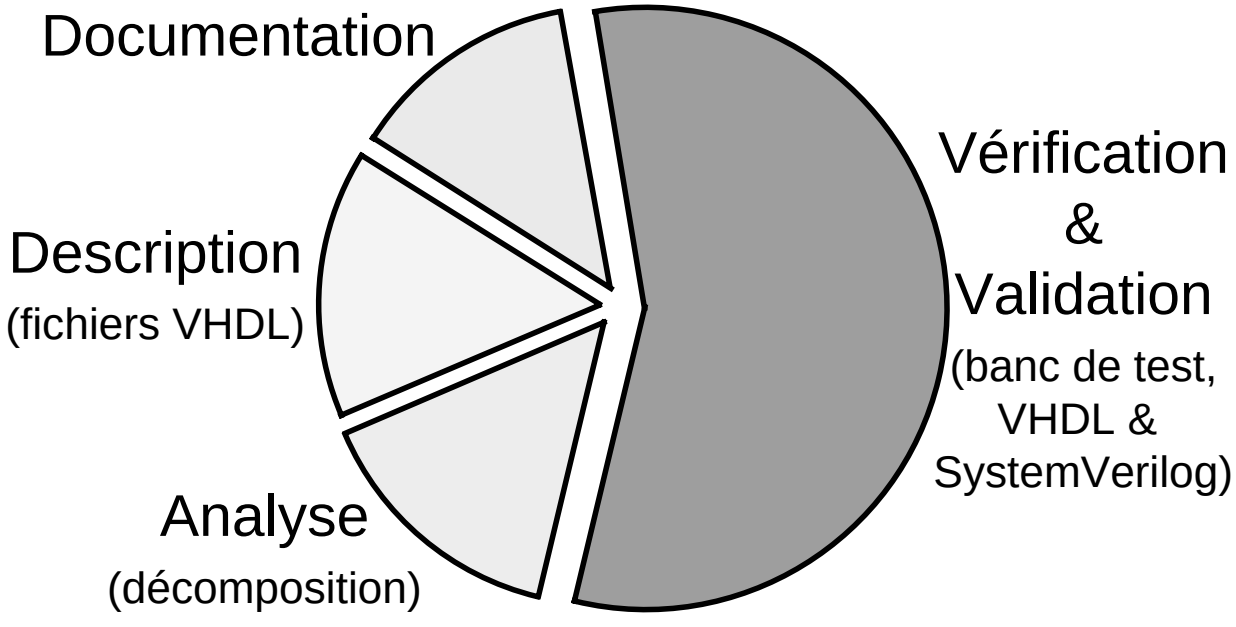
Les clés du succès …



… les clés du succès

- Importance d'une analyse détaillée
- Simplicité
  - ✓décomposer un système complexe en blocs simples
  - ✓à toutes les étapes : simplicité = clarté
- Modularité
  - ✓augmente le rendement
  - ✓permet la ré-utilisation (IP)
- Abstraction
  - ✓permet d'augmenter la productivité des descriptions
  - ✓imaginer l'interaction entre les blocs

Phases d'un projet



Méthodologie de conception …

- Analyse du cahier des charges
- Décomposition du système (*Top – Down)*
  - ✓si nécessaire plusieurs décomposition
  - ✓obtenir des blocs simples, avec une seule fonction
- Description en VHDL et validation de chaque bloc
  - ✓garantir la fonctionnalité de chaque bloc
- Rassembler les blocs élémentaires du système
  - ✓validation à chaque étape

… méthodologie de conception ..

- Analyse du cahier des charges
  - ✓jamais trop détaillée
  - ✓déterminer les contraintes des E/S (timing, asynchrone, …)
- Décomposition du système (*Top – Down)*
  - ✓étudier plusieurs architectures
  - ✓choix judicieux de l'architecture permet d'optimiser la quantité de logique
    => souvent implique une solution plus rapide
    => gain double !!
  - ✓si nécessaire plusieurs décompositions (hiérarchies)
  - ✓obtenir des blocs simples, avec une seule fonction

… méthodologie de conception …

- Description des blocs en VHDL:

  la description doit être simple

  que des avantages :
  - ✓simple égal matériel optimum
  - ✓lisible égal maintenance facilitée
  - ✓facilite le dépannage de la description et du système
  - ✓**améliore** la traduction par le synthétiseur
- Description en VHDL :
  - ✓rigueur et méthode
  - ✓respecter cahier méthodologique (voir ci-après)

… méthodologie de conception

- Méthodologie de vérification
  - ✓validation de bas en haut (bottom-up)
  - ✓utiliser des bancs de test automatique (GO – noGO)
  - ✓simuler chaque bloc séparément, cela diminue le nombre de pas du tests global
  - ✓Utiliser des stimuli aléatoires
- Rassembler les blocs élémentaires du système avec une validation à chaque hiérarchie
  - ✓à chaque hiérarchie : vérification des interconnexions
  - ✓au top : vérification de la fonctionnalité global en sachant que les briques du système sont fonctionnelles

Cahier méthodologique

- Défini des règles pour la description de systèmes numériques avec un langage de haut niveau
  - ✓rigueur indispensable
  - ✓description uniforme => facilite la portabilité entre collègue
  - ✓cahier méthodologique devrait être défini dans chaque entreprise

- Exemple de méthodologie avec le langage VHDL
  - ✓facilement adaptable pour Verilog/SystemVerilog

Généralités

- Nom *directory* :
  - ✓uniquement des lettres, chiffres et souligné _
- Structurer vos projets:

<Dir_Projet>\src	fichiers sources VHDL
...\lib	fichiers sources des paquetages
...\comp	fichiers pour la simulation
...\synth	fichiers pour la synthèse
...\p_r	fichiers pour le placement & routage

Mots réservés et identificateurs

Méthodologie institut REDS

- Mots réservés :
  - ✓ ils seront écrits en minuscule, affichés en gras (couleur).
- Identificateurs :
  - ✓ ils seront écrits en minuscule avec la première lettre en majuscule
  - ✓ chaque mot recommence par une majuscule séparé par un souligné (underscore)

  Exemple : `Bus_Data, Entree_Serie`, ...

Uniquement des lettres, des chiffres et le souligné

Convention pour les identificateurs

Méthodologie institut REDS

- Déclaration au top :

```
n<Nom_Signal>_o     pour un signal actif bas
                    (polarité négative)
```

- Déclaration dans l'entité :

```
<Nom_Signal>_i      pour une entrée (in)
<Nom_Signal>_o      pour une sortie (out)
<Nom_Signal>_io     pour une entrée/sortie (inout)
<Nom_constante>_g   pour une constante générique
```

- Déclaration dans l'architecture :

```
<Nom_Signal>_s      pour un signal interne
<Nom_Const>_c       pour une constante
```

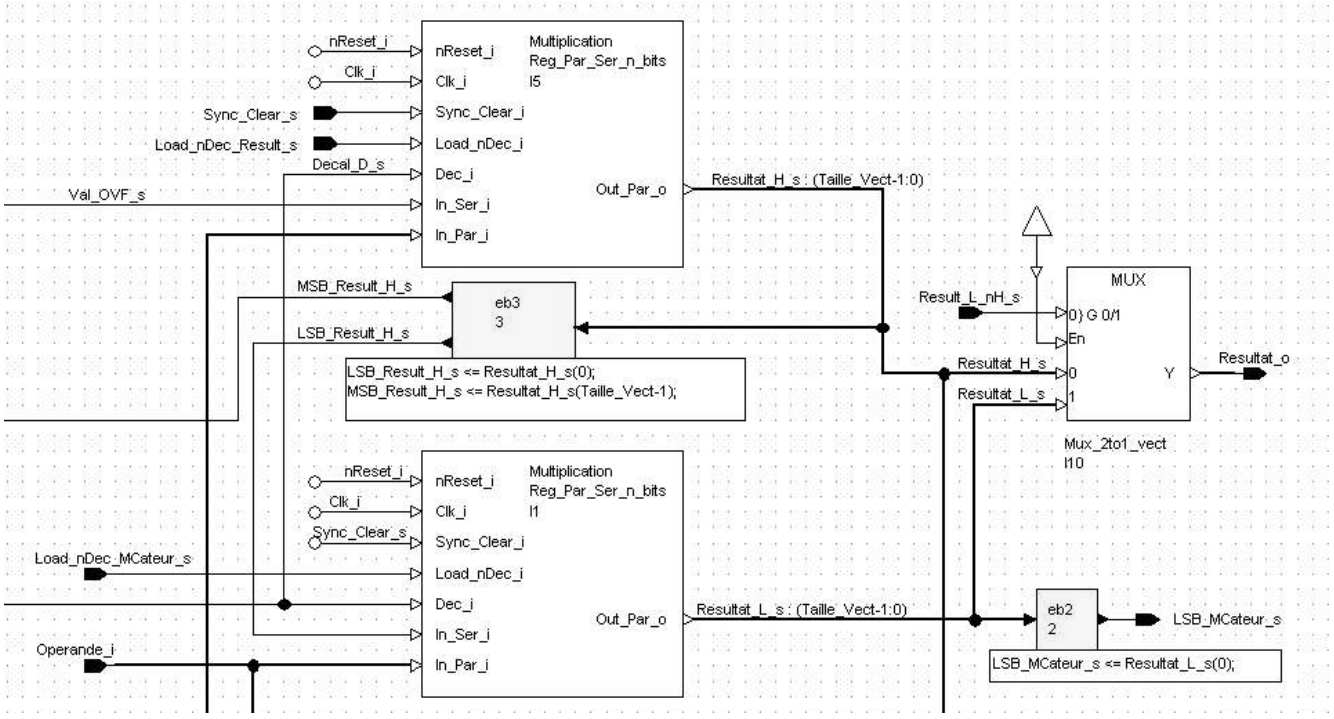
- Déclaration dans un processus :

```
<Nom_Variable>_v    pour une variable
```

