

SOC et IP

Patrice Nouel



Défit technologique

- Intégrer ensemble
 - 0.10µm CMOS pour le numérique
 - 0.12 µm Bipolaire pour la RF
 - 0.18 µm DMOS pour la puissance
 - DRAM
 - Mémoires FLASH
 - Mems, senseurs, actuateurs électromécaniques

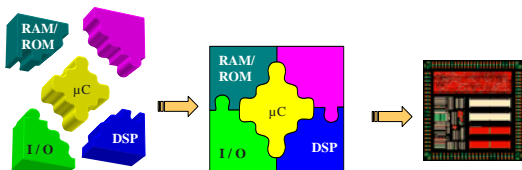


SOC et IP

2

System On Chip

- Comporte au moins un processeur



IP « core »
virtual components



SOC et IP

3

IP: Composants virtuels

- Pour être capable d'intégrer un système, il faut:
 - Réutiliser les blocs déjà conçus (Design Reuse)
 - Acheter ou trouver des blocs génériques chez des fournisseurs extérieurs
- Les IP sont des blocs de propriété intellectuelle encore appelés: IP, IP blocks, cores, system-level blocks (SLB), macros, system level macros (SLMs), or Virtual Components (VCs).

IP: classification

- Soft: Source VHDL ou Verilog doit être synthétisée et optimisée au niveau physique
 - Avantage: Indépendance vis à vis du choix technologique.
 - Inconvénient: Travail d'optimisation peut être long.
- Hard: Description ciblée technologiquement
 - Avantage: performances escomptées (à vérifier)
 - Inconvénient: Impossible de changer de cible.
- Firm: Niveau portes (netliste)
 - Avantages : Flexible et prédictible

Design Reuse

- Méthodologie de conception des circuits
- Règles à respecter pour la conception des IP
- La qualification garantie le respect de ses règles
- VSIA-Virtual Socket Interface Alliance: Groupement de sociétés s'ayant donné pour but d'unifier le monde SOC
 - <http://www.vsi.org>
- Livre conseillé: Reuse Methodology manual For System On Chip Designs. Michael Keating and Pierre Bricaud – Kluwer Academic Publishers



IP: Fournisseurs

- Nombreux fournisseurs de blocs de propriété intellectuelle impossibles à dénombrer.

<http://www.opencores.org>

<http://www.design-reuse.com>

<http://www.mentor.com/inventra/>

<http://www.xilinx.com>

<http://www.altera.com>



IP: dominantes

- Traitement d'image
- Filtres numériques
- Circuits d'interface: USB ...
- Cryptage des données
- Processeurs et périphériques
- Communications sans fil
- DSP
- Mémoires



Codesign

- Concerne un projet mariant une étude de conception matérielle et logicielle conjointement.
- On peut être amené dans ce cadre à effectuer des co-simulations (débogage d'un programme sur une plate-forme matérielle simulée).
- Un projet SOC mêle trois aspects: architecture matérielle , technologie et logiciel.

Outils

- Vérification système : A partir du moment où on peut rapidement construire un gros système avec des IP disparates, la difficulté se reporte sur la vérification système matérielle et logicielle conjointement. C'est dans ce domaine que les outils se développent le plus à l'heure actuelle.
- Partition logiciel-matériel : Définir l'architecture idéale du système.



Les cœurs de processeurs

- Très nombreux (plusieurs centaines) sont inventoriés avec sur le site <http://www.design-reuse.com>
- Les leaders dans ce domaine sont ARM et MIPS. Plus de 50 sociétés possèdent 140 licences ARM (principalement des applications faible consommation). Mips est très répandu dans les circuits intégrés pour les jeux, les imprimantes et les communications



Critères de Choix

- Le processeur
 - RISC ou CISC
 - 8, 16, 32, 64 bits
 - IP SOFT ou HARD , configurable ou pas
- Le système de bus
- Les périphériques existantes
- Le système d'exploitation
- Les outils disponibles de codesign.
 - Implantation du cœur et des IP
 - Vérification et test



Pourquoi utiliser un processeur softcore ?

