Designs complexes sur FPGA

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

ReDS
Reconfigurable & Embedded Digital Systems

1ère partie

Etienne Messerli

24 septembre 2010

Design complexes sur FPGA …

- Explosion des FPGAs de dernières générations
- Forte demande pour des traitements d'informations performants et ultras rapides :
  - ✓Quantité de données de plus en plus important (Go)
  - ✓Débit de transfert de plus en plus rapide (Gbits/s!)
  - ✓Applications temps réel:
    Voice over IP, vidéo, télévision, ...
  - ✓Utilisation de réseau public : Internet
    →Confidentialité des données => cryptographie
  - ✓…

… designs complexes sur FPGA

- Concevoir des systèmes de plus en plus complexes
- Intégrer ceux-ci dans des PLDs (performance)
- Garantir le fonctionnement correct du système
- Garantir l'évolution des systèmes réalisés
- Réduire le temps de la conception à la réalisation :
  Time to market

Réutilisation indispensable : Design re-use

Nouveau défi : Vérification

Design re-use

- Description lisible (viser la simplicité)
- Description facilement ré-utilisable (modifiable)
- Description bien documentée
  important : commentaires dans le fichier
- Fonctionnement fiable
- Synthèse automatique garantie pour tous les outils EDA du marché

Vérification

- Garantir le fonctionnement des designs
- Nouvelle méthodologie
- Nouveau langage: SystemVerilog

Cette partie sera traité par Yann Thoma

Contenu de la présentation

- Evolution des technologies des PLDs
- Méthodologies de conception
  - ✓évolution des méthodologies
  - ✓conception full-synchrone
- Rappel sur le VHDL pour la synthèse automatique
  - ✓paquetage Numeric_Std (Addition, comparaison)
  - ✓Instruction *process* et variable
  - ✓éléments mémoires et systèmes séquentiels
- Design re-use

Designs complexes sur FPGA

Evolution des technologies des PLDs

Evolution "Circuits logiques programmables"

- Depuis les années 1995 :

  Formidable évolution des
  circuits logiques programmables

- Caractéristiques principales :

✓densité	↗	1'600K DFF, RAM jusqu'à 50Mbits
✓fréquence	↗	jusqu'à 800 MHz
✓prix par *gate*	↘	1996 : ≈ 1 ct/gate 2004 : ≈ 0,01 ct/gate 2006 : ≈ 0,0001 ct/gate 2010 : diminue!

Evolution PLD : caractéristiques …

- technologie 65 nm, 40-35 et 28 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 (PLL with 8 output)
- fréquence jusqu'à 800 MHz
  - ✓ FPGA Achronix jusqu'à 1.5 GHz !
- nombre de *gates* jusqu'à ~20M
- nombre de flip-flops jusqu'à 1'640K
- plusieurs Mbits de RAM, jusqu'à 50 Mbits

… évolution PLD : caractéristiques

- LUT à 6 entrées au lieu de 4! (Virtex5)
  ALM (2 LEs) à 8 entrées => LUT à 7 entrées (Stratix IV et V)
- blocs pré-câblés (DSP, SERDES, …) => up to 600MHz
  - ✓jusqu'à 130 transceivers high speed (max 25 GigaBit/s)
  - ✓jusqu'à 3'500 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 septembre 2010 …

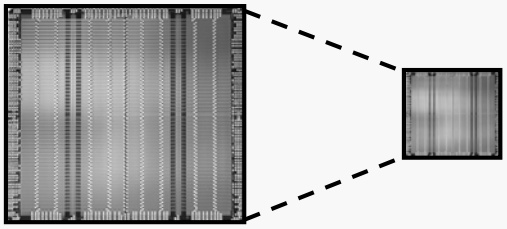
- PLDs de quelques francs à 11K francs !
- FPGA STRATIX-IVGT 530K, FBGA1152 : 11'000 $
  - ✓ 530K LEs, total RAM 27.3Mbits, 12 PLLs — 10'600 Kgates
  - ✓ 96 transceiver, 1'024 Multiplier 18x18,904 I/O — 0,001 $/gate
- *en mai 2008:*
  FPGA STRATIX-IV 150K, FBGA1152 : 11'000 $
  - ✓ *142K LEs, total RAM 7.3Mbits, 8 PLLs — 2'850 Kgates*
  - ✓ *380 Multiplier 18x18,744 I/O — 0,004 $/gate*
- FPGA Arria II GX260K, FBGA1152 : 2'160 $
  - ✓ 244K LEs, total RAM 11'756Kbits, 6 PLLs — 4'880 Kgates
  - ✓ 8 transceiver, 736 Multiplier 18x18, 612 I/O — 0,0044 $/gate
- FPGA Arria II GX45K, FBGA572 : 325 $
  - ✓ 43K LEs, total RAM 3'435Kbits, 4 PLLs — 860 Kgates
  - ✓ 8 transceiver, 232 Multiplier 18x18, 252 I/O — 0,0038 $/gate

… évolution PLD : prix septembre 2010

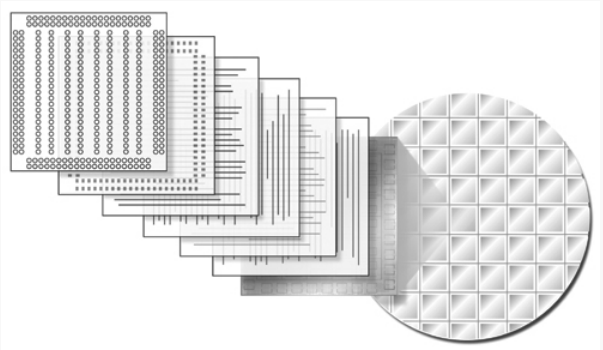
- FPGA CYCLONE-IV 119K, FBGA896 : 490 $
  - ✓ 150K LEs, total RAM 6'480Kbits, 8 PLLs — 3'000 Kgates
  - ✓ 360 Multiplier 18x18, 475 I/O — 0,00016 $/gate
- FPGA CYCLONE-III 5K, EQF144 : 13 $
  - ✓ 5K LEs, total RAM 414Kbits, 2 PLLs — 100 Kgates
  - ✓ 23 Multiplier 18x18, 94 I/O — 0,0001 $/gate
- CPLD MAX-II 2210 LE, FBGA256 32 $
  - ✓ 2210 LEs, Flash memory 8Kbits, 0 PLLs — 44 Kgates
  - ✓ 0 Multiplier 18x18, 80 I/O — 0,0007 $/gate
- CPLD MAX-II 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



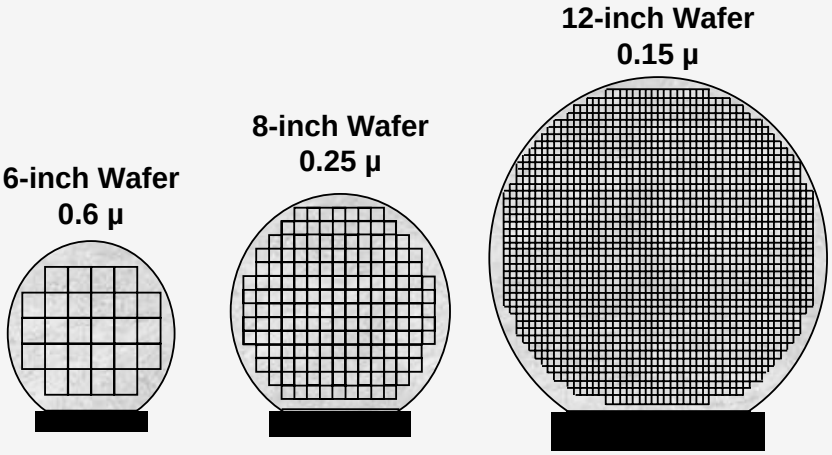
- **Drawbacks**
  - Higher Risks
  - Higher Invest Costs



Wafer/Die Sizes/Quantities

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

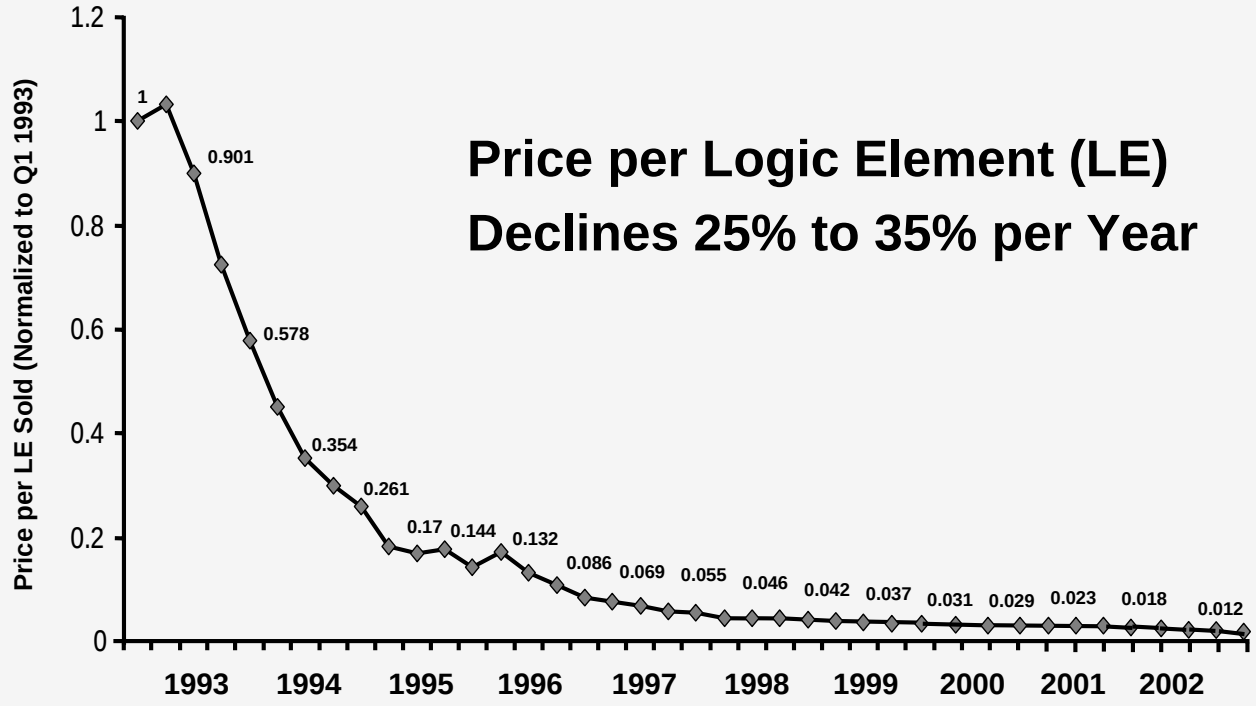
Toujours actuel:
wafer: 12 inch/300mm



Typical Net Die Per Wafer:

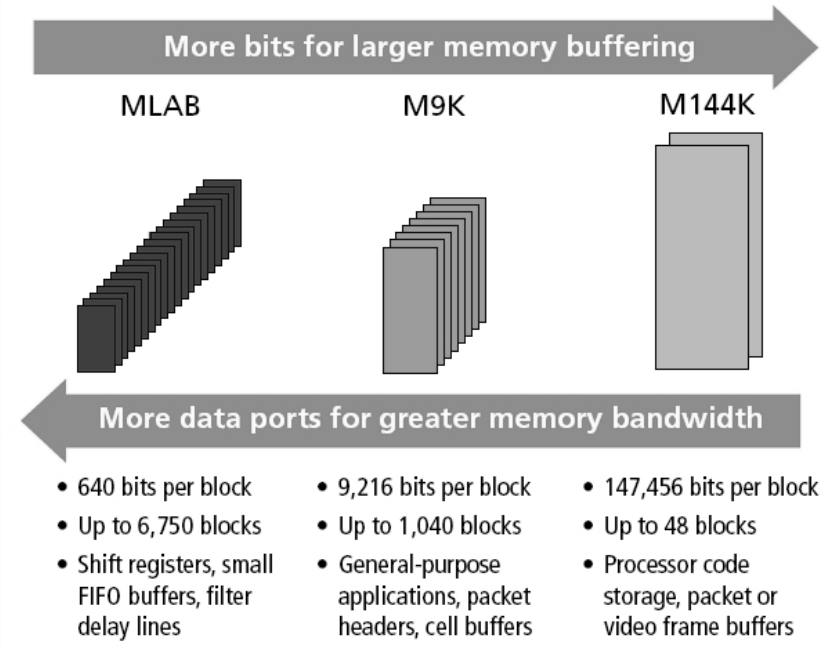
1x ***30x*** ***200x***

Price Trend



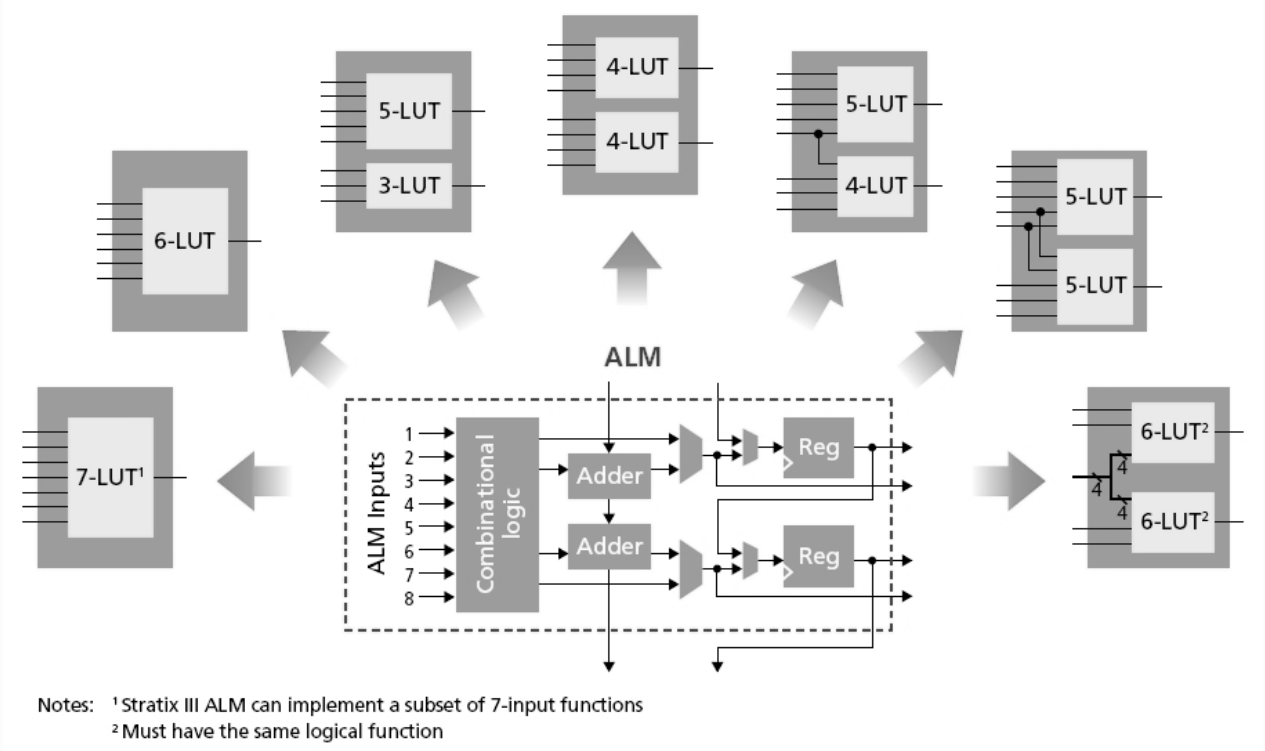
Price per Logic Element (LE)
Declines 25% to 35% per Year