Choisir des noms explicites …

- Noms des composants et des signaux doivent être explicites
  - ✓auto-documente le projet
  - ✓facilite la lecture des descriptions, des schémas, …
- Nom composant :
  - ✓ doit indiquer la fonction
- Noms signaux dans un composant :
  - ✓explicite avec la fonction interne
- Noms signaux dans un schéma :
  - ✓explicite avec la fonction réalisée dans le schéma

… noms de signaux explicites



Fichiers VHDL

- Un fichier VHDL contient une seule entité et son architecture avec la déclaration des paquetages
- Nom fichier VHDL = nom entité
  - ✓uniquement : lettres, chiffres, underscore
- Fichiers VHDL de simulation :
  - ✓ <nom_fichier>_tb.vhd

Entête des fichiers VHDL

- L'entête doit contenir :
  - ✓société, nom fichier, auteur
  - ✓logiciels utilisés (nom et version)
  - ✓explication du fonctionnement du module
  - ✓liste des fichiers utilisés (hiérarchie)
  - ✓liste des modifications (date, qui, quoi, version, ...)
  - ✓description des entrées-sorties dans l'entité

Top du projet

- Adapter la polarité des signaux au top
  - ✓logique mixte à l'extérieur (polarité positive ou négative)
  - ✓à l'intérieur uniquement en logique positive
- Signaux bidirectionnels SEULEMENT au top
  - ✓au top: instanciation des portes 3 états
  - ✓transmettre dans la hiérarchie des signaux unidirectionnels
  - ✓Identifier les signaux en entrée, en sortie ou bidirectionnel
- Type des signaux au top uniquement :

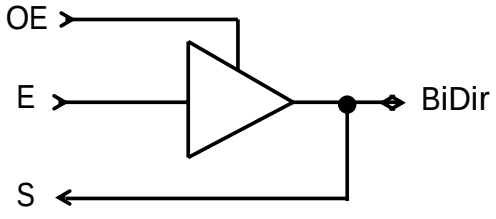
```
Std_Logic, Std_Logic_Vector
```


… au top du projet (porte BiDir)

- Instanciation des portes bi-directionnelles au top
  Utiliser la fonction To_X01

```
BiDir_io <= E_i  when OE_i = '1' else 'Z';
S_o      <= To_X01(BiDir_io);
```

La fonction To_X01 permet de convertir l'état 'H' ou 'L' dans un état logique '1' ou respectivement '0'.



Types des signaux autorisés

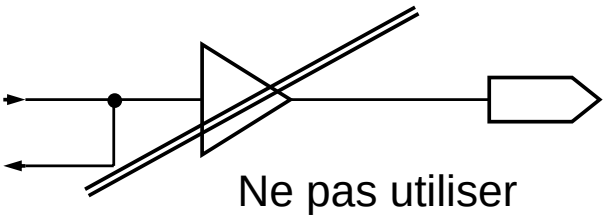
- Types utilisés dans l'entité (interconnexion entre module) : uniquement `Std_Logic` et `Std_Logic_Vector`
- Types utilisable à l'intérieur d'un module :
  - ✓tous les types sont autorisés
  - ✓types recommandés : `Std_Logic, Std_Logic_Vector et Unsigned, Signed` (opérations arithmétiques).
  - ✓les types `Boolean, Natural` et `Integer` sont à utiliser par des personnes expérimentées.

Types de *port* ...

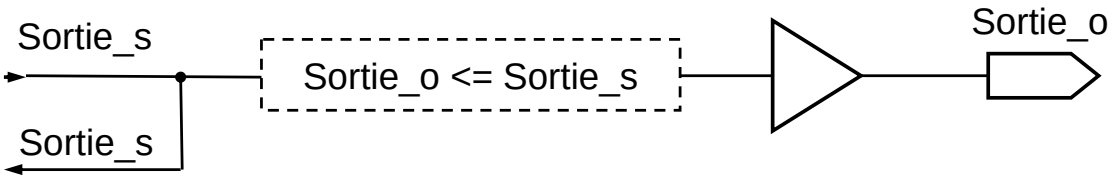
- Utiliser : **`in`**, **`out`**, **`inout`**.
- Le type **`inout`** sera utilisé uniquement dans le module top. Il n'y a pas de ligne trois états à l'intérieur d'un PLD, sauf
  - ✓avec certains FPGA !
  - ✓dans le cas des ASICs (autre méthodologie)
- Le type **`buffer`** ne sera jamais utilisé. Il faut utiliser un signal interne.

… types de *port*

`port` **`buffer`**



Remplacer par un signal interne et un *port* **`out :`**



Déclaration des vecteurs

Toujours utiliser la déclaration :

n-1 downto 0, voici un exemple :

```
signal Vecteur : Std_Logic_Vector(31 downto 0);
```

- Si vecteur définit : 0 to n-1 NE JAMAIS UTILISER !!!!

```
signal Vect_to : Std_Logic_Vector(0 to 31);
```

Inversion de tous les bits lors de l'affectation de deux vecteurs déclarer différemment :

```
Vecteur <= Vect_to; --Tous les bits sont croisés !!
```

Bibliothèques

- Bibliothèque IEEE autorisées :
  - ✓`Std_Logic_1164` types et fonctions de base.
  - ✓`Numeric_Std` opérateurs arithmétiques.
- plus bibliothèques propres au projet, à l'entreprise :
  - ✓regrouper les déclarations communes
  - ✓bien gérer les versions
  - ✓fonctions et procédures pour la simulation.
  - ✓etc.

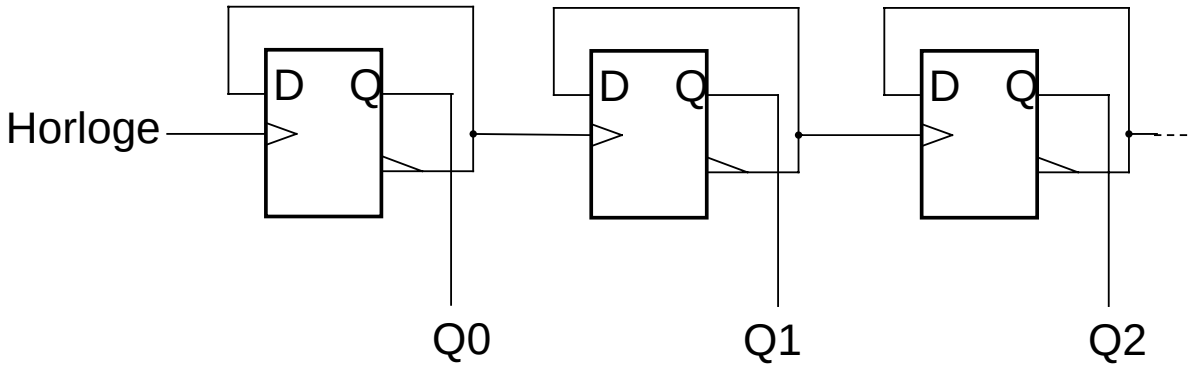
Pourquoi "full synchrone" ?