- Durée de vie du produit
- Indépendance de la technologie
- Prototypage en vue de réaliser un ASIC
- Propriété et maîtrise des sources
- Souplesse d'évolution (VHDL est standard)

Processeurs disponibles

- Cœur PowerPc en dur dans Virtex (xilinx)
- Hardcores (Arm et MIPS par exemple)
- Softcores commerciaux multi-cibles ASIC ou FPG
- Softcores commerciaux à cible propriétaire
 - NIOS pour Altera
 - Microblaze et Picoblaze pour Xilinx
 - Autres
- Softcore opensource
 - Leon Sparc V8
 - OpenRISc (OpenCores.org)

Exemple d'IP gratuit: PIC P1684

- <http://www.opencores.org> Projet PPX16mcu
- Site créé en mai 2002, revu en octobre 2002
- 10 fichiers VHDL
- Apparence du VHDL: Bon choix de bibliothèques, peu ou pas de commentaire dans le code.
- Compilation: une erreur facile à corriger
- Simulation: certaines fonctionnalités fausses. Trois erreurs détectées et corrigées dont une complexe.

Projet PIC - Méthodologie

- Mettre au point le logiciel (fichier *.hex)
- Créer le composant ROM en VHDL contenant le programme. → Rom84.vhd
- Simuler au niveau RTL
- Synthétiser
- Placer-router après avoir choisi la cible.
- Remarque: Modèle de simulation diffère légèrement du modèle réel.

Projet PIC sur FPGA

- Cible: kit de développement Nios Altera
- FPGA: EP1S10F780C6
- Réalisation: Chenillard à diode par interruption timer.
- Teste les prédiviseurs , le timer, l'interruption.

Structure du circuit de test

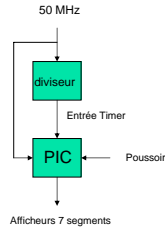
- LIBRARY IEEE;
- USE IEEE.std_logic_1164.ALL;
- USE IEEE.numeric_std.ALL;
- ENTITY pic IS
- GENERIC (
■ simulation : boolean := false); -- change les constantes de temps
- PORT (
■ clk : IN std_logic; -- horloge systeme
- reset_n : IN std_logic; -- reset general
- int : IN std_logic; -- demande interruption
- port_a : INOUT std_logic_vector(7 DOWNTO 0);
- port_b : INOUT std_logic_vector(7 DOWNTO 0));
- END pic;

Circuit de test

```

P1: P1S04 PORT MAP (
    clk => clk,
    reset_n => reset_n,
    T0CLK => T0CLK,
    K0 => K0,
    port_a => port_a,
    port_b => port_b);
diviseur PROCESS
    CONSTANT rapport : natural := 749; -- si 50 Mhz alors 15 us
    CONSTANT rapport_s : natural := 9; -- pour simu
    VARIABLE c : natural RANGE 0 TO rapport;
    BEGIN -- PROCESS diviseur
        WAIT UNTIL rising_edge(clk);
        T0CLK => '0';
        IF simulation THEN -- simulation
            IF c < rapport_s THEN
                c <= c + 1;
            ELSE
                c <= 0;
            END IF;
        ELSE -- synthèse
            IF c < rapport THEN
                c <= c + 1;
            ELSE
                c <= 0;
            END IF;
        END IF;
    END PROCESS diviseur;
    END PROCESS diviseur;

```



Le circuit ALTERA EP1S10F780C6

- EP1S: Famille Stratix technologie cuivre 1,5volt 0,13µm (fréquence limite 300 MHz)
- 10 : Surface et ressources
 - 10570 Logic element
 - 920448 bits de RAM soient
 - 94 blocs de RAM 32x18 bits
 - 128 blocs de RAM 128x36 bits
 - 1 bloc de RAM 4Kx144 bits
 - 6 Blocs DSP
 - 48 multiplieurs 9X9 bits ou 24 multiplieurs 18x18 bits ou 6 multiplieurs 36x36 bits

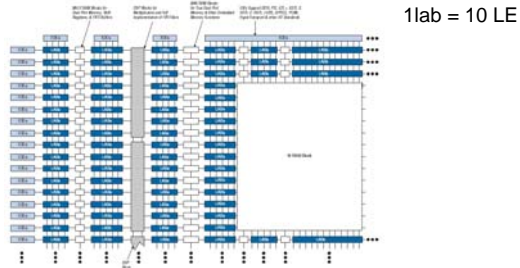


EP1S10F780C6 (suite)

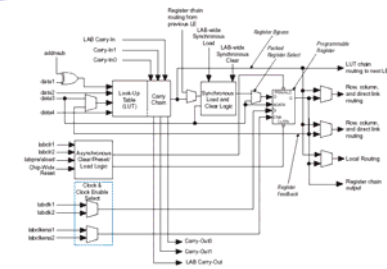
- Ressources (suite)
 - 6 PLLs
- F: Type de package -> Fine line BGA
- 780 : Nombres de broches
 - 426 I/O max : Nombreux standards de communication rapide simple ou différentielle.
- C: Commerciale -> circuit garanti de 0°C à 85°C
- 6 : rapidité moyenne (échelle de 5 rapide à 7 lent)