TriMatrix memory



Stratix III, 2008

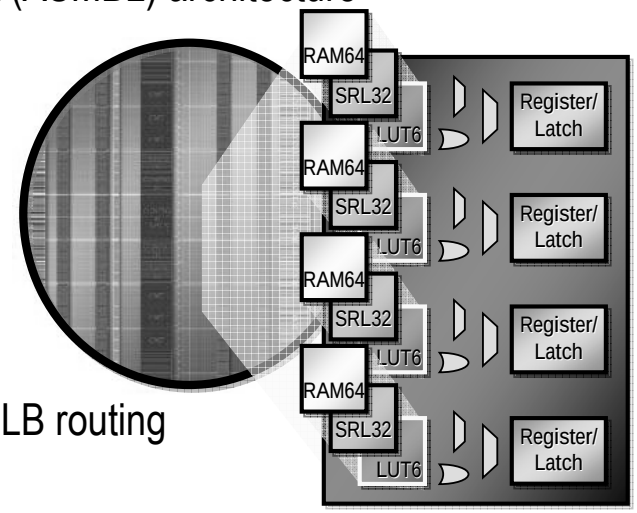
ALM block diagram example LUT configurations



Stratix III, 2008

Virtex-5 Logic Architecture

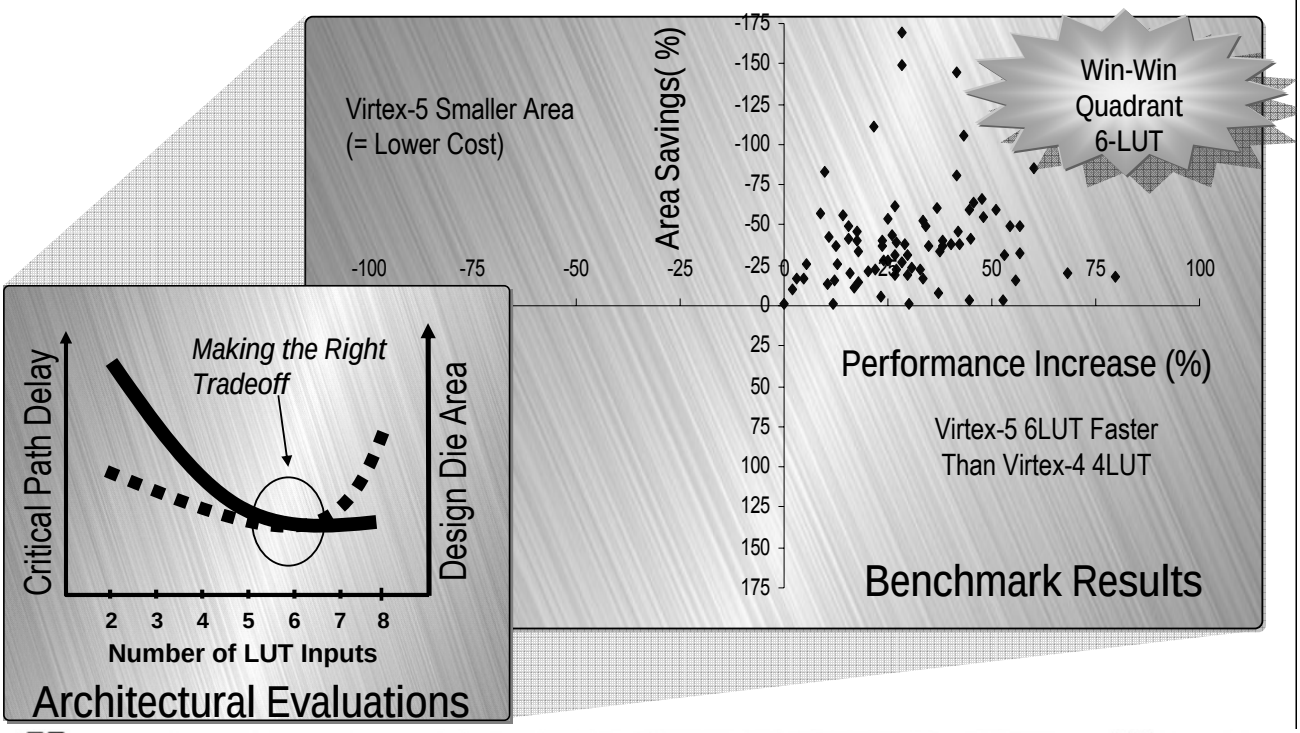
- Second-generation column-based Advanced Silicon Modular BLock (ASMBL) architecture
- Advanced logic structure
  - ✓ True 6-input LUTs
  - ✓ Exclusive 64-bit distributed RAM option per LUT
  - ✓ Exclusive 32-bit or 16-bit x 2 shift register
- More efficient and flexible inter-CLB routing
  - ✓ Increased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



Virtex-5 is the flagship of the FPGA industry

Optimal Performance/Area Trade-off

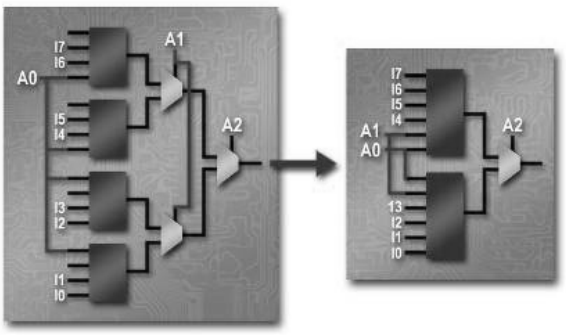
6-Input LUT Yields Best Results at 65nm



Logic Compaction with LUT6

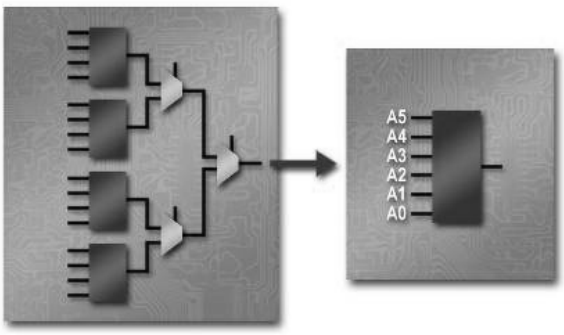
Use Fewer LUTs, Faster, Less Routing

8 to 1 Multiplexer



LUT4 LUT6

64 bit RAM



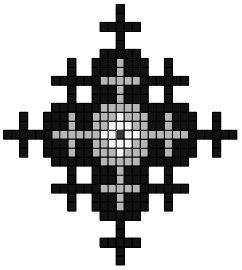
LUT4 LUT6

Interconnect Architecture

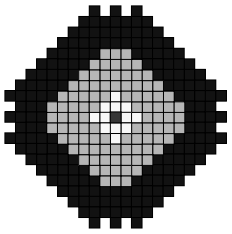
Reduce Routing Delay

- 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



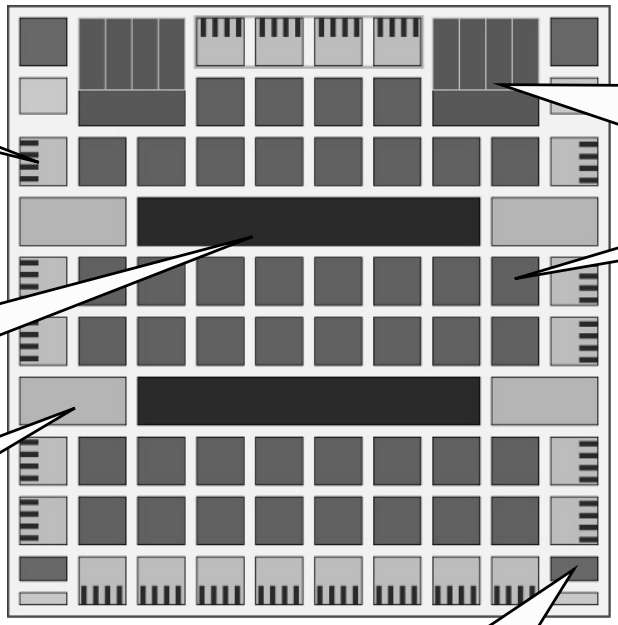
CLB

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

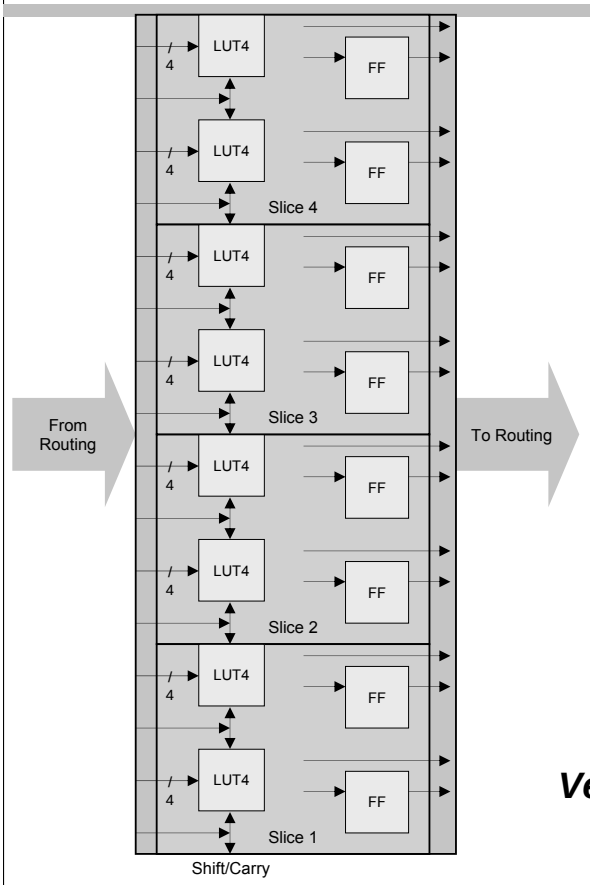
MACO: Embedded Structured ASIC Blocks (LatticeSCM Devices)

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

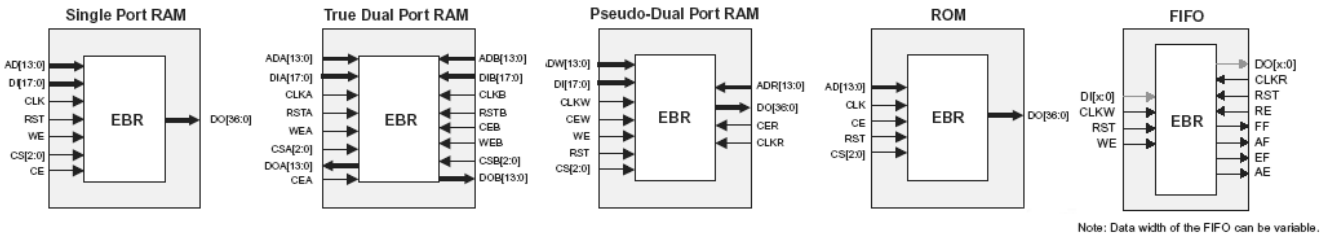
Programmable Function Unit (PFU)

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

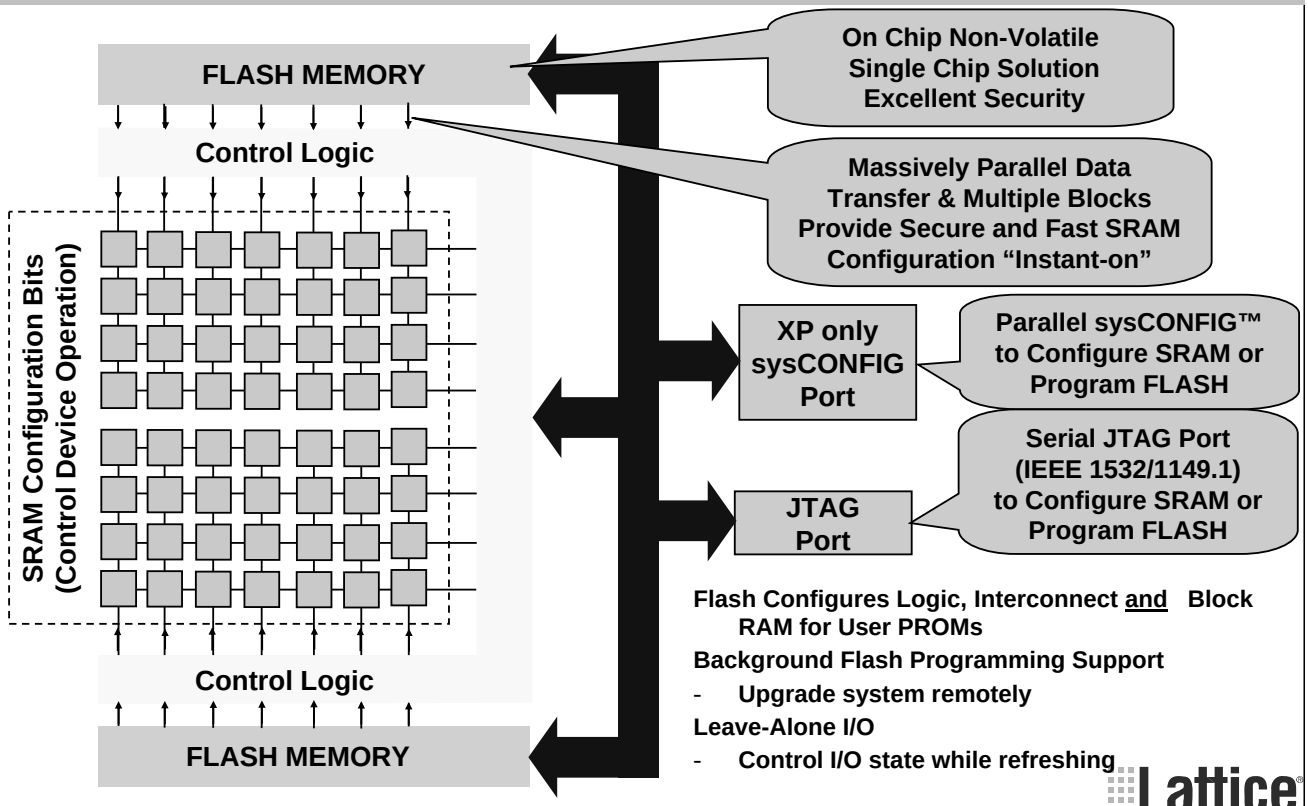
Flexible Configuration Capabilities:



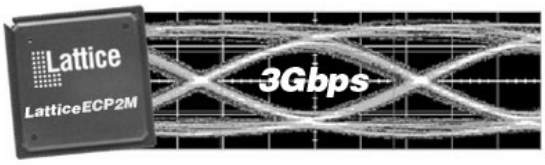
Large Memory Capacity to Support Fast, Flexible RAM for Packet Buffering, Clock Domain Transfers, and Context Memory

Lattice Semiconductor Corporation
Bringing the Best Together

Programmable Function Unit (PFU)



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

FPGA Altera: Stratix V new

Pas encore livrable

- Technologie 28 nm
- Stratix V devices (FPGAs with transceivers)
  - ✓ Up to 1'090K logic elements (LEs), 1'640'000 DFF,
    Up to 16 global/ 92 regional clocks, up to Up to 50 MB RAM
    48 full-duplex transceiver (SERDES) at up to 25 Gbps,
    210 High-Speed LVDS channel at up to 1.4 Gbps
- Stratix V GT B7
  - ✓ 622K logic elements (LEs), 939'000 DFF, 50Mbit RAM,
    transceiver 4 x 28 Gbps/ 32 x 12,5 Gbps
- Stratix V E devices (enhanced FPGAs)
  - ✓ 1'090K LEs, 1'640'000 DFF, 43Mbit RAM,
    no tranceiver at 28/12,5 Gbps

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
  - ✓ Power up to 40 A at 0.95 V !
- 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 Altera: Arria II

- Technologie 40 nm
- Arria II GX devices (FPGAs with transceivers):
  - ✓up to 244K logic elements (LEs), 500K DFF,
  - ✓736 18 x 18-bit multipliers,
  - ✓1 PCI Express Hard IP Blocks
  - ✓up to 16 full-duplex 6.375-Gbps transceivers,
  - ✓up to 11.8 Mbits RAM
  - ✓up to 612 I/Os

FPGA Altera: Arria II

- Technologie 40 nm
- Arria II GZ devices (FPGAs with transceivers) new
  - ✓up to 348K logic elements (LEs), 205'200 DFF,
  - ✓1'040 18 x 18-bit multipliers,
  - ✓1 PCI Express Hard IP Blocks
  - ✓up to 24 full-duplex 6.375-Gbps transceivers,
  - ✓up to 16.4 Mbits RAM
  - ✓up to 726 I/Os

FPGA low cost : Cyclone III & IV

- Voir présentation Altera
  - ✓ Présentation Altera Cyclone III et IV

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

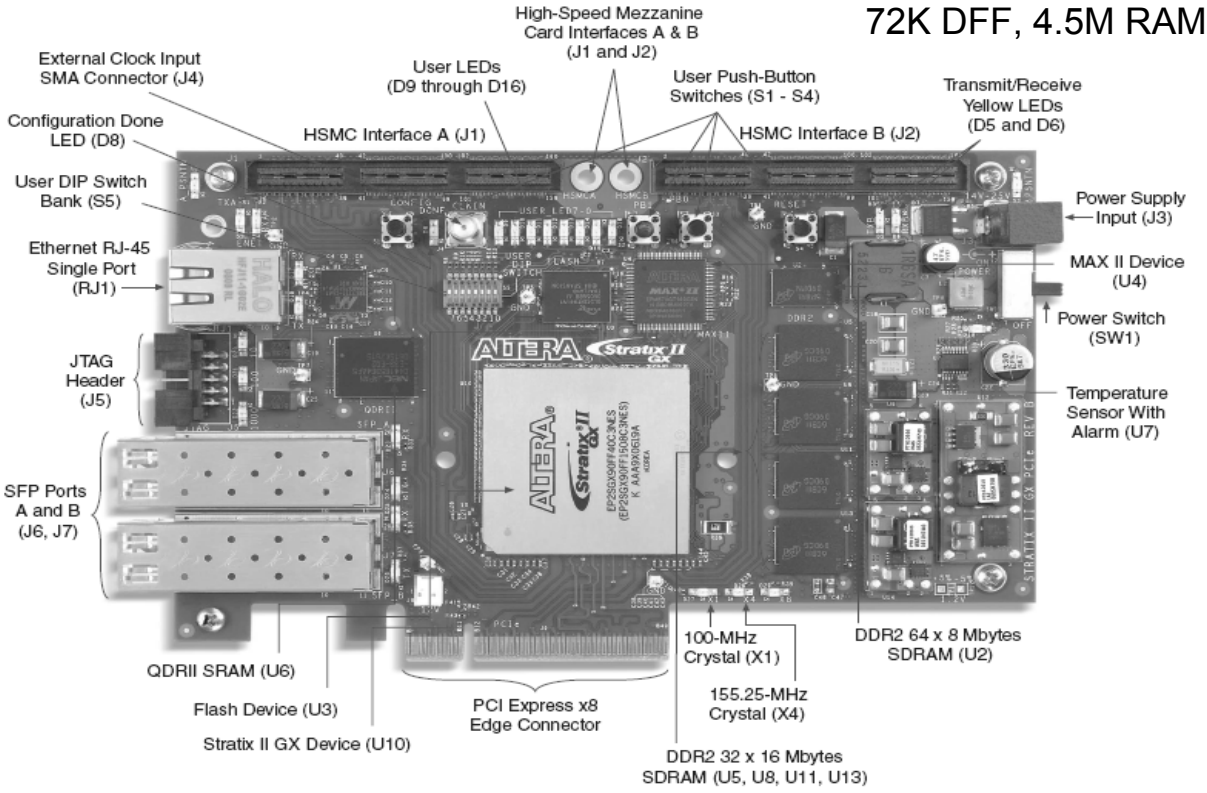
Info site "www.achronix.com", janvier 2009

Carte de développement FPGA

Figure 2–1. Stratix II GX PCI Express Development Board

FPGA: EP2SGX90FF

72K DFF, 4.5M RAM

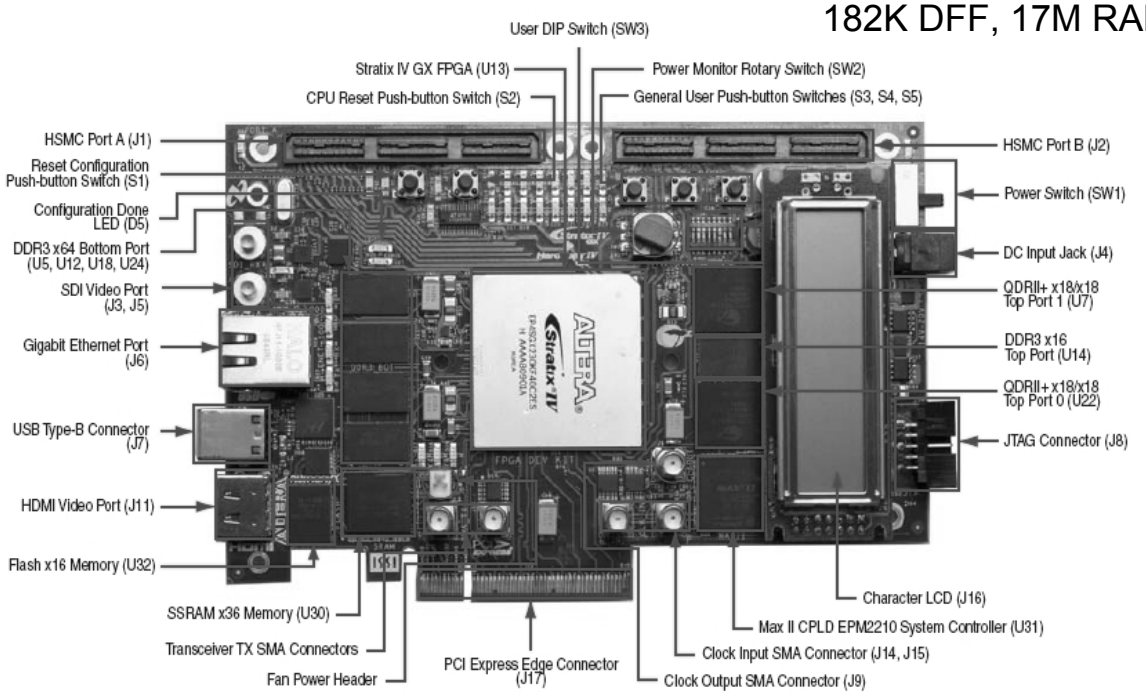


Carte de développement FPGA

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

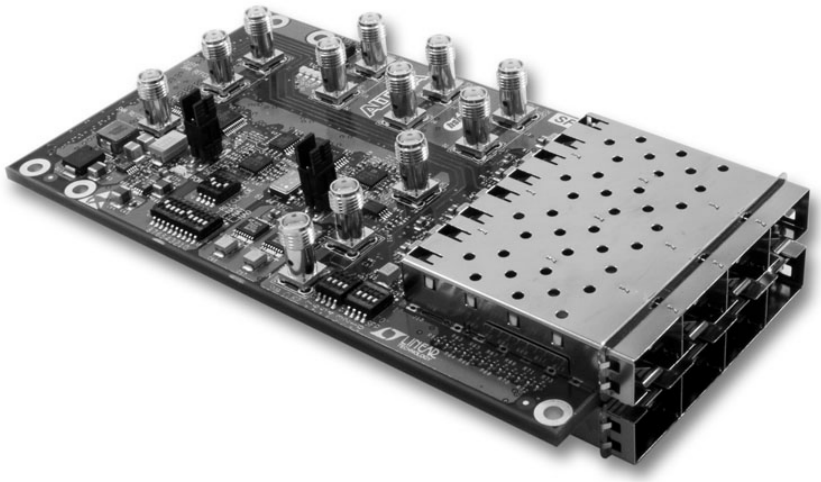
FPGA: EP4SGX230

182K DFF, 17M RAM



Carte de développement & FPGA

- Carte d'extension avec 8 liaisons séries par fibre optique: SFP HSMC Board with 8 optical link, Terasic



Technologie des PLDs

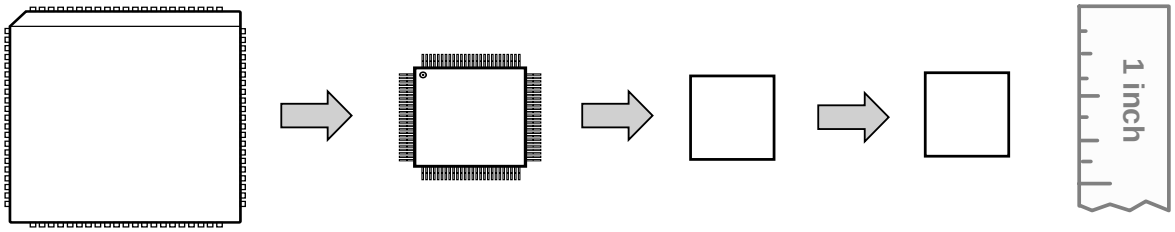
- Les PLDs utilisent de nombreuses technologies différentes :
  - ✓ Volatile ⇒ SRAM
  - ✓ Reprogrammable ⇒ EEPROM
    EEPROM Flash
    SRAM + EEPROM intégrée } évolution
  - ✓ Programmable une seule fois ⇒ Anti-fuse, Via-link