- Avantages conception full synchrone :
  - ✓permet une abstraction, facilite la conception
  - ✓évolution prédictible, changement vu au prochain flancs
  - ✓retard identique pour chaque sortie de flip-flop
  - ✓analyse statique permet de déterminer la fréquence maximum de fonctionnement du système, Fmax
  - ✓arbre d'horloge prédéfinit dans les PLDs
    => prévu pour l'intégration de designs full synchrone

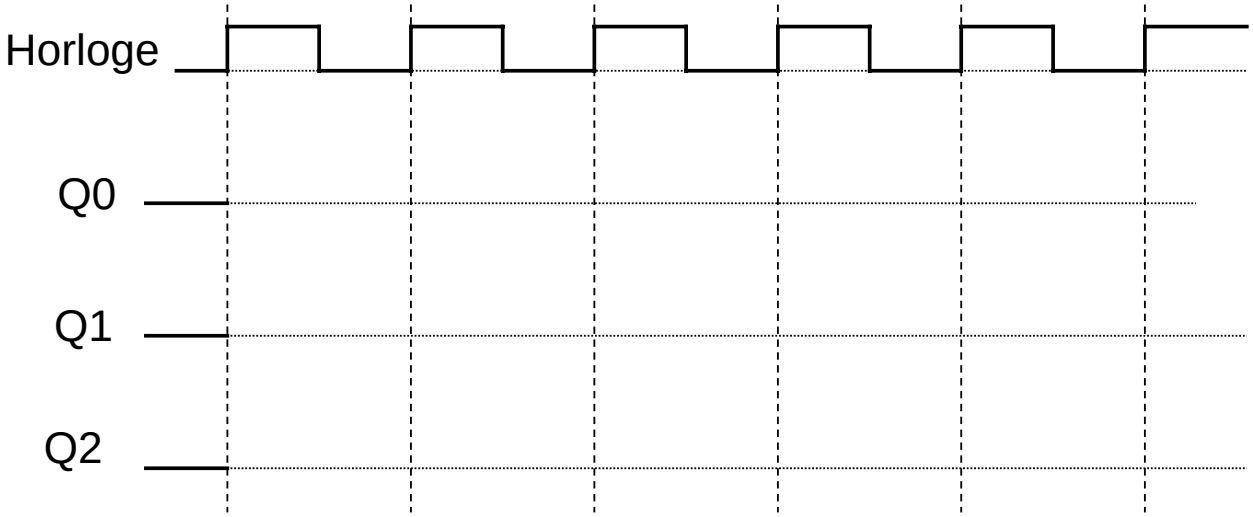
Type de conception

- Pseudo-synchrone
  - ✓Horloge d'un flip-flop dépend du ou des flip-flops précédents
- Full synchrone
  - ✓Toutes les flip-flops sont synchronisées par le même signal d'horloge

Compteur pseudo synchrone (ripple counter)



Compteur pseudo synchrone

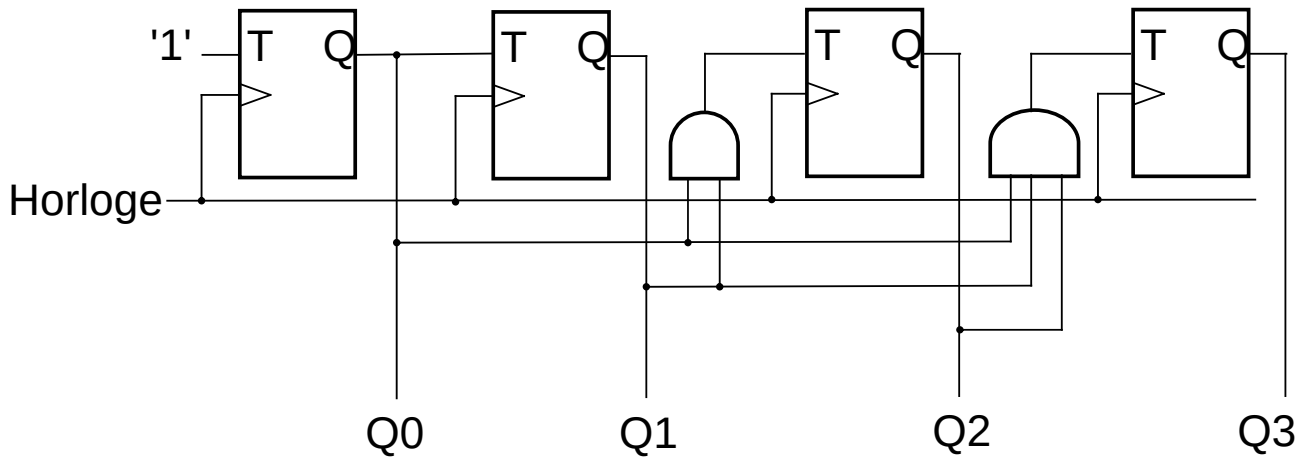


Analyse compteur pseudo synchrone

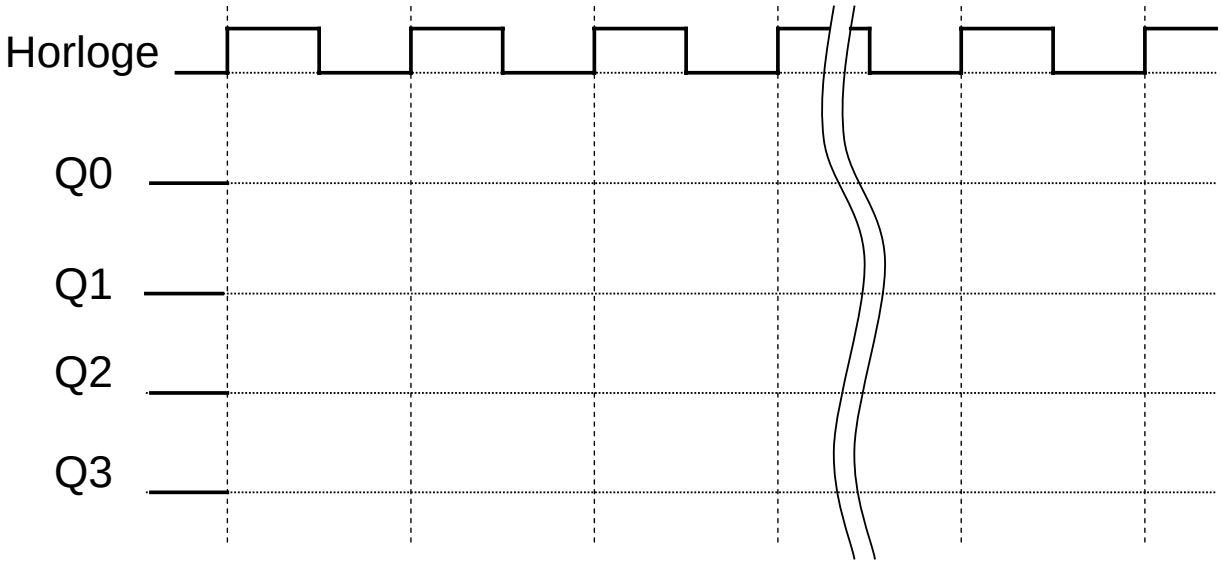
- Construction très simple
- Pas synchrone
  - ✓Retard variable pour chaque sortie
  - ✓Retard cellule N : Tn = N x tp
  - ✓Très difficile à tester (vérification timing)
  - ✓Possible sortie change après flanc suivant de l'horloge
- Très mauvaise intégration dans des PLDs