EP1S10 (suite) Architecture interne



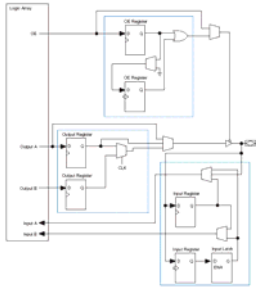
EP1S10 (suite) Logic element



EP1S10 (suite) Les horloges disponibles

- 16 entrées réservées aux horloges. Cela va piloter l'ensemble des horloges globales ou locales.
- 6 PLL pour diviser ou multiplier les fréquences d'horloge
- Le réseau d'horloge est structuré:
 - globale = sur l'ensemble du circuit, on garantie un minimum de skew
 - Locale = idem mais sur un quadrant particulier

EP1S10 (suite) Les I/O



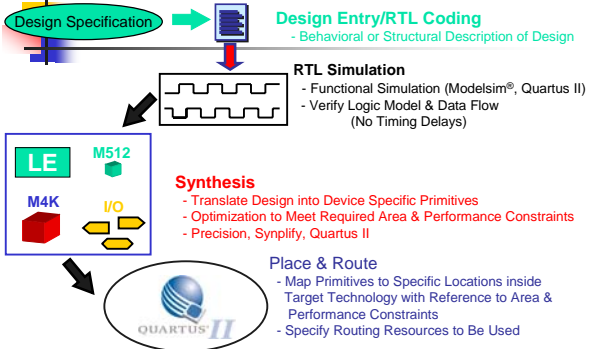
- JTAG
- LVDS
- 3.3v PCI
- Pullup resistor
- Tri-state buffer
- Open-drain outputs
- Skew control
- Programmable delay

ALTERA

SOC et IP

28

PLD Design Flow

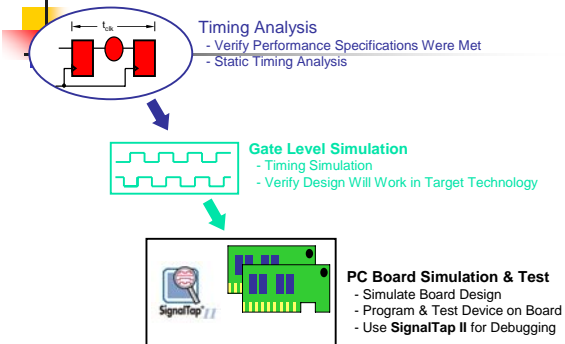


ALTERA

SOC et IP

29

PLD Design Flow



ALTERA

SOC et IP

30

Principes de conception VHDL

- Tout doit être synchrone
- On rentre sur des registres, on sort sur des registres
- Choix des bibliothèques VHDL normalisées par IEEE
 - Std_logic_1164
 - Numeric_std
- Commentaires
- Utilisation de conventions pour les identificateurs: reset_n , clear, clock
- Chaque module doit être accompagné de son test bench

Pourquoi utiliser des scripts ?

- Pour lier des outils entre eux (precision ou modelsim + quartus)
- Pour automatiser des tâches
- Pour diminuer la taille mémoire d'un projet
- Pour faciliter la maintenance d'un projet
- Pour créer des outils (ex désassembleur dans Modelsim)

Le langage Tcl/Tk

- TCL = Tool Command Language Tk est une extension qui introduit le graphique
- Fonctionne sur tous types de machine
- Langage interprété (n'a pas besoin d'être compilé)
- Quartus II et modelsim utilisent TCL comme langage de script
- Editeur dédié dans Quartus

Syntaxe TCL

- Sensible à la casse (majuscules-minuscules)
- Syntaxe courante
 - Command `arg1 arg2 arg3...`
 - Exemple : `set var 3`
- Substitution de variable
 - `puts "La valeur de la variable est $var "`
 - Ecrit "La valeur de la variable est 3" dans la fenêtre message
- Substitution de commande
 - `puts "La valeur de la variable est [expr $var + $var]"`
 - Ecrit "La valeur de la variable est 6" dans la fenêtre message

Syntaxe TCL (suite)

- Arithmétique
 - On passe par la commande `expr`
- Listes
 - `set a { 1 2 3 }` donne 3 valeurs à a
 - `set b [length $a]` b est égal à la longueur de a
 - `set c [lindex $a 2]` c est égal à a[2] ou à 3