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

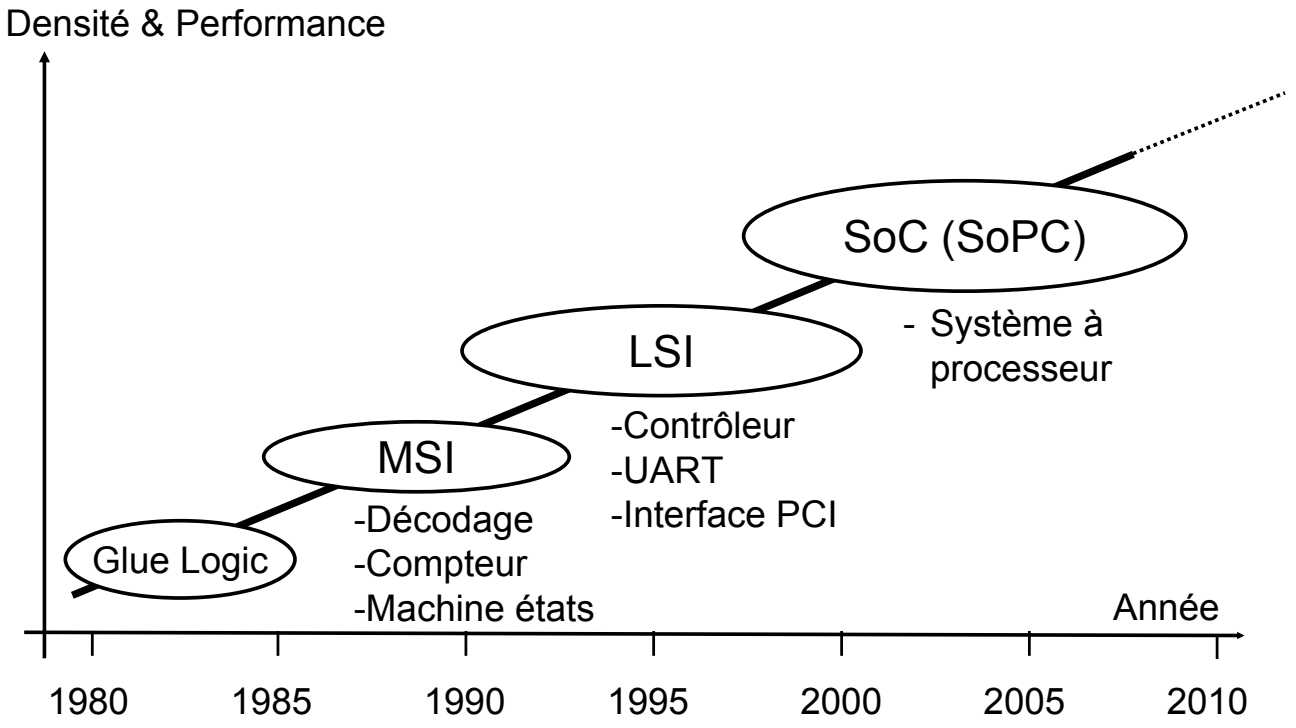
Evolution des boitiers

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

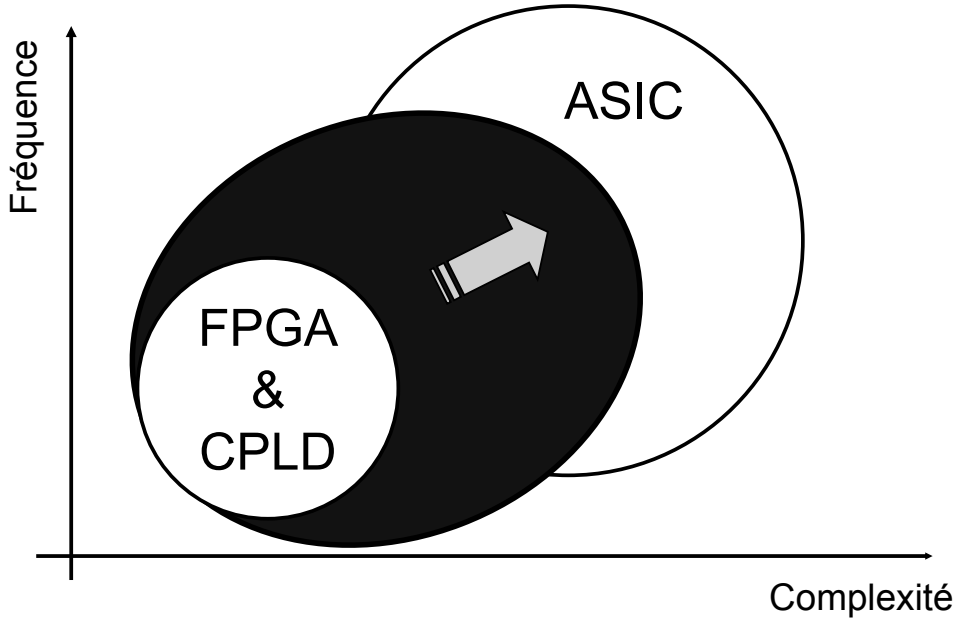


	84-Pin PLCC	100-PQFP	100-FBGA	169-UBGA
Surface relative	100%	23%	12%	5%

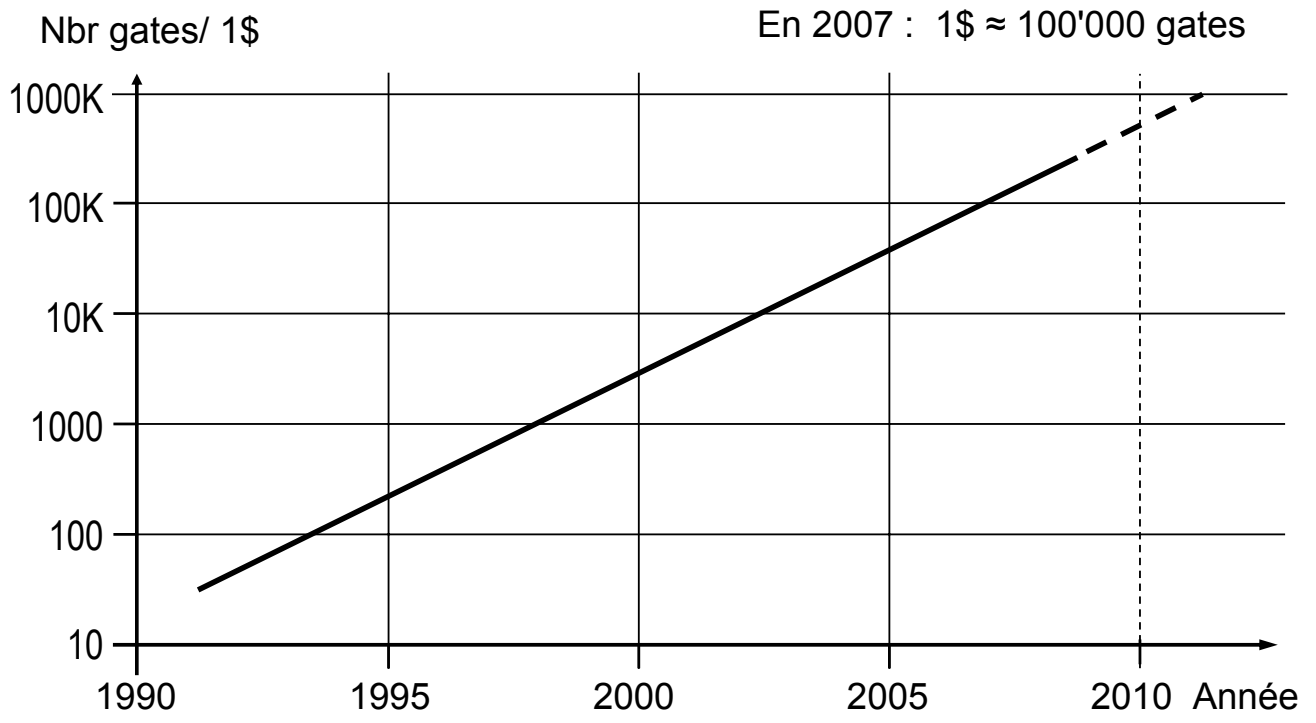
Domaine d'utilisations des PLDs



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

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"
    (1 pièce $99 sept 2010)

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.

FPGA low cost : Cyclone III

- Technologie SRAM
- EP3C10F256C8
  - ✓ 23.5 $
  - ✓ 182 IOs
  - ✓ 10'320 LEs
  - ✓ 10'320 DFF
  - ✓ 23 multiplicateurs, 2 PLLs
  - ✓ env 200'000 gates (80 x EPM128!)
  - ✓ 414Kbits memory
  - ✓ 400 MHz
  - ✓ prix ~ 0.0002 Fr/gate

Boitier 256-FBGA

Prix Cyclone III, ALTERA

- Info septembre 2010
- Cyclone III, EP3C25
  - ✓ 24,624 LEs
  - ✓ env 500'000 gates
  - ✓ 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 II GX, ALTERA

- Info septembre 2010
- Arria II GX, 40 nm
  - EP2AGX45
    - 43K LEs
    - 56 multiplicateurs
    - 3'435Kbits de RAM
    - 4 x PLLs
    - 8 x transceiver 6.375Gbps
  - EP2AGX45DF25C6
    - boitier 572 FBGA
    - 1 pièce $325

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

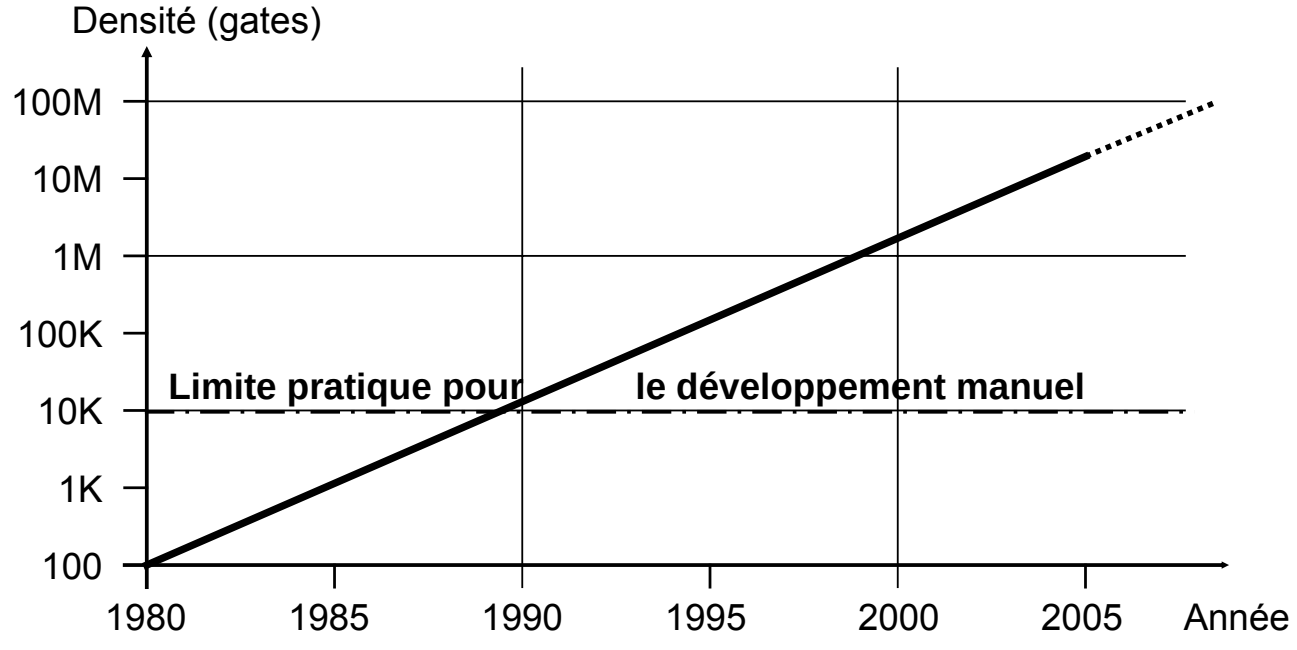
Feature	EP2AGX45	EP2AGX65	EP2A
Total Transceivers	8	8	1
ALMs	18,050	25,300	37,
LEs	42,959	60,214	89,
PCIe hard IP blocks	1	1	
M9K Blocks	319	495	6
Total Embedded Memory in M9K Blocks (Kbits)	2,871	4,455	5,5
Total On-Chip Memory (M9K + MLABs) (Kbits)	3,435	5,246	6,6
Embedded Multipliers (18 × 18) *(1)*	232	312	4
General Purpose PLLs	4	4	

page volontairement laissée vide

Designs complexes sur FPGA

Evolution des méthodologies de conception

Densité des PLDs (loi de MOORE)



Complexité et coûts...

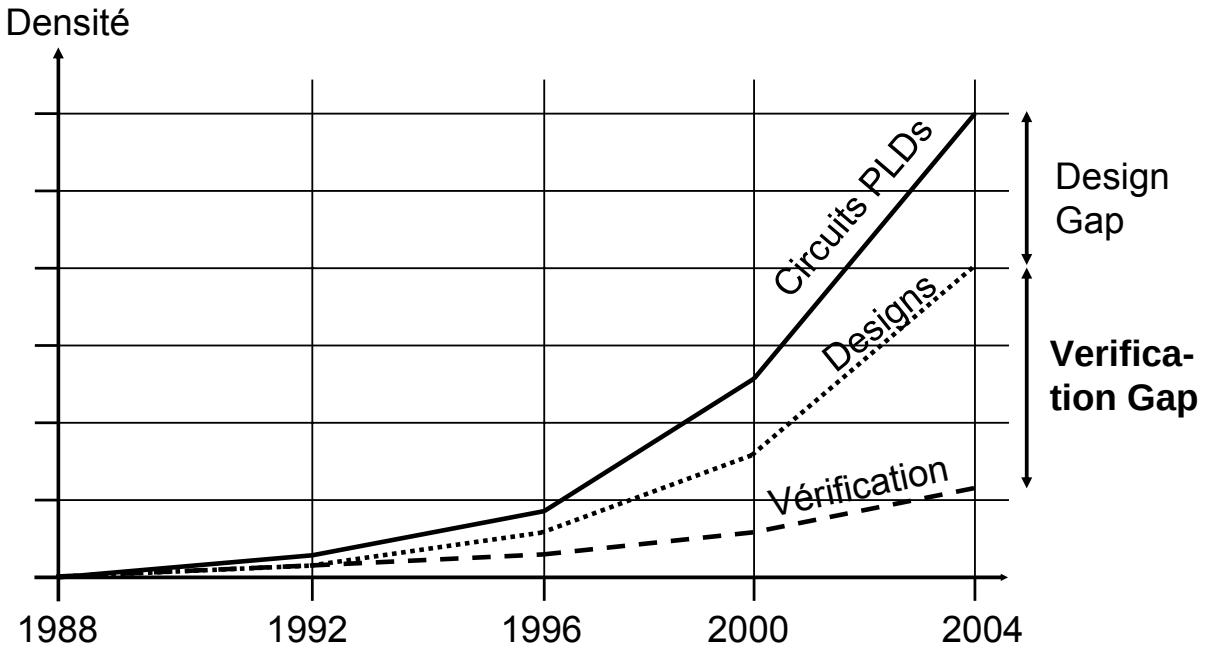
- Nouveau développement:
  - ✓PLD indispensable (CPLD, FPGA, HardCopy, ASIC)
- Niveau de complexité augmente rapidement :
  - ✓dizaines vers centaines de milliers de portes, voir millions de portes
- Développement "à la main" trop difficile et trop coûteux :

  outils informatiques performants, méthodologies et bibliothèques de "pièces" sont indispensables