Dia laissé vide volontairement

Compteur *full* synchrone



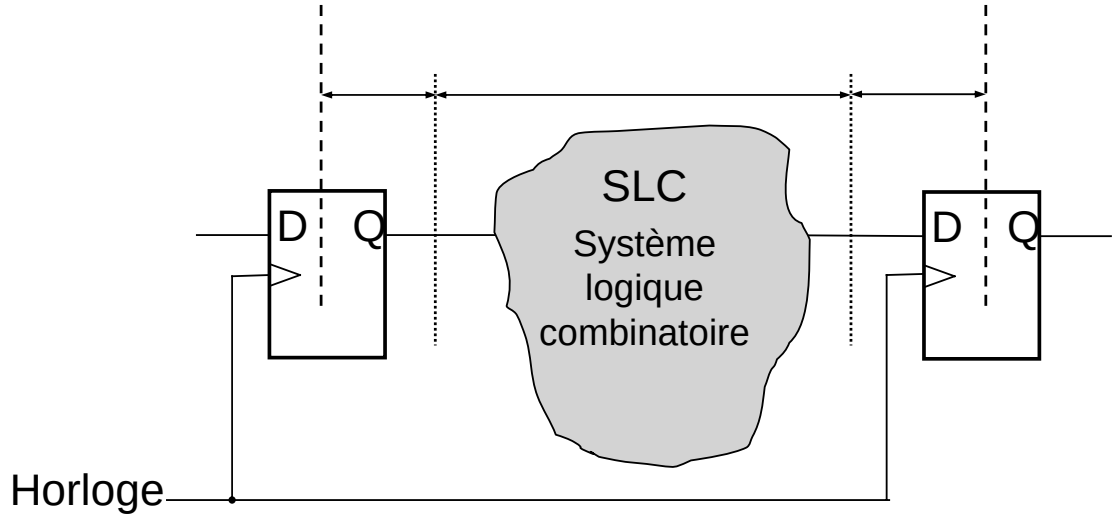
Compteur *full* synchrone



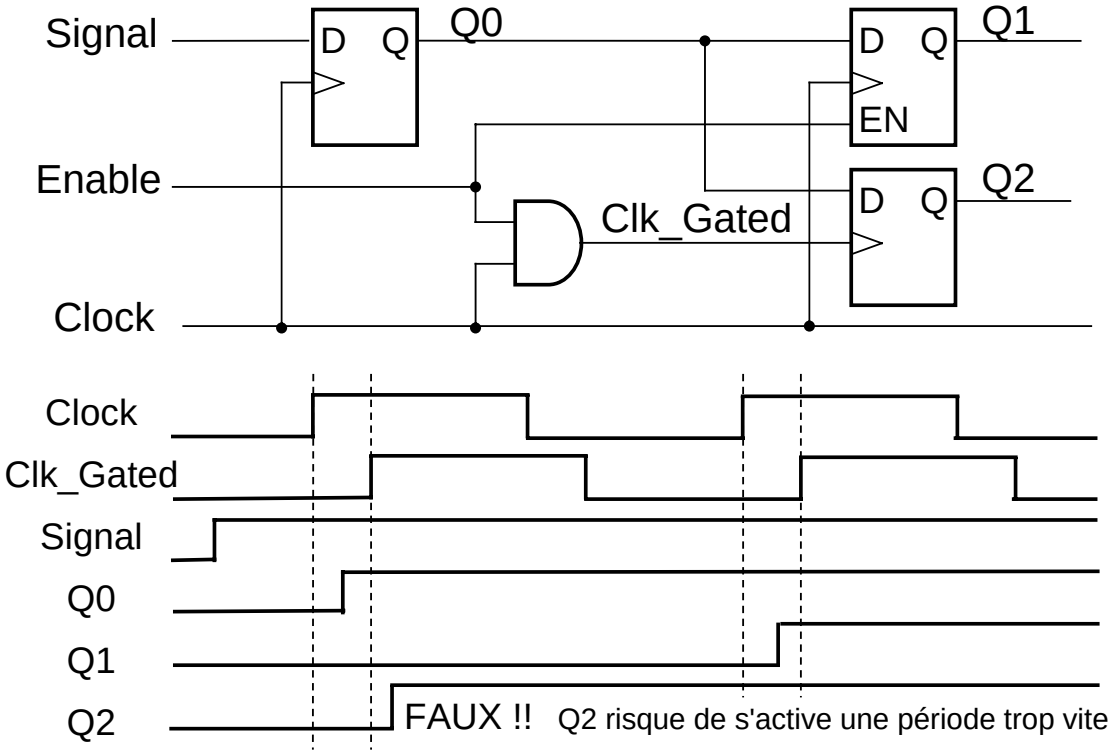
Analyse du compteur synchrone

- Construction plus complexe
- Synchrone
  - ✓Retard identique pour chaque sortie indépendant de la taille du compteur
  - ✓Vérification fréquence maximum simple : $Fmax < 1 / (tp_{DFF} + tp_{COMB} + tset\text{-}up_{DFF})$
  - ✓Mauvais fonctionnement impossible si F < Fmax
- Très bonne intégration dans des PLDs

Analyse statique des temps



Problème avec "gated clock" …



… problème avec "gated clock"

- Analyse bascule Q1 :
  - ✓Q0 arrivera TOUJOURS après le flanc montant
  - ✓Q1 sera mis à jour au prochain flanc
  - ✓fonctionnement indépendant du tp des flip-flops suffit Fclock < Fmax
- Analyse bascule Q2 :
  - ✓Q0 arrivera TOUJOURS après le flanc montant
  - ✓mais le flanc montant de Clk_gated est retardé !
  - ✓risque que la bascule Q2 voit le changement de Q0 durant le même cycle d'horloge !
  - ✓fonctionnement dépend des timings des portes !!

Arbre d'horloge dans un PLD

- Dans PLD il y a un arbre d'horloge pré-cablé
- L'arbre garanti que TOUS les flip-flops voient l'horloge au même instant
  - ✓ skew (décalage) doit être être le plus faible possible
  - ✓ skew inférieur aux tp internes
  - ✓ caractéristiques pour Cyclone II de Altrera

Cyclone II, Altera

Table 5–35. Clock Network Specifications

Name	Description	Max	Unit
Clock skew adder EP2C5, EP2C8 *(1)*	Inter-clock network, same bank	±88	ps
	Inter-clock network, same side and entire chip	±88	ps
Clock skew adder EP2C15, EP2C20, EP2C35, EP2C50, EP2C70 *(1)*	Inter-clock network, same bank	±118	ps
	Inter-clock network, same side and entire chip	±138	ps

Note to Table 5–35:
(1) This is in addition to intra-clock network skew, which is modeled in the Quartus II software.

Table 5–16. LE_FF Internal Timing Microparameters (Part 1 of 2)

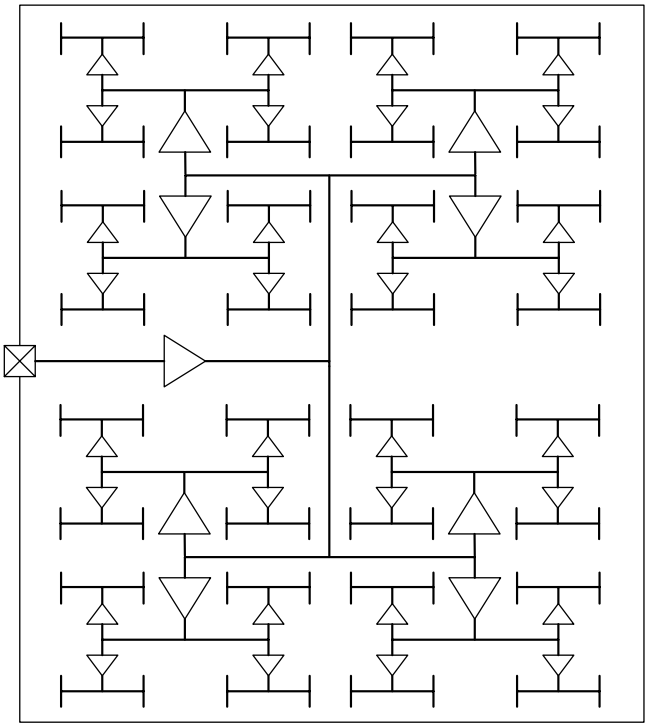
Parameter	-6 Speed Grade *(1)*		-7 Speed Grade *(2)*		-8 Speed Grade *(2)*		Unit
	Min	Max	Min	Max	Min	Max	
TCO	141	250	141	277	135	304	ps
					141		ps

Cyclone II, Altera
TCO: clock-to-out

Structure d'un arbre d'horloge

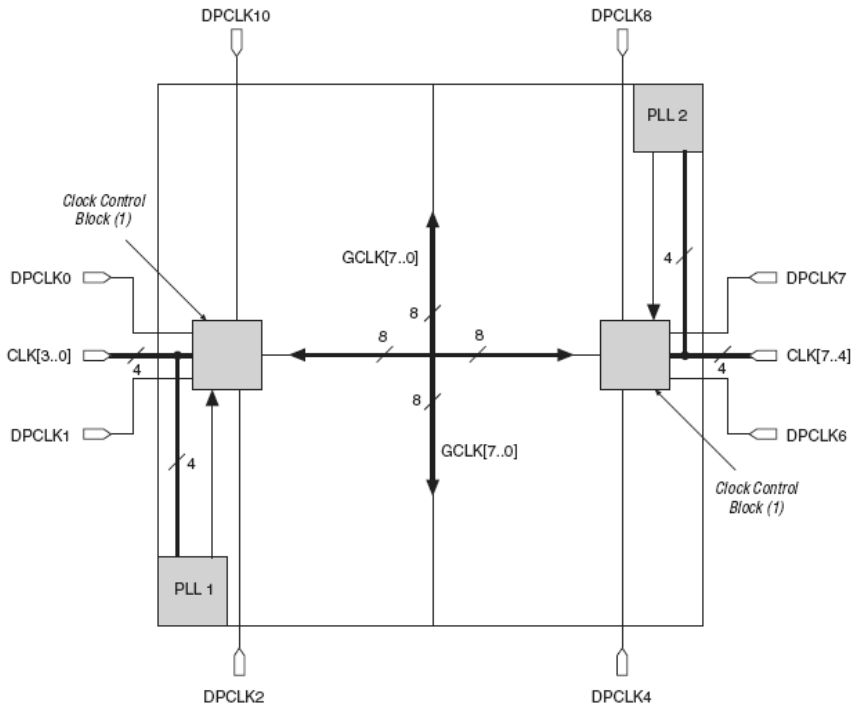
- Arbre équilibré afin de garantir un temps de propagation constant pour tous les flip-flops