Syntaxe TCL (suite)

- Structures de commande
 - `if-elseif-else`
 - Boucles `for`
 - Boucles `foreach`
 - Boucle `loops`
- Procédures
 - Commande `proc`
 - Utiles pour créer des fonctions ou des sous-programmes



Paquetages TCL

- Regroupent un certain nombre de fonctions proches
- Exemple de packages dans Quartus:
 - ::quartus::project pour projets et affectations
 - ::quartus::flow pour compilation et run
 - ::quartus::report pour créer des rapports particuliers
 - ::quartus::timing_report pour les chemins critiques

Load_package <name> [-version <version>]





Exemple de script TCL

```

package require ::quartus::project

set need_to_close_project 0
set make_assignments 1

# Check that the right project is open
if {[is_project_open]} {
  if {[string compare $quartus(project) "pic"]} {
    puts "Project pic is not open"
    set make_assignments 0
  }
} else {
  # Only open if not already open
  if {[project_exists pic]} {
    project_open -revision pic pic
  } else {
    project_new -revision pic pic
  }
  set need_to_close_project 1
}

```





Exemple de script TCL

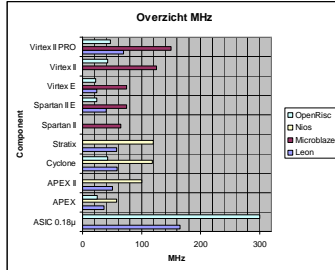
```

# Make assignments
if ($make_assignments) {
  set_global_assignment -name ORIGINAL_QUARTUS_VERSION "4.1 SP2"
  set_global_assignment -name PROJECT_CREATION_TIME_DATE "10:35:48 JANUARY 25, 2005"
  set_global_assignment -name LAST_QUARTUS_VERSION 4.2
  set_global_assignment -name VHDL_FILE rtl/new_ppx_pcs.vhd
  # etc..
  set_global_assignment -name VHDL_FILE rtl/ppx_ram.vhd
  set_global_assignment -name VHDL_FILE rtl/ppx_tmr.vhd
  set_global_assignment -name VHDL_FILE rtl/rom84.vhd
  set_global_assignment -name IGNORE_CLOCK_SETTINGS ON
  set_global_assignment -name FMAX_REQUIREMENT "70.0 MHz"
  set_global_assignment -name DEVICE_FILTER_PACKAGE FBGA
  set_global_assignment -name DEVICE_FILTER_PIN_COUNT 780
  set_global_assignment -name DEVICE_FILTER_SPEED_GRADE 6
  set_global_assignment -name FAMILY Stratix
  set_global_assignment -name DEVICE EP1S10F780C6
  set_location_assignment PIN_K17 -to clk
}

```



Performances comparées (Patrick Pelgrims -De Nayer Instituut)



SOC et IP

40

Performances comparées (Patrick Pelgrims -De Nayer Instituut)



Device Family	Timing	D-MIPS	MUL	RAM	Cache (MB)
Virtex II 1000-4	100 MHz	62	HW	blockRAM	NA
Virtex II 1000-4	100 MHz	65	SW	blockRAM	NA
Virtex II 1000-4	150 MHz	4	HW	SDRAM	NA
Virtex II 1000-4	50 MHz	6	HW	SDRAM	2
Virtex II 1000-4	50 MHz	13	HW	SDRAM	8



Device Family	Timing	D-MIPS	RAM	Cache (MB)
Altera APEX20K200-2K	33 MHz	13	SRAM	0
Altera APEX20K1000E-2K	40 MHz	9	SDRAM	0
Altera Cyclone EP1C20	50 MHz	17	SRAM	4
Altera Cyclone EP1C20	50 MHz	15	SDRAM	4
Altera Stratix EP1S10	50 MHz	17	SRAM	4
Altera Stratix EP1S10	50 MHz	15	SDRAM	4



Device Family	Timing	D-MIPS	RAM
Virtex-E 1500	25 MHz	20	SRAM



Device Family	Timing	D-MIPS	RAM	Cache (MB)
Xilinx Virtex-E 1000E-4	25 MHz	21	SRAM	1
Xilinx Virtex-E 1000E-4	25 MHz	21	SRAM	1
Altera APEX20K200-2K	26 MHz	21	SRAM	2
Altera APEX20K1000E-2K	20 MHz	4	SDRAM	8
Virtex Virtex-II 4000S-5	40 MHz	27	SRAM	4
Altera Cyclone EP1C20	50 MHz	33	SRAM	4



SOC et IP

41