Méthodologie de conception

- Conception *Top - Down*
- Décomposition du système
  - ✓choisir l'architecture
  - ✓modules de bases doivent être simples
  - ✓réutilisation des descriptions
- Conception *full synchrone*
  - ✓fiabilité, testabilité
- Vérification de chaque module

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 blocs IPs (Intellectual Property)
- Assurer la maintenance, l'évolution des designs
  - ✓lisibilité, rigueur, fiabilité des descriptions
- Garantir la vérification des designs

Vérification : le nouveau défi …

- Dépannage d'un PLD programmé :

  très coûteux voir impossible

- Simulation indispensable,

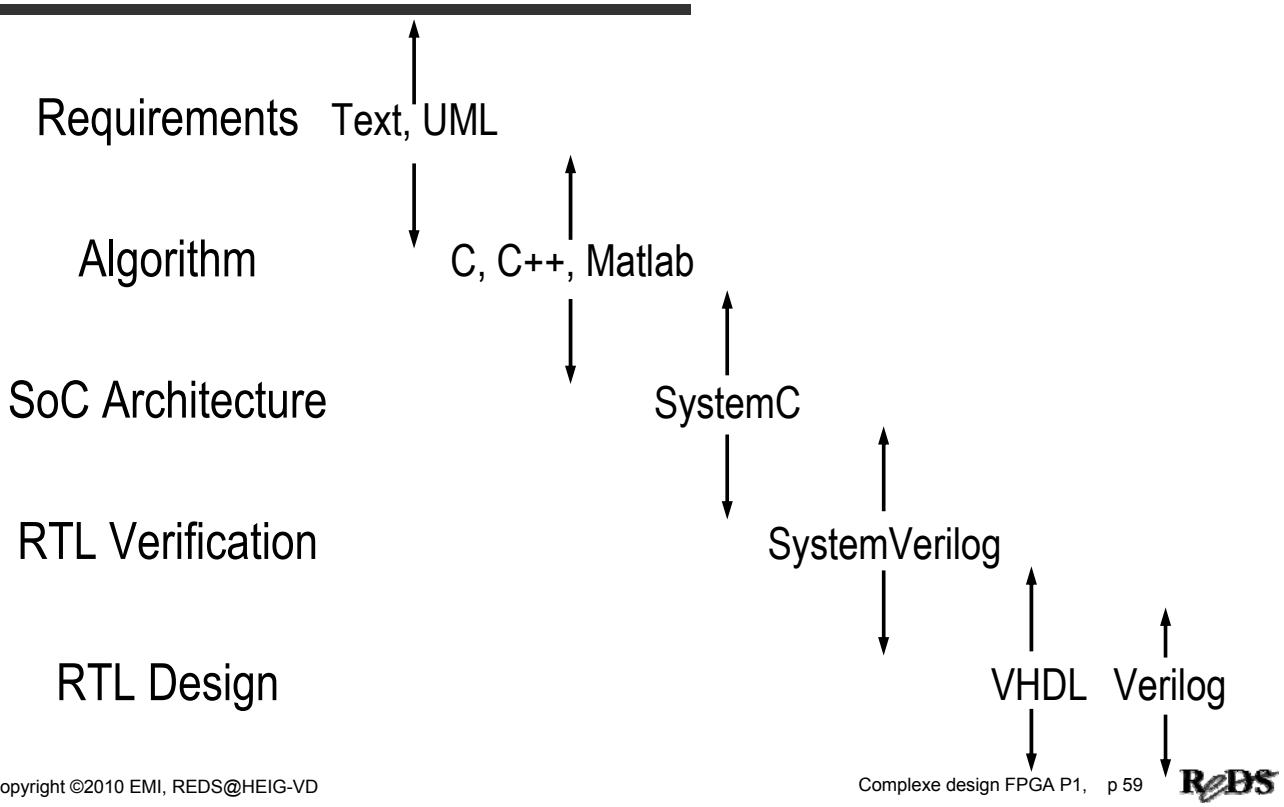
  nécessite :

  - penser vérification lors de la conception
  - outils performants
  - fichiers de simulation automatique

… vérification : le nouveau défi …

- Vérifier la conformité avec le cahier des charges
- Maîtriser les coûts de la vérification
  - ✓représente env. 70% du temps de développement!
- Détecter les erreurs au plus vite
  - ✓règle des x10
- Assurer fonctionnement après la synthèse et le placement-routage

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 du langage Verilog
  - ✓ pratique d'utiliser des fonctionnalités de SystemVerilog

Langages pour la vérification

Projets de petites à moyennes tailles:

- ✓Utilisation du langage VHDL

Projets de tailles moyennes à élevés, nécessité de :

- ✓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, …

La portabilité

La portabilité est le maître mot. Dans le monde des circuits ce mot a un double sens :

- portabilité vis-à-vis des circuits.
- portabilité vis-à-vis des outils.

Citation de [2] .

Solution :

Nouvelles méthodes de conception associées avec un langage de haut niveau

soit :

VHDL (préféré en Europe)

ou Verilog (préféré aux USA)

\+ Outils de développement modernes

Bricolages interdits !!

Historique du langage VHDL

- VHDL signifie :
  - ✓VHSIC Hardware Description Langage
    (Very High Speed Integrated Circuit)
- Développé par le DOD (années 1980)
  (Département Américain de la défense)
- Normes IEEE: 1076 (1987, 1993),
  1164 (1993), 1076.3 (1997), 1076.6 (1999)
  (Institute of Electrical and Electronics Engineers)

Langage VHDL pour la synthèse

- Définition d'un standard pour la synthèse :
  - ✓norme IEEE-1164, 1987, révision en 1993
    paquetage Std_Logic_1164
    (définit le type Std_Logic)
  - ✓norme IEEE-1076.3, 1997
    paquetages Numeric_Bit & Numeric_Std
    (représentation des nombres en binaire
    pour les opérations arithmétiques)
  - ✓norme IEEE 1076.6, 1999
    Standard for VHDL Register Transfer Level Synthesis
    Sous-ensemble **synthétisable** du VHDL !!!!

Norme IEEE-1076 : 2000

- Principale différence est le nouveau mot réservé en 2000 : protected
  - ✓ne pas utiliser celui-ci comme identificateur
- Actuellement la norme 1993 est la plus utilisée. Elle est supportée par la majorité des outils

Norme IEEE-1076 : 2008

- Evolution importante du langage
  - ✓Quelques améliorations pour la synthèse
  - ✓Améliorations très importantes pour la vérification
  - ✓Norme publiée
  - ✓Livre "VHDL 2008"
- Pas d'outils EDA mis à jour avec cette nouvelle norme
- Arrivée trop tardive de la norme IEEE-1076 2008

Pourquoi utiliser le VHDL ?

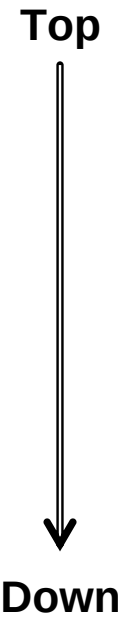
- Normalisé (pérennité)
- Moderne, lisible et puissant
- Permet la réutilisation de bloc (IP)
- Disponible avec beaucoup d'outils
  - ✓outils de 0.- à 100'000.- et plus
- Compatible avec le domaine ASIC

Un langage incontournable en synthèse

Représentations de fonctions logiques :

- Schéma-bloc
- Algorithme, flot de données
- Table de vérité
- Equation logique
- Schéma logique (portes, fonctions courantes)
- Schéma électrique (transistors, résistances, etc)
  (PLD : table de fusible)
- Photographie du circuit (chip ou carte)

VHDL



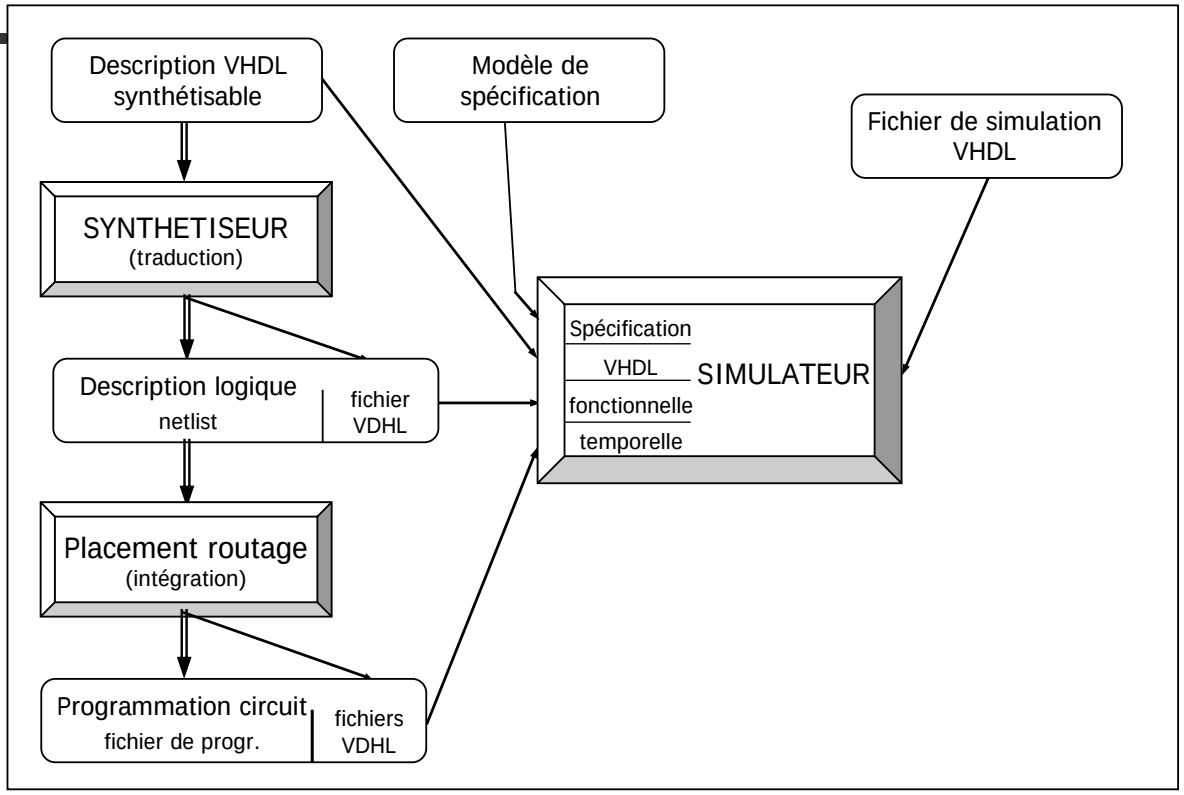
Niveaux de description en VHDL

- Haut niveau comportemental
  - ✓non synthétisable actuellement
- RTL Register Transfert Level (inclu schéma bloc)
  - ✓synthétisable
- Netlist de primitive
  - ✓résultat obtenu après synthèse
- niveau porte
  - ✓résultat obtenu après placement-routage

Applications du langage VHDL

- Spécification (cahier des charges, doc)
- Conception (comparaison de différents algorithmes ou solutions)
- Description pour la synthèse automatique
- Simulation fonctionnelle
- Vérification de contraintes temporelles
- Preuve formelle

Réalisation d'un circuit : *Design flow*



Caractéristiques des deux outils

- Le simulateur :
  - ✓interprète le langage VHDL
  - ✓comprend l'ensemble du langage VHDL
- Le synthétiseur :
  - ✓traduit la description VHDL en une *netlist* logique
  - ✓comprend uniquement un sous-ensemble du langage VHDL

Objectif de la méthodologie

Description VHDL simulée
avant et après synthèse, l'objectif est :

même fonctionnement dans les deux cas

Une méthodologie est indispensable

Le troisième outil

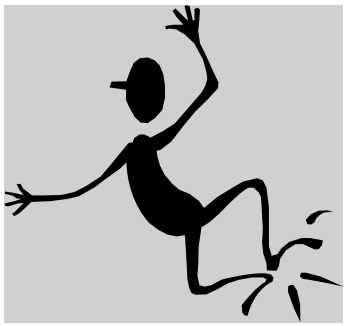
Finalement la *netlist* obtenue doit être intégrée dans le circuit cible.

- C'est l'outil de placement et routage
  - ✓celui-ci est fourni par le fabricant du circuit
  - ✓les temps de propagations seront déterminés

Le fonctionnement logique
n'est pas modifié durant cette étape

Finalement .....

Les applications les plus complexes
sont possibles dès maintenant



Designs complexes sur FPGA

Méthodologies de conception

Préambule

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

- Dans le cas de réalisation avec un ASIC ou de design asynchrone les méthodologies et les règles à utiliser peuvent être différentes

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

Complexité ?

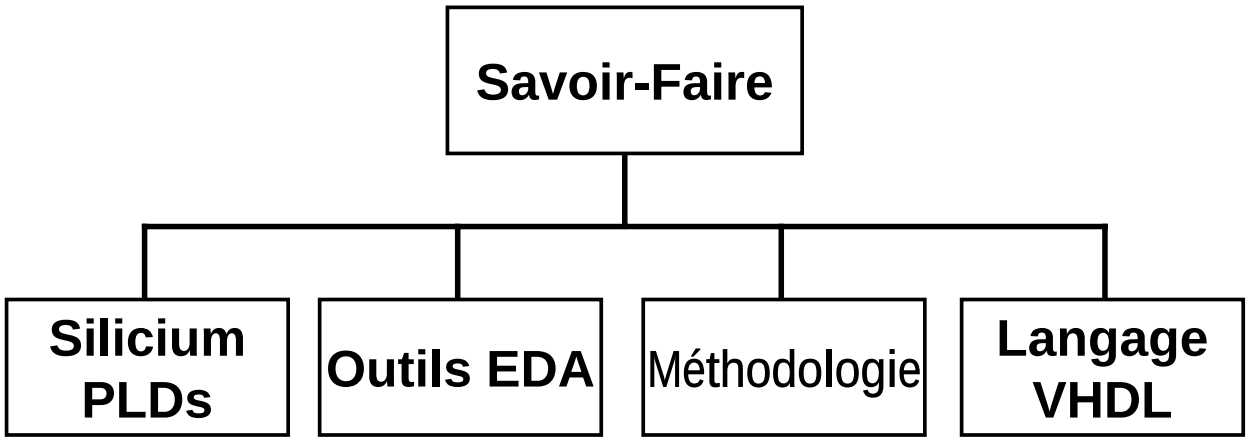
Qu'est-ce que la complexité d'un design ?