Entrée d'horloge



Exemple arbre d'horloge Cyclone II

Figure 2–11. EP2C5 & EP2C8 PLL, CLK[], DPCLK[] & Clock Control Block Locations



Dia laissé vide volontairement

Méthodologie de conception pour FPGA

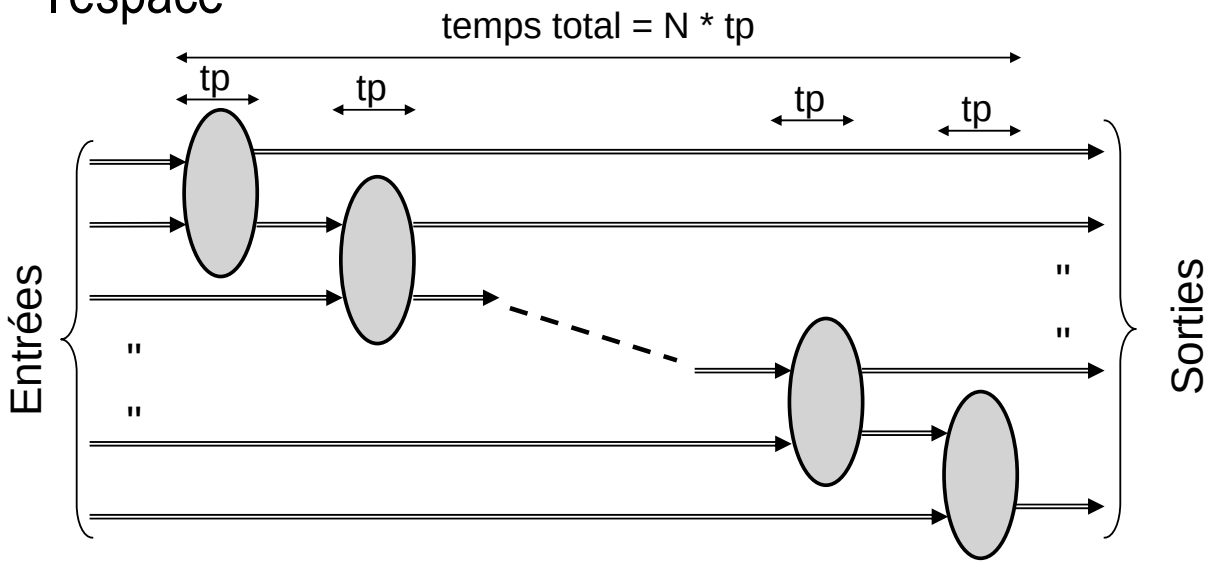
Types de décomposition

Décomposition circuits combinatoires

- Réalisation de système de traitement de données:
  - ✓il s'agit souvent d'opérations combinatoires complexes
- Deux structures possible:
  - ✓solution purement combinatoire : décomposition spatiale
  - ✓solution séquentielle : décomposition temporelle

Décomposition spatiale …

- La complexité est décomposée en modules de base combinatoire qui sont ensuite interconnectés dans l'espace

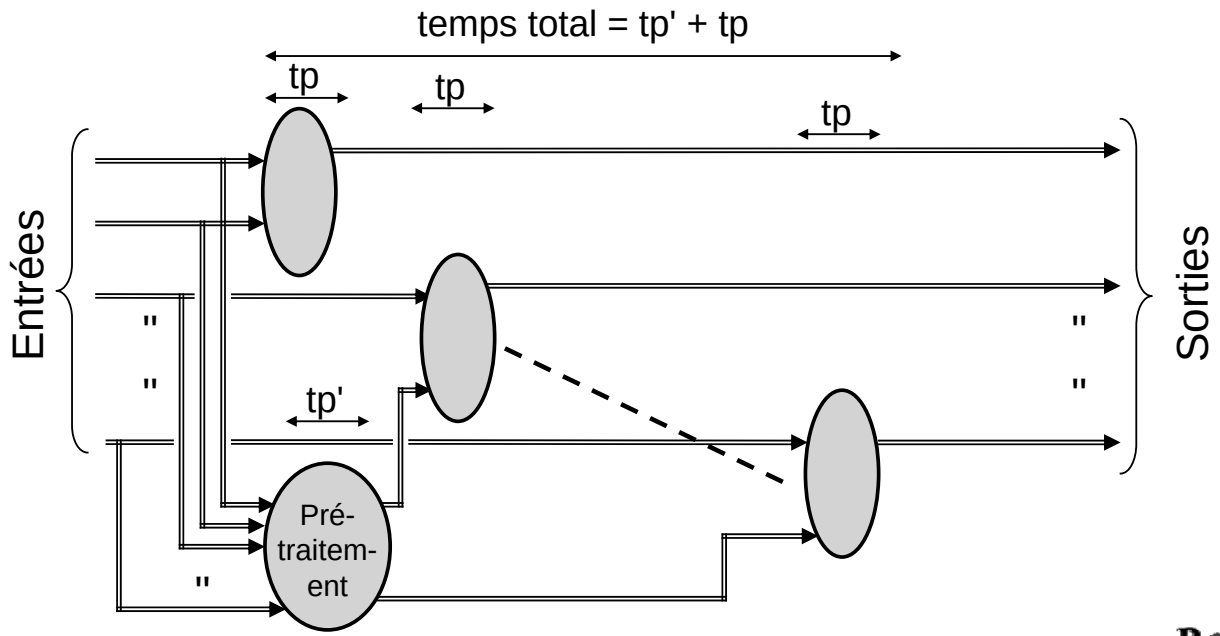


Décomposition spatiale : optimisation …

- L'optimisation d'une solution spatiale nécessite de connaître les chemins critiques (le plus long)
- Dans le cas de l'addition, le chemin critique est le report. Le résultat de l'addition est valide seulement lorsque le dernier report est calculé
- La solution est de trouver un ou plusieurs chemins parallèles qui permettent de raccourcir le temps de calcul. Cela nécessite du matériel supplémentaire.
  - ✓ pour l'addition il s'agit de la technique d'anticipation du report (carry look-ahead)

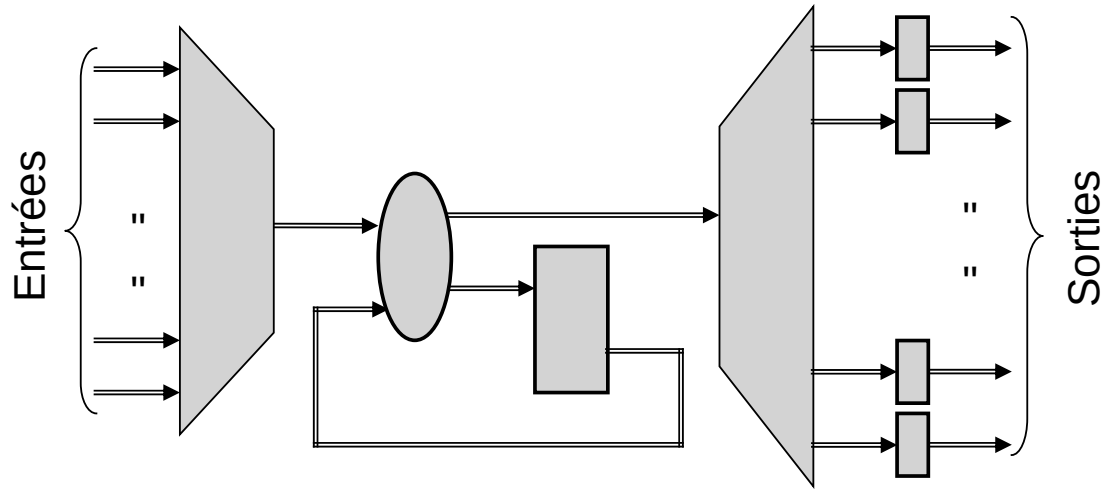
… décomposition spatiale : optimisation

- Un ou plusieurs modules en début de traitement permettent de supprimer le chainage des modules



Décomposition temporelle

- Le système est composé d'un seul module de base qui est utilisé séquentiellement pour réaliser toutes les étapes nécessaires au calcul



Décomposition temporelle : optimisation

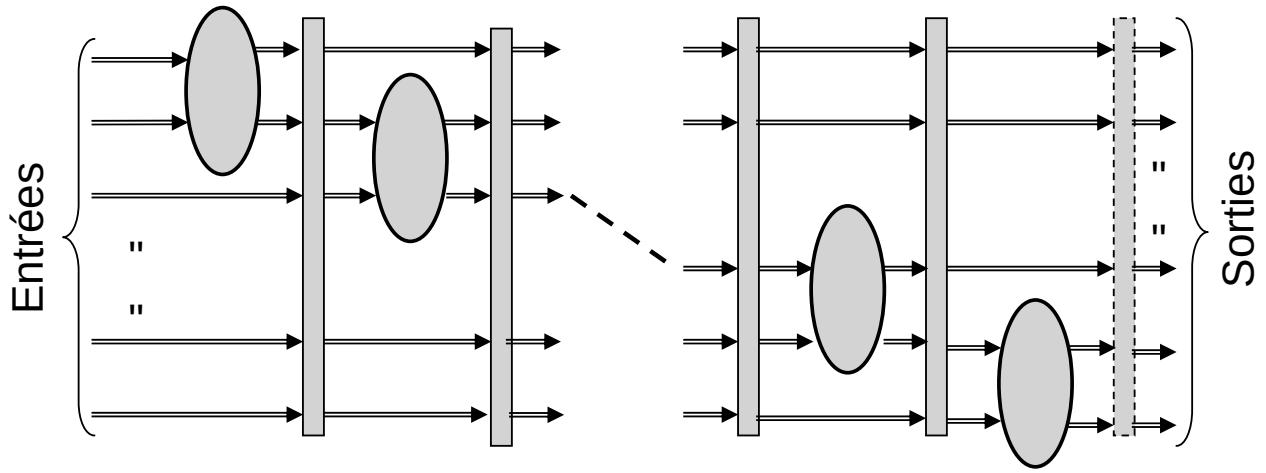
- La partie principale à optimiser : module combinatoire
  - ✓Ce module est implanté une seule fois, il possible de paralléliser le traitement (augmentation de la quantité de matérielle)
  - ✓Il est aussi possible d'utiliser un bloc RAM du PLD Solution parfois plus rapide que de la logique
  - ✓Exemple : module de base additionneur 4 bits réalisation parallèle de l'additionneur (ne pas utiliser une structure de chaînage d'additionneur 1 bit)
- Concernant les registres, seul un changement de technologie permet d'améliorer les performances

Structure pipeline …

- L'objectif est d'augmenter le débit de traitement. Le système combinatoire est découpée en petite tranche afin de pouvoir utiliser un fréquence plus élevée.
- Réalisation :
  - ✓ Fractionner le système combinatoire (décomposition spatiale) en tranche de calcul. Chaque tranche nécessite un temps beaucoup plus court (tp)
  - ✓ Mémoriser les résultats intermédiaires entre chaque tranche du calcul
  - ✓ A chaque période, une étape est calculée. Il faudra N périodes d'horloge réaliser une opération (calcul)
  - ✓ Lorsque le pipeline est amorcé, il y a un résultat fourni à chaque période

… structure pipeline …

- Voici le schéma pour une réalisation avec une structure "pipeline"



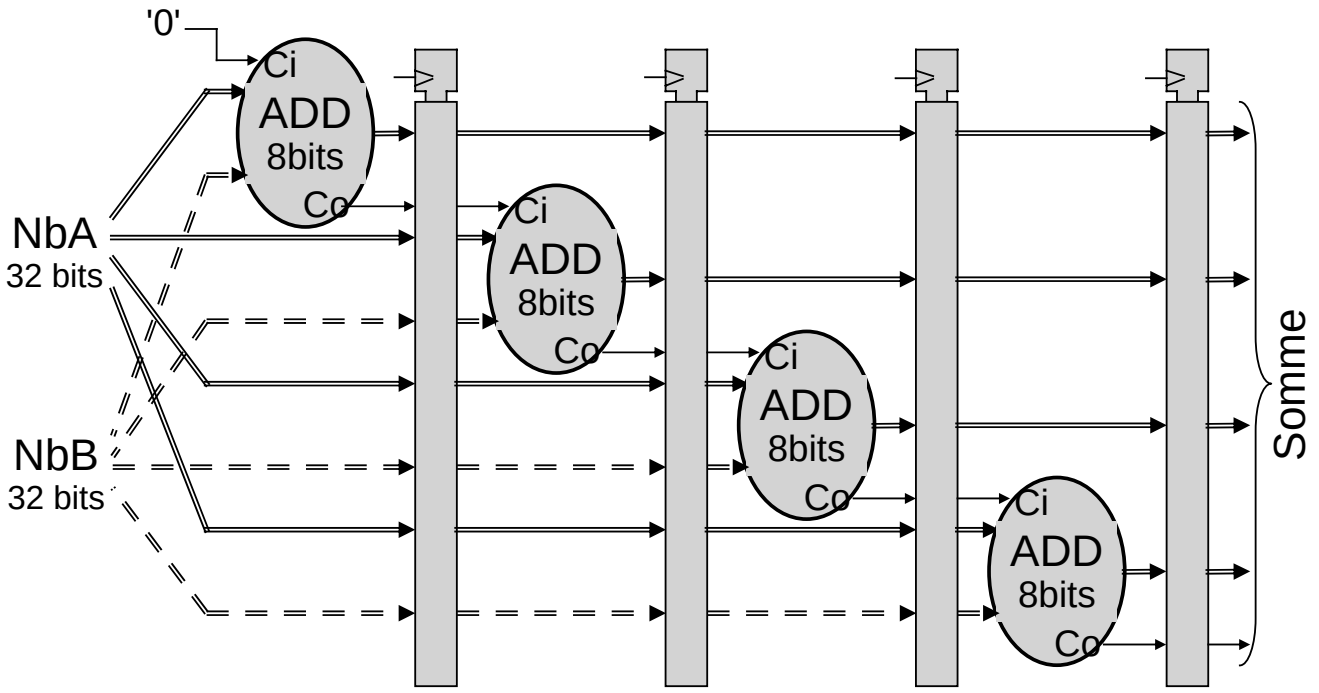
… structure pipeline …

- Cette solution est très performante dans le cas de traitement utilisé de nombreuse fois; nombre important de données à traiter
- Permet d'atteindre des débit de traitement très élevé
- Implique un temps de latence important, nombre d'étages du pipeline
- Cette structure utilise passablement de matériel (module combinatoire + registre intermédiaires)

Structure pipeline pour l'addition …

- Additionner deux nombres de 32 bits non signés
- On dispose d'additionneurs 8 bits avec tpADD = 10ns et de registres 8 bits avec tpREG = 3ns (8 flip-flops)
- On souhaite disposer du plus grand débit de traitement possible.
- Solutions possibles :
  - ✓Solution combinatoire: N * tpADD = 4 * 10ns = 40ns
  - ✓Solution séquentielle : pas performante
  - ✓Solution avec pipeline : tpADD + tpREG = 13ns
    - →par contre il faudra 4 périodes pour amorcer le pipeline

… structure pipeline pour l'addition

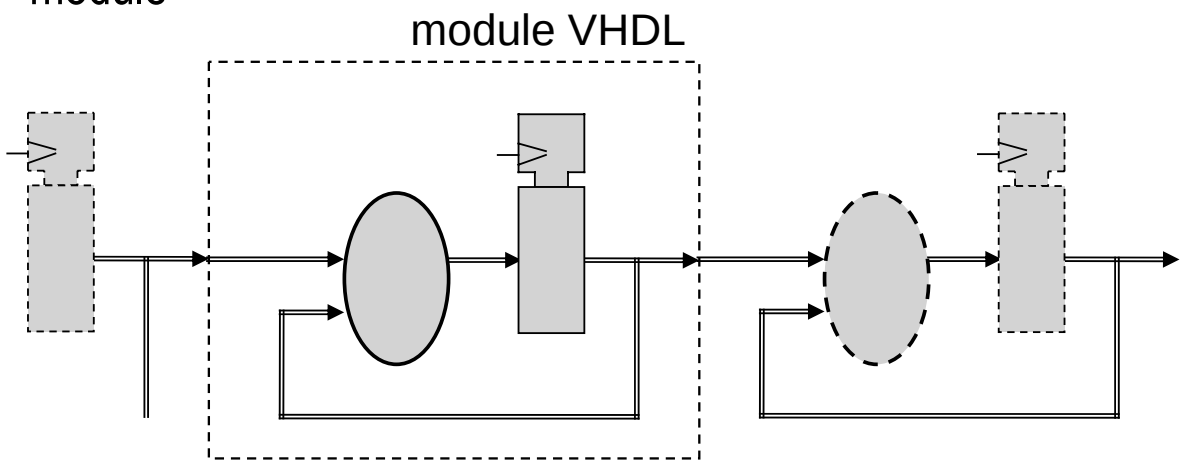


Amélioration performances

- Egaliser au maximum les chemins combinatoires
- Si fréquence fonctionnement trop faible
  - ✓il y a un ou plusieurs chemins combinatoire trop long
  - ✓rechercher les chemins critiques entre deux flip-flops
    => utiliser l'analyse de timing statique des outils PR
- Solution :
  - ✓déplacer le bloc combinatoire,
  - ✓anticiper l'opération combinatoire
  - ✓couper le bloc combinatoire en plusieurs parties selon le principe d'une structure pipeline

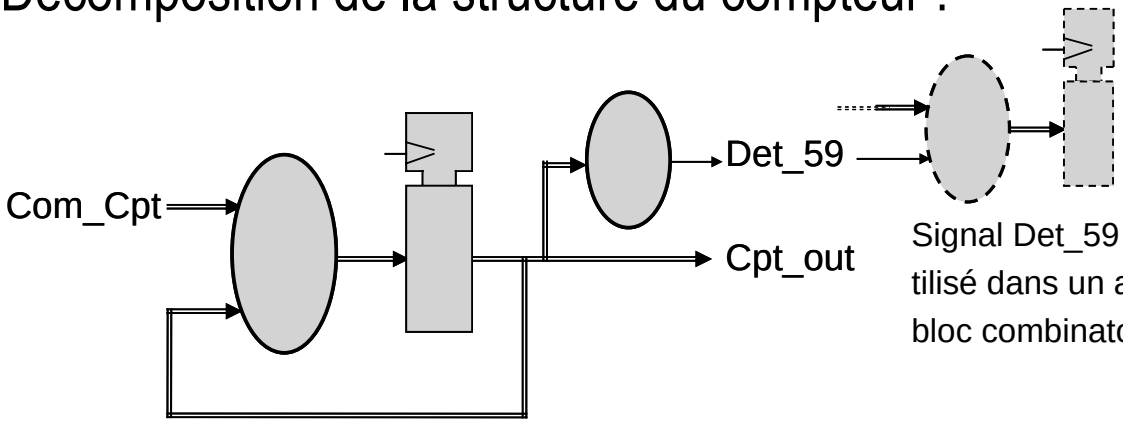
Maitriser des performances

- Sorties d'un module VHDL correspond toujours à la sortie d'une bascule
- Optimisation des timings possible séparément pour chaque module