- Complexité ≠ Densité

Quelques facteurs aggravants :

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

Les clés du succès …



.. les clés du succès : Savoir-faire …

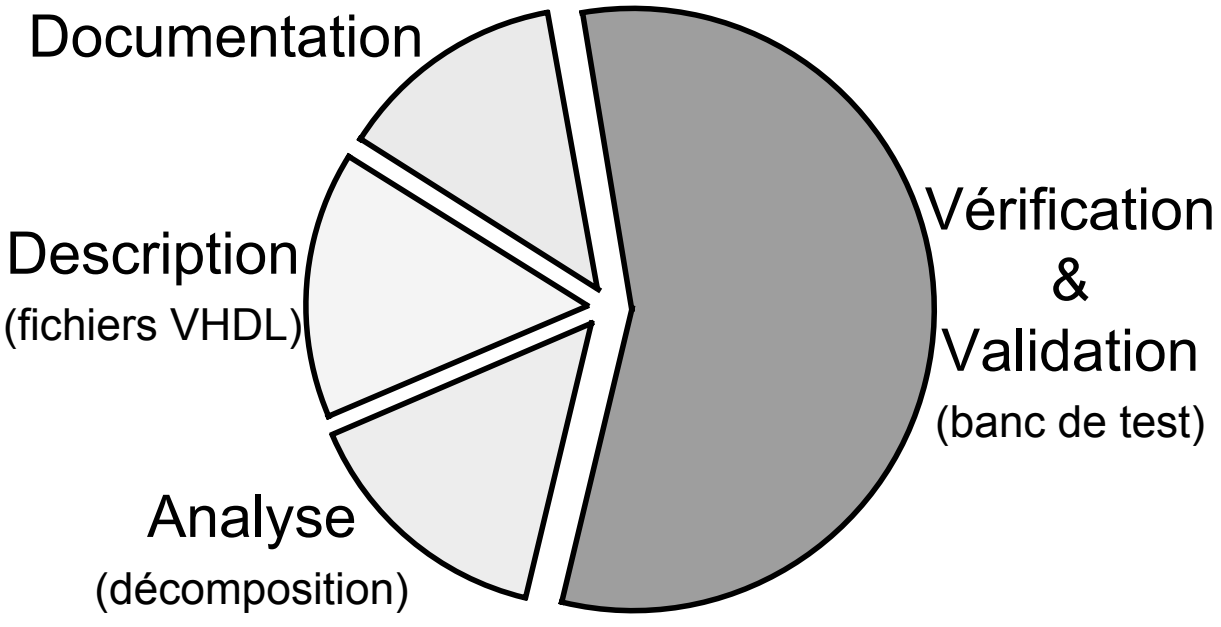
- Silicium : PLDs (FPGA ou CPLD)
- Outils EDA performants et modernes
- Langage de description HDL (VHDL ou Verilog)
- Méthodologie de conception
  - ✓décomposition, architecture
  - ✓full synchrone, vérification statique des timings
  - ✓modularité, ré-utilisation (IPs)
  - ✓méthodologie de description : fiabilité, synthèse garantie

    →définir un cahier méthodologique pour l'entreprise

… 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 (IPs)
- Abstraction
  - ✓permet d'augmenter la productivité des descriptions
  - ✓imaginer l'interaction entre les blocs

Phases d'un projet



Méthodologie de conception …

- Etapes de conception et de réalisation:
  - ✓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 (VHL)
  - ✓rigueur indispensable
  - ✓description uniforme => facilite la portabilité entre collègues
  - ✓cahier méthodologique devrait être définie dans chaque entreprise

Généralités

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

<Dir_Projet>\src	fichiers sources design (VHDL)
...\src_tb	fichiers sources test bench
...\work	fichiers compilé simulation
...\synth	fichiers pour la synthèse
...\p_r	fichiers pour le placement & routage

Conception et module VHDL

- Décomposer votre projet
- Une seule fonction par module VHDL
- Vos descriptions doivent être simples et lisibles
- Les descriptions doivent être facile à comprendre pour le synthétiseur
  $\Rightarrow$ permet une meilleure optimisation lors de la synthèse

Mots réservés et identificateurs

Convention au 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`, ...

  VHDL: uniquement des lettres, des chiffres et le souligné

Convention pour les identificateurs …

Convention au REDS

- Déclaration au top :

  `n<Nom_Signal>_o` pour 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_Generic>_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

… convention pour les identificateurs

Convention au REDS

- Dans les fichiers de simulation (test-bench) :

  ✓Nom des signaux de stimuli (entrées) :

```
signal <Nom>_sti : Std_Logic;
```

  ✓Nom des signaux observés (sorties) :

```
signal <Nom>_obs : Std_Logic;
```

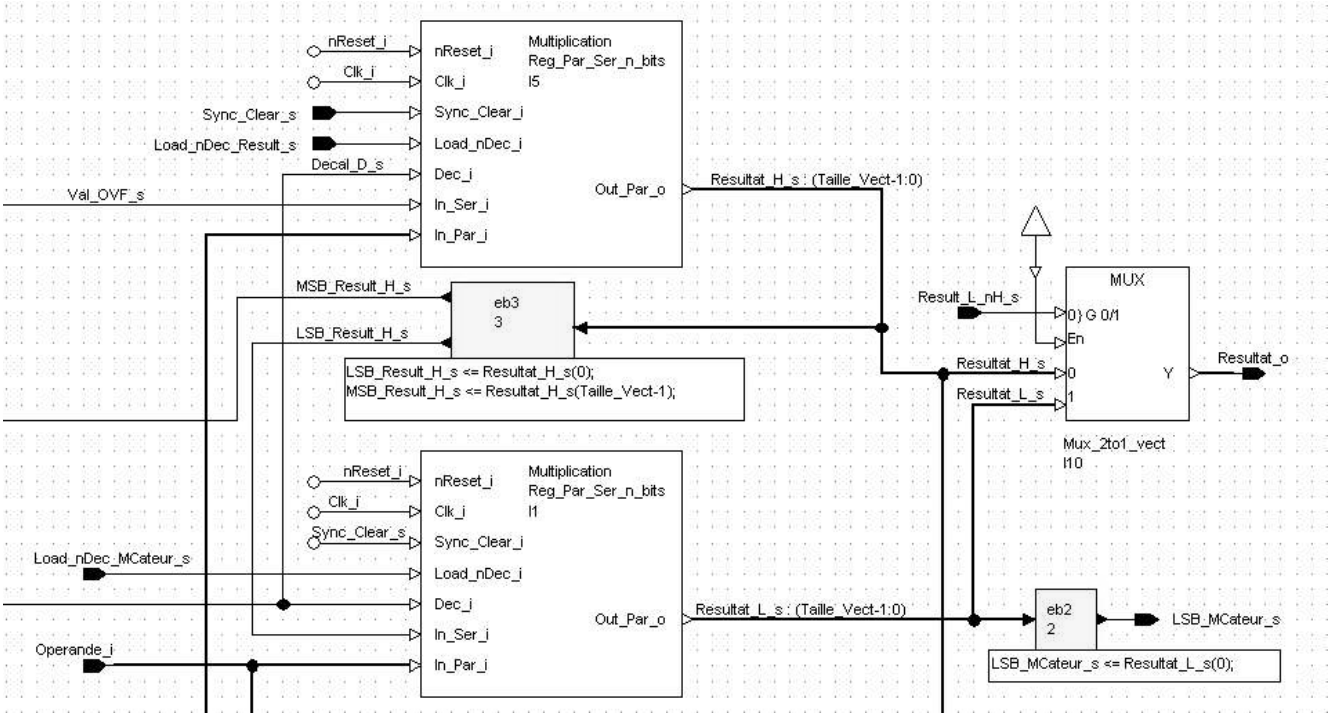
  ✓Nom des signaux de référence (calculés) :

```
signal <Nom>_ref : Std_Logic;
```

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
- Nom fichier VHDL :
  - ✓uniquement : lettres, chiffres, underscore
  - ✓identique au nom de l'entité
- Fichiers VHDL de simulation : <nom>_tb.vhd

Entête des fichiers VHDL

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

Au 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 :

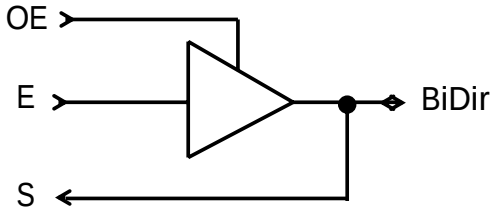
  `Std_Logic, Std_Logic_Vector`

… 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'.



Polarité des signaux

- Utiliser uniquement des signaux en logique positive à l'intérieur des descriptions.
- Adapter la polarité des signaux au top du projet.
- Indiquer les signaux en logique négative.
  - ✓exemple : nReset_i

Types des signaux autorisés

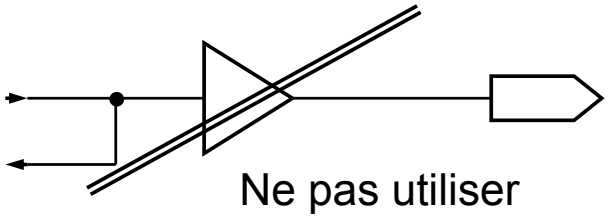
- Types utilisés dans l'entité (interconnexion entre module) : uniquement `Std_Logic` et `Std_Logic_Vector`
- Types utilisable à l'intérieur d'un module :
  - ✓tous les types sont autorisés
  - ✓types recommandés : `Std_Logic, Std_Logic_Vector et Unsigned, Signed` (opérations arithmétiques).
  - ✓les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

Types de *port* …

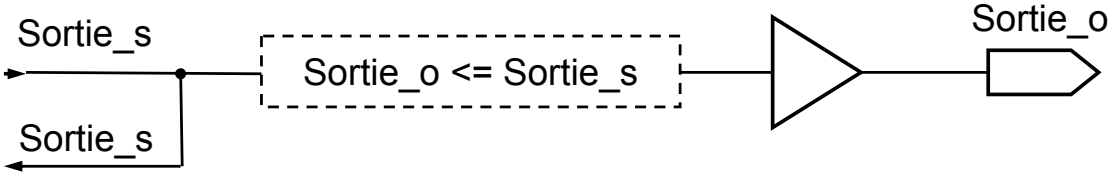
- Utiliser : **`in`**, **`out`**, **`inout`**.
- Le type **`inout`** sera utilisé uniquement dans le module top. Il n'y a pas de ligne trois états à l'intérieur d'un PLD (sauf cas particulier!).
  - ✓possible dans le cas des ASICs
- Le type **`buffer`** ne sera jamais utilisé (voir remplacement).

… 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 Vect32 : Std_Logic_Vector(0 to 31);
```

alors il y a inversion des bits lors de l'affectation