Exemple : report d'un compteur …

- Décomposition de la structure du compteur :

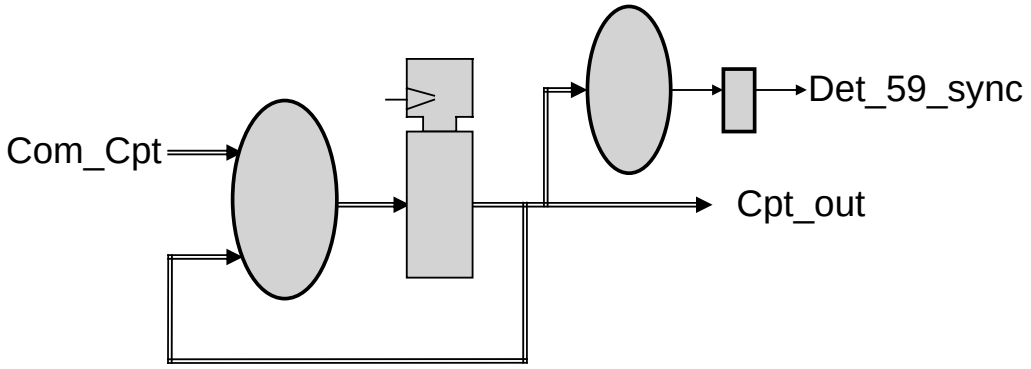


Analyse:

La sortie Det_59 a un délai de propagation non négligeable après l'activation de l'horloge (flanc).
Les blocs combinatoires utilisant ce signal seront retardés d'autant

… exemple : report d'un compteur …

- Solution : post-synchroniser le signal Det_59



Analyse:

La sortie Det_59_sync a un délai de propagation très faible par rapport au flanc de l'horloge.
Mais le signal est retardé d'une période d'horloge

… exemple : report d'un compteur …

- Description VHDL du report:

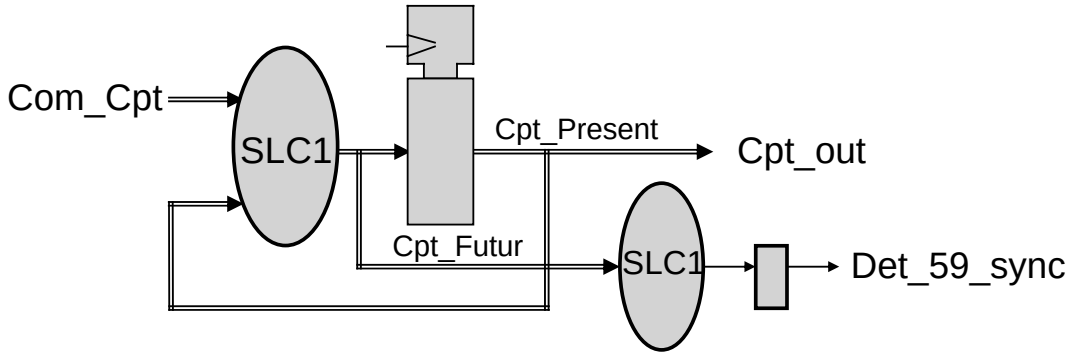
```
Det_59 <= '1' when Cpt_Present = 59 else
          '0';
```

- Description utilisant Cpt_Futur et la post-synchronisation

```
process (horloge)
begin
  if Rising_Edge(Horloge) then
    if Cpt_Futur = 59 then
       Det_59_sync <= '1';
    else
       Det_59_sync <= '1';
  end if;
end process;
```

… exemple : report d'un compteur …

- Cela correspond au schéma suivant :

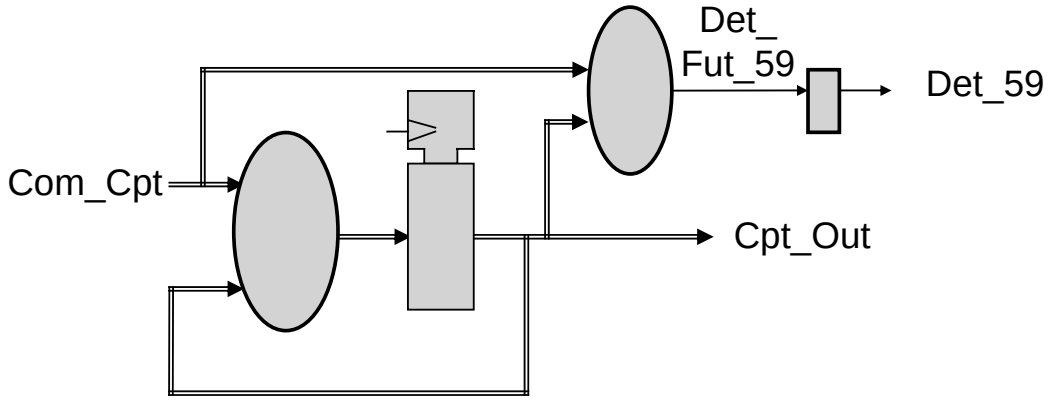


Analyse:

La sortie Det_59_sync est correcte avec un délai de propagation très court.
Par contre la suite des blocs SLC1 + SLC2 risque de devenir un nouveau chemin critique

… exemple : report d'un compteur …

- Anticiper la détection du compteur :



Analyse:

Solution optimale
Attention de bien déterminer la condition "Cond_Comptage" du compteur pour déterminer Det_Fut_59

… exemple : report d'un compteur

- Description utilisant Cpt_Present et la condition de comptage :

```
Det_59_Fut <= '1' when Cpt_Present = 58 and
                       Cond_Comptage = '1' else
              '0';

process (horloge)
begin
  if Rising_Edge(Horloge) then
       Det_59_sync <= Det_59_Fut ;
  end if;
end process;
```

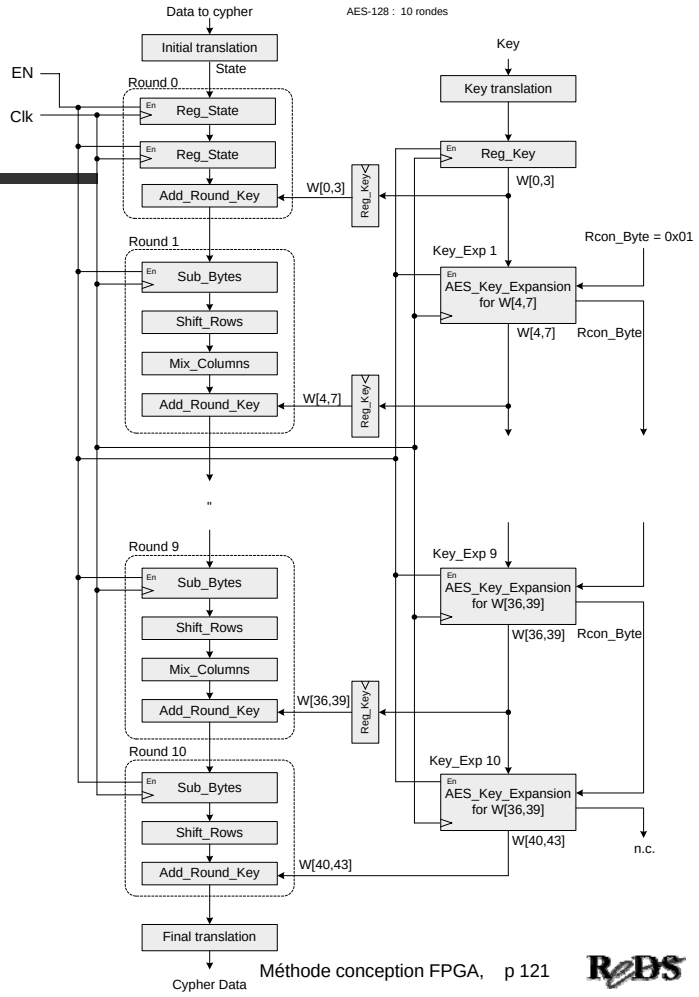

Exemple implantation de AES

- Algorithme de cryptage AES
  - ✓Version 128 bits
  - ✓structure full pipeline :
    - →débit maximum
    - →nécessite beaucoup de logique
  - ✓Structure mixte : séquentiel/pipeline
    - →excellent débit
    - →quantité de logique divisée par deux
      Remarque: logique divisé par 4 pour FPGA X2C3000
    - →très bon compromis

AES full pipeline

Implantation V1

- Vue de la structure de l'implantation
  "full pipeline" de l'algorithme AES
- Calcul en parallèle de la clé et du cryptage



Implantation full pipeline AES, V1

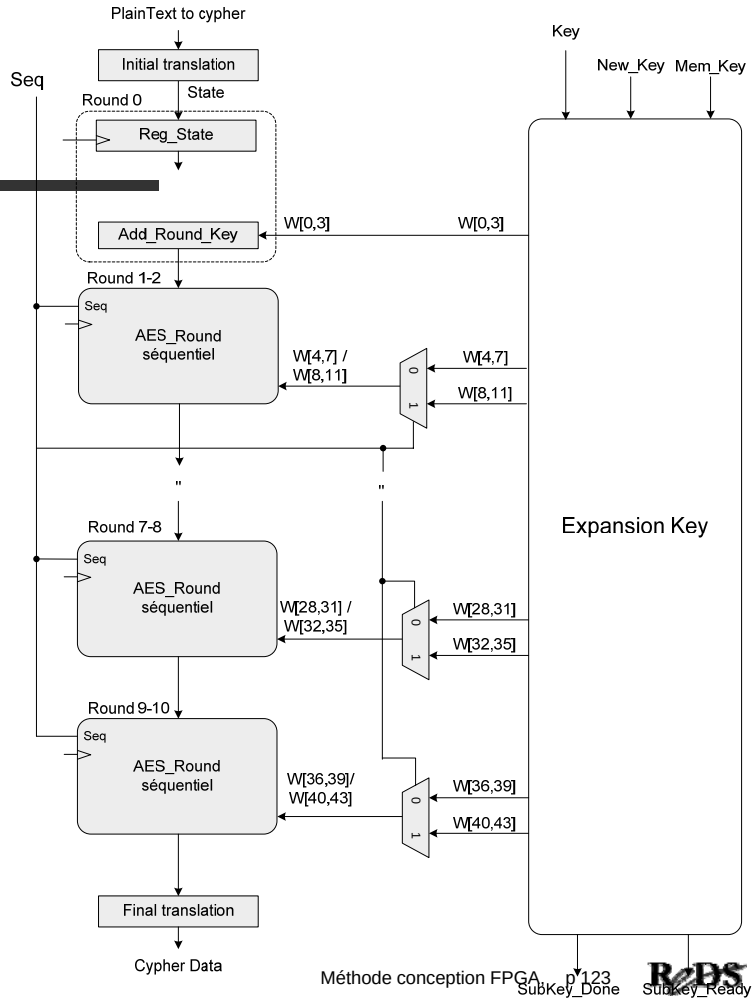
- Description nécessite 200 bloc RAM de 256x8bits
- FPGA XC2V3000 dispose de 96 bloc RAMs de 8Kbits, solde implanté dans des LUT
- Résultat du placement-routage: AES utilise 80% de la FPGA!

Resource	Used	Avail.	Utilization
CLB Slices	12226	14336	85%
DFFs or Latches	3670	30220	12%
Block RAMs	96	96	100%
Timing			
ClockDomain	Clk_i	11.464 ns	(87.23 MHz)

- Débit de l'algorithme AES : 11 Gigabits/s

AES mixte

- Calcul préalable de la clé en séquentiel, puis mémorisation
- Cryptage avec en 2 cycles :
  - ✓un cycle séquentiel
  - ✓un cycle pipeline
  - ✓réduction par deux du nombre de rondes



Implantation mixte AES

- Clé calculée séquentiellement avant le cryptage
- Description nécessite 84 bloc RAM de 256x8bits
- FPGA XC2V3000 dispose de 96 bloc RAMs de 8Kbits, OK
- Résultat du placement-routage:

Resource	Used	Avail.	Utilization
CLB Slices	2772	14336	19%
DFFs or Latches	1595	30220	5%
Block RAMs	84	96	88%
Timing			
ClockDomain	Clk_i	9.982 ns	(100.180 MHz)

- Débit de l'algorithme AES (calcul en 2 cycles): 6 Gigabits/s

Comparaison implantations

- Résultat placement-routage des implantations de AES :

Resource	Full pipe	Seq-Pipe	Avail.
CLB Slices	12226 (85%)	2772 (19%)	14336
DFFs or Latches	3670 (12%)	1595 (5%)	30220
Block RAMs	96 (100%)	84 (88%)	96
Timing			
ClockDomain Clk_i	11.464 ns 87.23 MHz	9.982 ns 100.180 MHz	
Débit AES-128	11 Gigabits/s	6 Gigabits/s	

Implantation Seq-Pipe très bon compromis :

- ✓ Débit divisé par 2
- ✓ Quantité de logique divisée par 4 !

Méthodologie de conception pour FPGA

Annexes

Bibliographie: verification …

- Open Verification Methodology Handbook: Creating Testbenches in Systemverilog and SystemC
  M. Glasser, H. Foster, T. Fitzpatrick, A. Rose, D. Rich, novembre 2009 !
- Step-by-step Functional Verification with SystemVerilog and OVM
  Dr. Sasan Iman, Hansen Brown Publishing, mai 2008
- Le langage SystemVerilog : Synthèse et vérification des circuits numériques complexes
  Sébastien Moutault et Jacques Weber, Dunod, mars 2009
- Digital System Design With Systemverilog
  Mark Zwolinski, Prentice Hall, novembre 2009
- Verification methodology manual for SystemVerilog,
  Janick Bergeron, Springer-Verlag, 2005 (biblio HEIG-VD)

… bibliographie: verification

- Open Verification Methodology Cookbook
  Mark Glasser, Springer, juillet 2008

- SystemVerilog Golden Reference Guide, Doulos
- OVM Golden Reference Guide, Doulos

Normes et méthodologies

- IEEE Std 1800™-2007, Standard for SystemVerilog – Unified Hardware Design, Specification, and Verification Language
- IEEE Std 1364™-2005, Standard for Verilog® Hardware Description Language
- OVM SystemVerilog User Guide, v 2.0.2, Cadence-Mentor, june 2009

Bibliographie: design VHDL

[1] Manuel VHDL, synthèse et simulation, Etienne Messerli. HEIG-VD, 2007

[2] VHDL, Introduction à la synthèse logique, Philippe Larcher, Eyrolles, 1997 (Livre simple et facile d'accès, très bien pour les étudiants)

[3] Le langage VHDL, J. Weber & M. Meaudre, Dunod, 2001
(Bon livre pour débuter en VHDL)

[4] VHDL. Méthodologie de design et techniques avancées. Thierry Schneider, Dunod, 2001

[5] VHDL for Engineers, Kenneth L. Short, 2008, Pearson Internationnal (très bien)

[6] VHDL-2008 Just the new stuff, Peter L. Ashenden, Jim Lewis, Morgan Kaufman, 2008

[7] Digital System Design with VHDL, 2000, Mark Zwolinski, Prentice Hall

[8] VHDL du langage à la modélisation, Airiau & Bergé & Olive & Rouillard, édition 1990 et 1996, PPUR (référence pour concept du langage VHDL)

[9] VHDL Made Easy !, D. Pellerin et D. Taylor, Hardcover, 1996

… bibliographie: design VHDL

Normes IEEE:

[10] IEEE Standard VHDL Language Reference Manual (VHDL-1993), IEEE 1076-1993

[11] IEEE Standard Multivalue Logic System for VHDL Model Interoperability, IEEE Std 1164-1993

[12] IEEE Standard VHDL Synthesis Packages, IEEE-1076.3, 1997
Définit en autre le paquetage Numeric_Std, avec les types Unsigned et Signed.

[13] IEEE Standard for VHDL Register Transfer Level Synthesis, IEEE 1076.6-1999

Guides de références:

[14] The VHDL Golden Reference Guide, compatible IEEE std 1076-2002 disponible chez : Doulos, http://www.doulos.com/

… bibliographie: design VHDL

Guides de références:

[14] The VHDL Golden Reference Guide, compatible IEEE std 1076-2002 disponible chez : Doulos, http://www.doulos.com/

[15] ACTEL HDL Coding, Style guide, ACTEL, Edition 2003 disponible en PDF sur le site http://www.actel.com/

Articles:

[16] Circuits programmables et langages de conception, une évolution en parallèle, C. Guex & E. Messerli, Revue Vision 1998, EIVD

[17] Conception numérique: Description VHDL et synthèse, D. Gauthey & E. Messerli, Revue Vision 2000, EIVD

Sites internet outils EDA

- http://www.aldec.com/
- http://www.cadence.com/
- http://www.eve-team.com/index.php
- http://www.mentor.com/
- http://www.nusym.com/
- http://www.synopsys.com/home.aspx

Sites internet: SystemVerilog

- http://www.systemverilog.org/
- http://www.ovmworld.org/
- http://www.vmm-sv.org/
- http://www.doulos.com/knowhow/sysverilog/
- http://en.wikipedia.org/wiki/SystemVerilog

Sites internet vendeur PLDs

- http://www.achronix.com/
- http://www.actel.com/
- http://www.atmel.com/
- http://www.altera.com/
- http://www.cypress.com/
- http://www.latticesemi.com/
- http://www.quicklogic.com/
- http://www.siliconbluetech.com/
- http://www.xilinx.com/

Sites internet PLDs

- Site donnant la liste des vendeurs de PLDs
  http://www.fpgacentral.com/vendor/directory

- http://en.wikipedia.org/wiki/Programmable_logic_device
- http://en.wikipedia.org/wiki/Hardware_description_language
- http://www.doulos.com/knowhow

FIN présentation VHDL !

Questions