```
Vecteur <= Vect32; --Tous les bits sont croisés !!
```


Bibliothèques

- Bibliothèque IEEE autorisées :
  - ✓`Std_Logic_1164` types et fonctions de base.
  - ✓`Numeric_Std` opérateurs arithmétiques.
- plus bibliothèque propre à l'entreprise :
  - ✓à utiliser avec modération (gestion des versions).
  - ✓fonctions et procédures pour la simulation.
  - ✓etc.

Designs complexes sur FPGA

Conception full synchrone

Pourquoi "full synchrone" ?

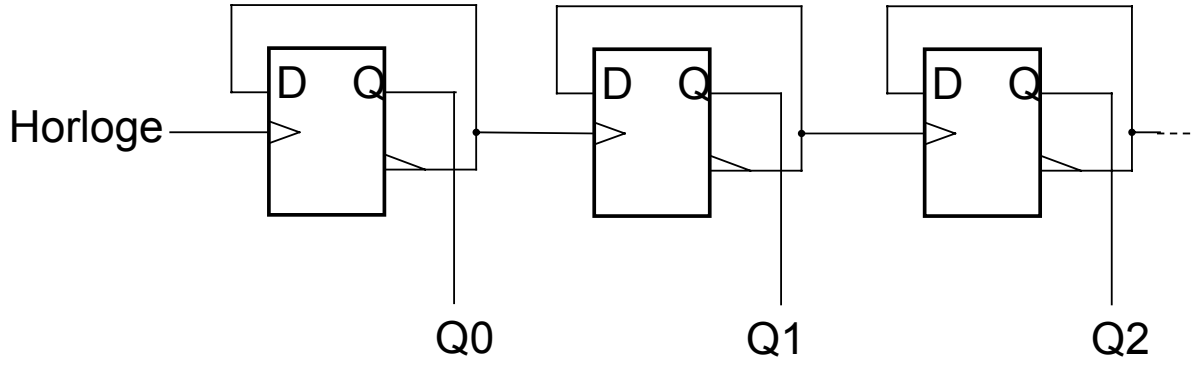
- Avantages conception full synchrone :
  - ✓permet une abstraction, facilite la conception
  - ✓évolution prédictible, changement vu au prochain flancs
  - ✓retard identique pour chaque sortie de flip-flop
  - ✓analyse statique permet de déterminer la fréquence maximum de fonctionnement
  - ✓arbre d'horloge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

Type de conception

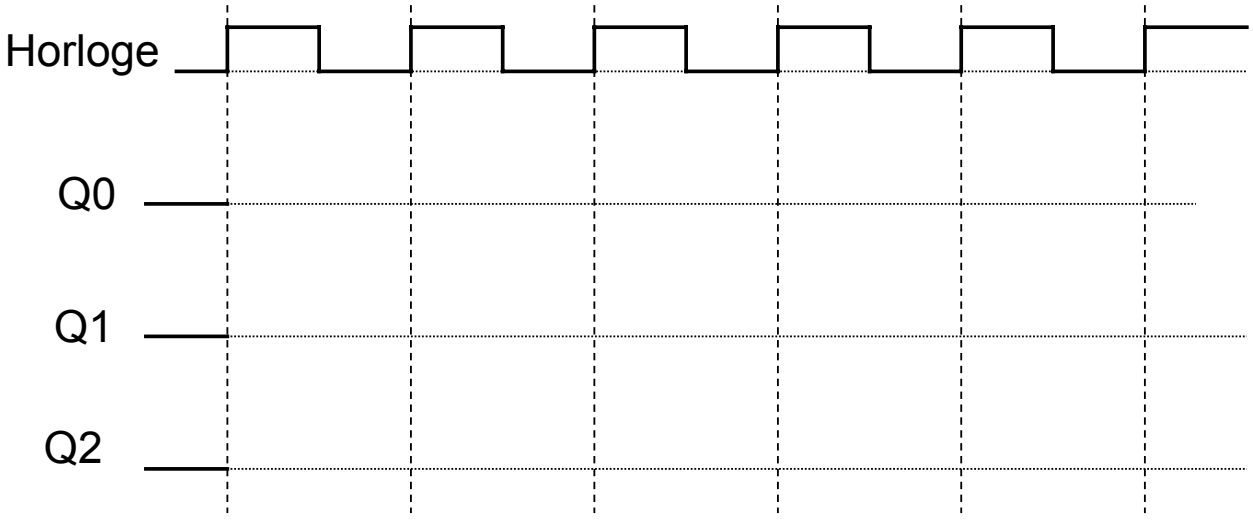
- Pseudo-synchrone
  - ✓Horloge d'un flip-flop dépend du ou des flip-flops précédents
- Full synchrone
  - ✓Toutes les flip-flops sont synchronisées par le même signal d'horloge

page volontairement laissée vide

Compteur pseudo synchrone (ripple counter)



Compteur pseudo synchrone

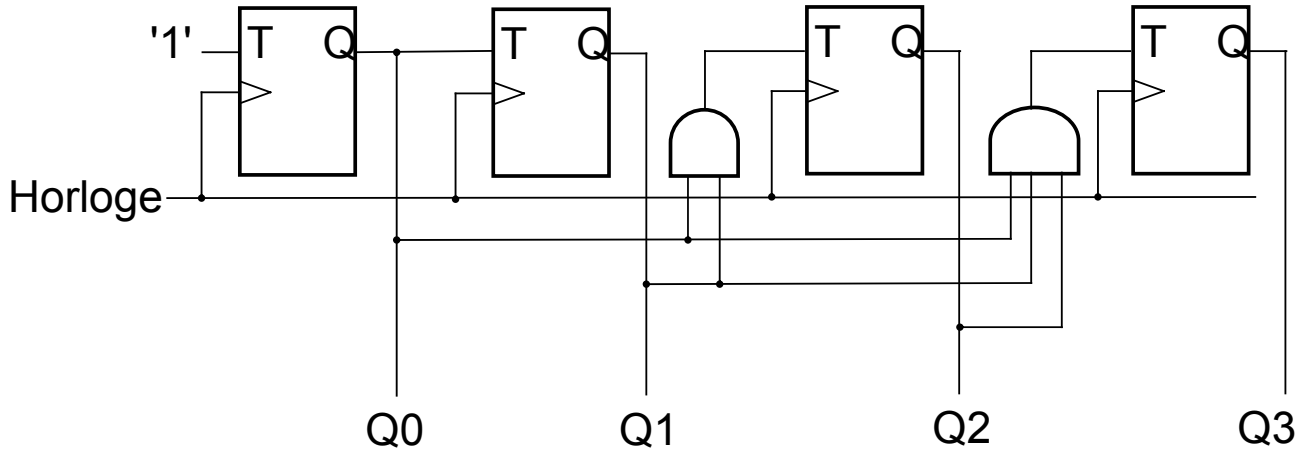


Analyse compteur pseudo synchrone

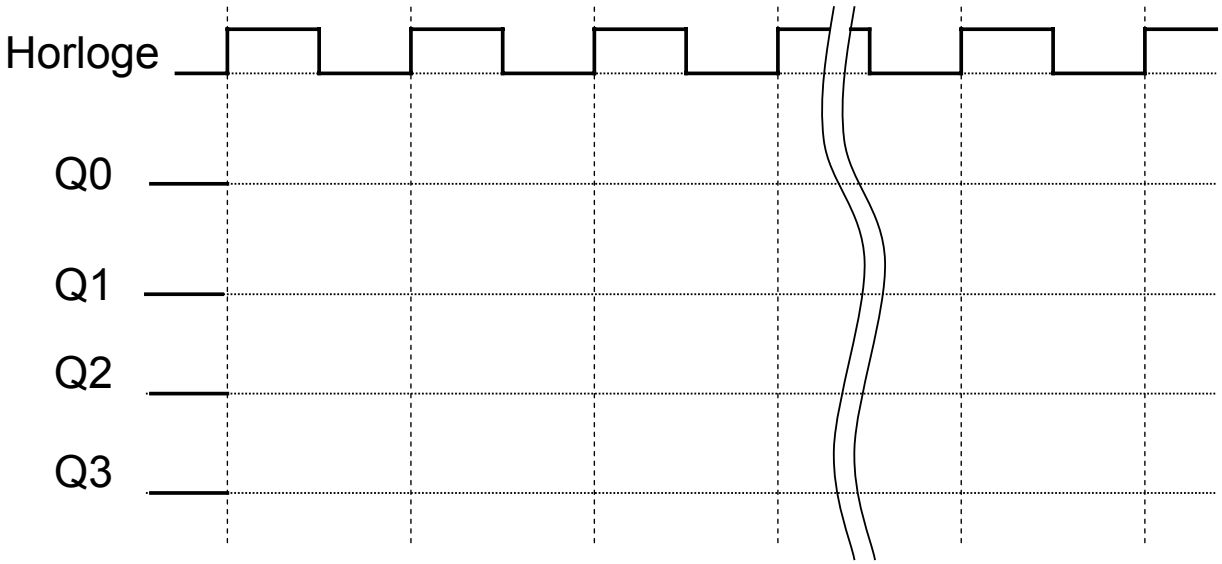
- Construction très simple
- Pas synchrone
  - ✓Retard variable pour chaque sortie
  - ✓Retard cellule N : Tn = N x tp
  - ✓Très difficile à tester (vérification timing)
  - ✓Possible sortie change après flanc suivant de l'horloge
- Très mauvaise intégration dans des PLDs

Dia laissé vide volontairement

Compteur *full* synchrone



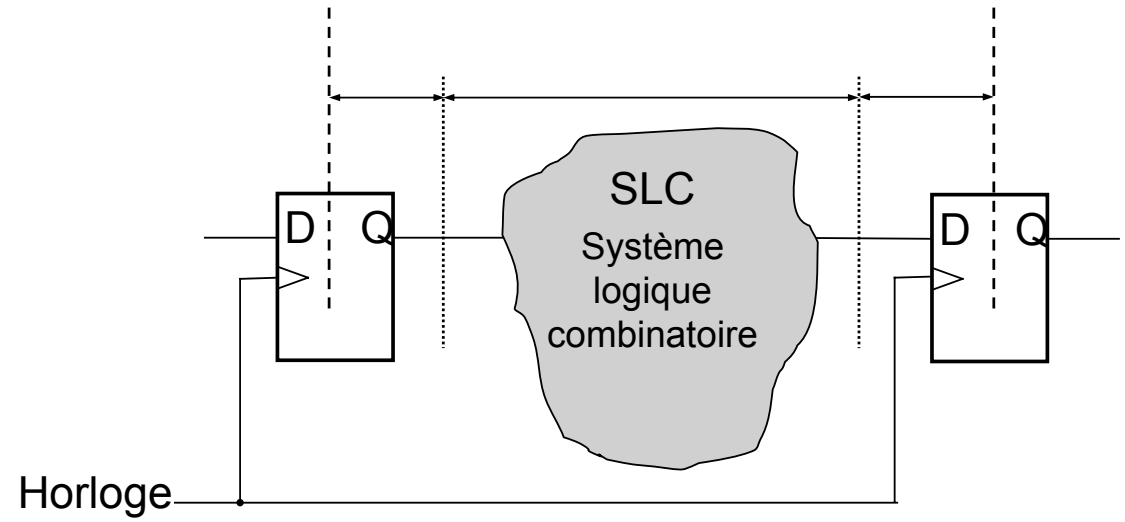
Compteur *full* synchrone



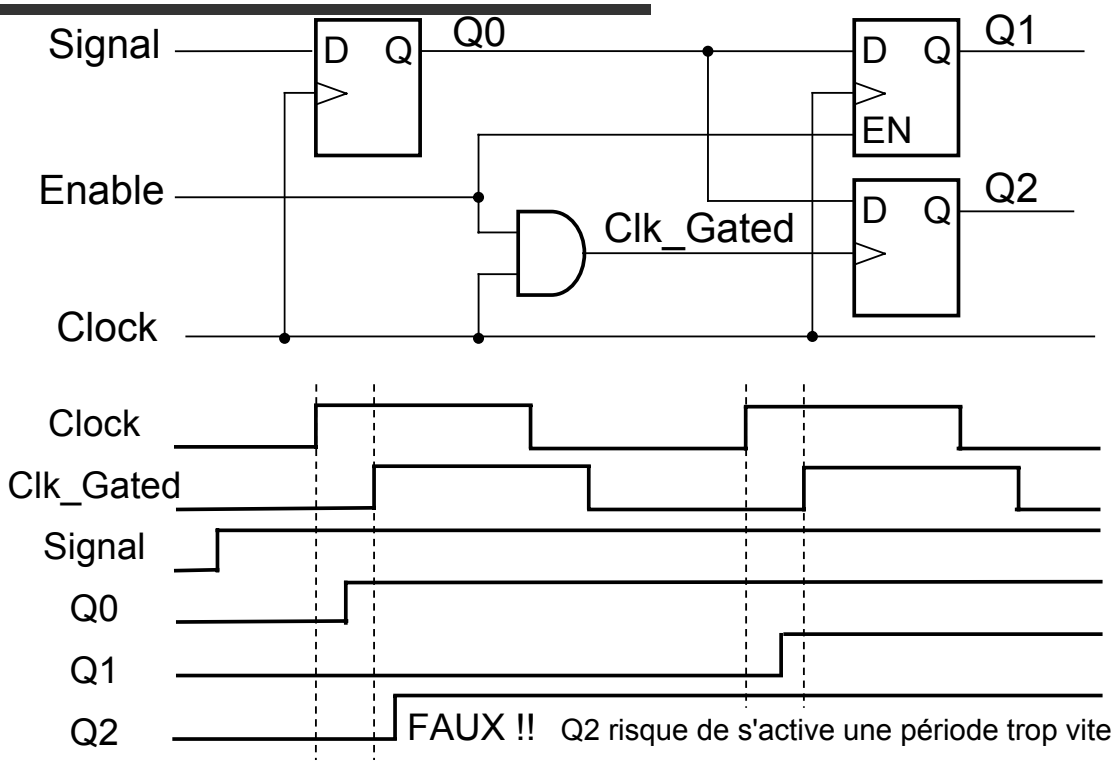
Analyse du compteur synchrone

- 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 des timings du circuit
- 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 changment de Q0 durant le même flanc
  - ✓fonctionnement dépend timings portes & connections !!

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)

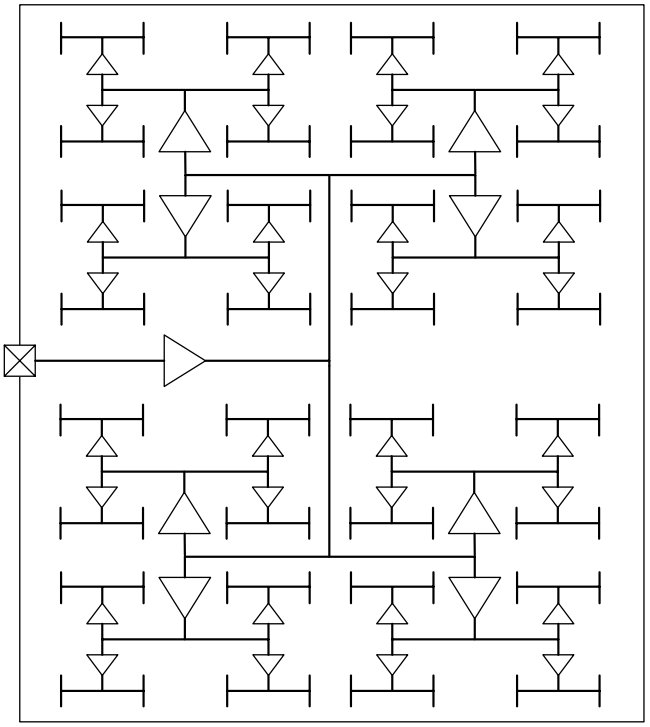
Parameter	-6 Speed Grade *(1)*		-7 Speed Grade *(2)*		-8 Speed Grade *(2)*		Unit
	Min	Max	Min	Max	Min	Max	
TCO	141	250	141	277	135	304	ps
					141		ps

Cyclone II, Altera
TCO: clock-to-out

Structure d'un arbre d'horloge

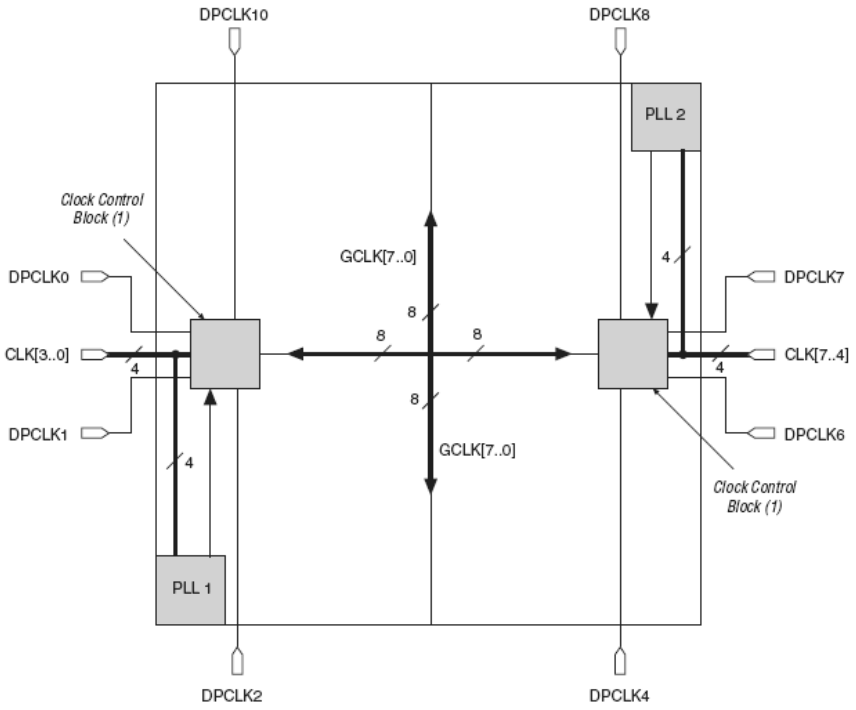
- Arbre équilibré afin de garantir un temps de propagation constant pour tous les flip-flops

Entrée d'horloge



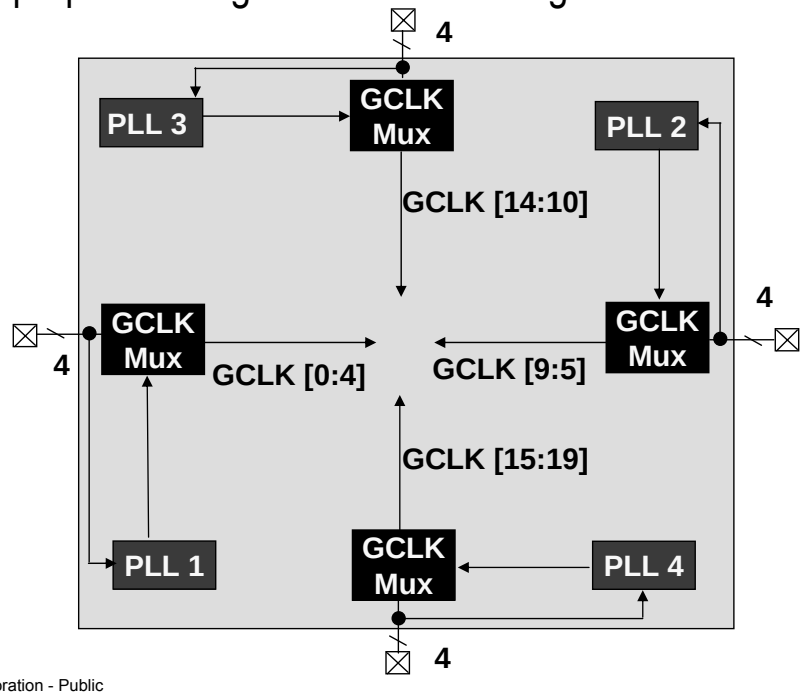
Exemple arbre d'horloge Cyclone II

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



Clock Networks & PLLs Overview

- Up to 20 Global Clocks Per Device
- Dual purpose as high fan out control signals

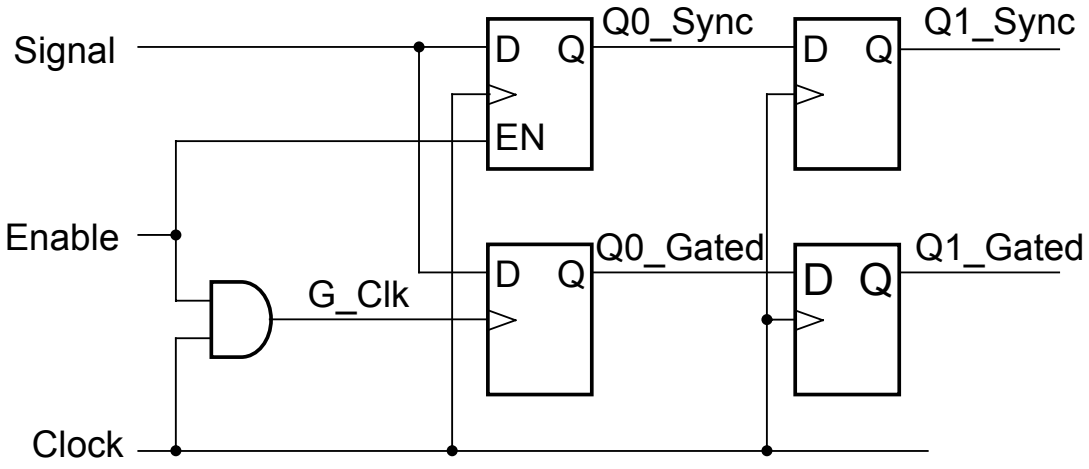


Exercices 1 et 2

- L'objectif de ces deux exercices est de vérifier si l'utilisation d'un "Gated Clock" est fiable dans un PLD
- Marche à suivre :
  - ✓Simuler le circuit avant synthèse (simulation VHDL)
  - ✓Faire la synthèse puis le placement-routage dans un CPLD de type MAX7000S: EPM7128SLC-15
  - ✓Simuler le circuit après PR pour un CPLD
  - ✓Faire la synthèse puis le placement-routage dans un FPGA de type Cyclone: EP1C3T100C8
  - ✓Simuler le circuit après PR pour un CPLD
  - ✓Comparer et commenter les résultats

Exercice 1

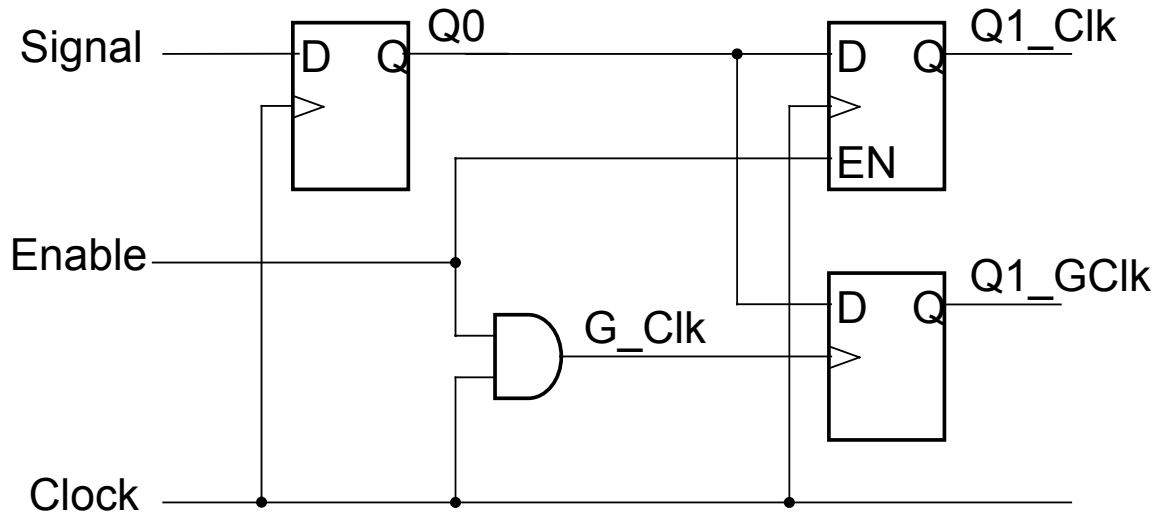
- Mesurer l'écart de propagation entre les sortie Q0 et Q1 du schéma ci-dessous.



Bascules de post-synchronisation indispensable pour mesurer les délais internes à la FPGA

Exercice 2

- Vérifier, après placement-routage, s'il y a une course entre la bascule Q1 et Q0 (si Q1 change durant la même période d'horloge que Q0)!



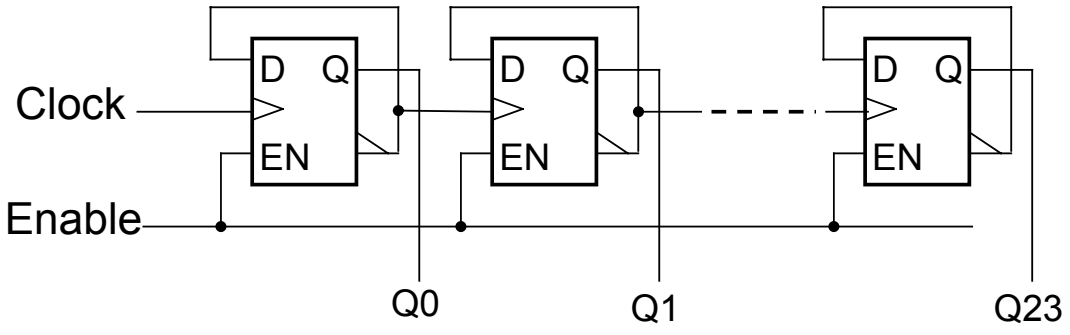
page volontairement laissée vide

Exercice 3 …

- Réaliser un compteur 24 bits de type "ripple counter" (asynchrone). Celui-ci disposera d'un signal *Enable*
  - ✓déterminer la fréquence de fonctionnement après P-R
    - →Famille CPLD: MAX7000S EPM7128SLC84-15
    - →Famille FPGA: Cyclone EP1C3T100C8
- Vous utiliserez l'instruction *generate* pour la description structurelle de votre compteur
- Simuler votre compteur, après 4096 pas de comptage, désactiver le signal *Enable* après le flanc actif de l'horloge après 40 ns pour EPM et 5 ns pour EP1C Quelle constatation pouvez-vous faire ?

… exercice 3

- Schéma du "ripple counter" :

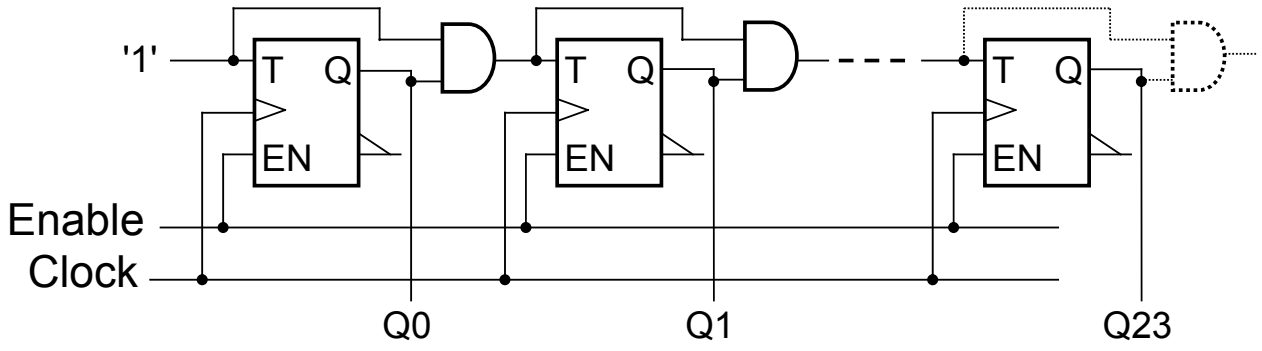


Exercice 4 …

- Réaliser un compteur synchrone de 24 bits avec des flip-flops T. Celui-ci disposera d'un signal *Enable*
  - ✓déterminer la fréquence de fonctionnement après P-R
    - →Famille CPLD: MAX7000S EPM7128SLC84-15
    - →Famille FPGA: Cyclone EP1C3T100C8
- Vous utiliserez l'instruction *generate* pour la description structurelle de votre compteur
- Simuler votre compteur, après 4096 pas de comptage, désactiver le signal *Enable* après le flanc actif de l'horloge après 40 ns pour EPM et 5 ns pour EP1C Quelle constatation pouvez-vous faire ?

… exercice 4

- Schéma du compteur synchrone



Designs complexes sur FPGA

Annexes

Les mots réservés du VHDL93 …

abs
access
after
alias
all
and
architecture
array
assert
attribute

begin
block
body
buffer
bus

case
component
configuration
constant

disconnect
downto

else
elsif
end
entity
exit

file
for
function

generate
generic
group
guarded

… mots réservés du VHDL93 …

if
impure
in
inertial
inout
is

label
library
linkage
loop

map

mod

nand
new
next
nor
not
null

of
on
open
or

others
out

package
port
postponed
procedure
process
protected
pure

range
record

… mots réservés du VHDL93

register
reject
rem
report
return
rol
ror

select
severity
signal
shared
sla

sll
sra
srl
subtype

then
to
transport
type

unaffected
units
until

use
variable

wait
when
while
with

xnor
xor

Historique normes du langage VHDL …

- 1980 : Début du projet financé par le DoD
- 1987 : Standard IEEE Std 1076-1987 (VHDL 87)
- 1993 : Standard IEEE Std 1164-1993 (Std_Logic_1164)
- 1993 : Standard IEEE Std 1076-1993 (VHDL 93)
- 2000 : Standard IEEE Std 1076-2000 (VHDL 2000)
- 2002 : Standard IEEE Std 1076-2002 (VHDL 2002)
- 2008 : Standard IEEE Std 1076-2008 (VHDL 2008)
  - ✓ norme 2008 actuellement pas supportée par les outils EDA!

… historique normes du langage VHDL …

- 1995 : Standard IEEE Std 1076.4
  - ✓ Vital_Primitive et Vital_Timing pour la simulation après P-R
- 1997 : Standard IEEE Std 1076.3
  - ✓ Normalisation des paquetages Numeric_Bit et Numeric_Std
- 1999 : Standard IEEE Std 1076.6
  - ✓ Normalisation de la syntaxe pour la synthèse RTL: Standard for VHDL Register Transfer Level Synthesis
- 2004 : Standard IEEE Std 1076.6 -2004
  - ✓ Evolution de la syntaxe pour la synthèse RTL, pas de changement significatif.

… historique normes du langage VHDL

Autres étapes de l'historique

- 1994 : Approbation ANSI (ANSI/IEEE Std 1076-1993)
- 1996 : Standard IEEE Std 1076.2 (Mathematical Packages)

Autres normalisations

- 1999 : Standard IEEE-1076.1, VHDL-AMS (modélisation mixte)

Références HEIG-VD

- Présentations des collègues
  - ✓Serge Boada
  - ✓Maurice Gaumain
- Manuels VHDL
  - ✓Darryl Gauthey
  - ✓Claude Guex
  - ✓Michel Salamin
  - ✓Yves Sonnay

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

… 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

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

Sites internet outils EDA

- Informations des vendeurs d'outils EDA
  http://www.vhdl.org/
- 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