# Découverte du système NIOS II Altera

Note: Les illustrations correspondent à la version logicielle Quartus 8.1

## 1) Objectif pédagogique

Cette première séance, incontournable, offre la possibilité de découvrir les outils de développement du matériel et du logiciel. On construira un premier système simple, on en fera la synthèse complète, on y associera une application logicielle. L'ensemble sera chargé et testé sur la carte de développement. On aura de ce fait une première approche du **SOPC Builder** pour la génération du système, de **Quartus** pour la synthèse et le placement/routage, de l'IDE pour le développement logiciel, compilation et debug.



## 2) Élaboration du système

- → Recopier dans votre espace de travail le répertoire Tp\_sopc et ouvrir avec Quartus le projet uPC1S10.qpf qu'il contient.
- → Ouvrir la feuille de schéma de niveau supérieur: file>open> uPC1S10.bdf.Elle contient les entrées-sorties non encore affectées du système que l'on va construire.

### 2.1) Démarrage du SOPC Builder

→ La commande Tools>SOPC Builder propose de créer un système que l'on appellera *nios2\_system* et on fera le choix de VHDL comme langage de génération.

| 12        | Create New System       | 0 |
|-----------|-------------------------|---|
| System N  | lame: Nios2_system      |   |
| Target HD | DL: O Verilog<br>O VHDU |   |
|           | OK Cancel               |   |

➔ Après ce choix (OK), l'interface graphique du SOPC Builder apparaît. On vérifiera que la fréquence du système est bien de 50 Mhz.

| Altera SOPC   | : Builder - unnamed.sopc (D:'patrice'tr<br>ols Help | avail\TpNios81\ressources\ | uPC1s10\unnamed.soj | oc)     | 000   |
|---|---|----------------------------|---------------------|---------|-------|
| System Contents System Generation   |   |                            |                     |         |       |
| Attera SOPC Builder   | Target<br>Device Family: Stratix                    | Clock Settings             | Source              | MHz     | Add   |
| Interface Protocols     Interface Protocols     Interface Protocols     Memories and Memory Control     Peripherals     PLL | Use Module Name                                     | Description                | Clock               | Base    | End   |
| G Video and Image Processing  |   |                            |                     |         |       |
| New Edt Add   | Remove Edit   | Move Up                    | ve Down             | ess Map | ilter |
| Info: No errors or warnings.  |   |                            |                     |         |       |
|   | Exit Help   | Prev Next                  | Generate            |         |       |

On va pouvoir maintenant construire le système voulu. Il comportera les composants suivants:

- 1. Le CPU NIOS
- 2. Un timer nécessaire pour les routines systèmes basées sur le temps;
- 3. Le contrôleur de mémoire externe flash
- 4. Le contrôleur de mémoire externe RAM
- 5. Le contrôleur de mémoire externe SDRAM
- 6. L'interface JTAG UART
- 7. Le BUS externe permettant de connecter ces mémoires(Avalon tri-state bridge)
- 8. L'interface pour l'afficheur LCD
- 9. Le PIO interface parallèle pour les LEDs
- 10. Le PIO interface parallèle pour les boutons poussoirs
- 11. Le PIO interface parallèle pour les afficheurs 7 segments
- 12. Le composant générateur d'identification du matériel
- 13. La pll génératrice d' horloge pour la mémoire SDRAM

Tous le composants seront extraits du menu du SOPC Builder: System contents

Ne pas s'inquiéter des différents messages (**warning** ou error) apparaissant pendant la construction du système. Ils sont temporaires et dûs à des éléments manquants. Cependant, avant de générer le système, ces messages devront avoir disparus en fin de construction.

#### 2.2) Le CPU NIOS

→ Effectuer un double-click sur la ligne Nios II Processor . On va implanter une version moyenne du processeur.

Comme le montre l'image ci-après, la première fenêtre « **Core NIOS II** » permet de faire ce choix mais ne pourra pas être entièrement configurée en ce qui concerne les vecteurs d'exceptions tant qu'on n'a pas de mémoire disponible, cela sera fait <u>un peu plus tard</u>.

|   |                               | Nios II Proces   | ssor - cpu   |                                   |         |
|---|-------------------------------|--|--|-----------------------------------|---------|
| Nio   | s II Processor                |  |  | About                             | tation  |
| Parameter<br>Settings<br>Core Nios II Ca  | ches and Memory Interfaces    | Advanced Features  | MMU and MPU Settings   | JTAG Debug Module 🔰 Custom Instru | uctions |
| Core Nios II  |                               |  |  |                                   |         |
|   | ○Nios II/e                    | Nios II/s  | ○Nios II/f   |                                   |         |
| Nios II<br>Selector Guide<br>Family: Stratix<br>1 <sub>system:</sub> 50,0 MHz<br>cpuid: 0 | RISC<br>32-bit                | RISC<br>32-bit<br>Instruction Cache<br>Branch Prediction<br>Hardware Multiply<br>Hardware Divide | RISC<br>32-bit<br>Instruction Cache<br>Branch Prediction<br>Hardware Multiply<br>Hardware Divide<br>Barrel Shifter<br>Data Cache<br>Dynamic Branch Predi | ction                             |         |
| Performance at 50,0 M   | IHz Up to 8 DMIPS             | Up to 32 DMIPS   | Up to 57 DMIPS   |                                   |         |
| Logic Usage   | 600-700 LEs                   | 1200-1400 LEs  | 1400-1800 LEs  |                                   |         |
| Memory Usage  | Two M4Ks (or equiv.)          | Two M4Ks + cache   | Three M4Ks + cache   |                                   |         |
| Reset Vector: M   | SP Block<br>lemory: ext_flash | Hardware Divide  | ) 0x   | :00000000                         |         |
| Exception Vector: Me  | emory: ext_ram                | Offset: 0x20   | ) 0x0  | 00900020                          |         |
| Only include the MMU Fast TLB Miss Exception  | when using an operating syste | m that explicitly supports an M  | NU<br>Offset:  |                                   |         |
|   |                               |  |  | Cancel < Back Next.>              | Finish  |

#### → Faire Next

|   |                 | -  |
|---|-----------------|--|
| Nios II Processor                             |                 | About  |
| arameter<br>ettings                           |                 |  |
| Core Nios II Caches and Memory Interfaces Ad- | vanced Features | ightarrow MMU and MPU Settings $ ightarrow$ JTAG Debug Module $ ightarrow$ Custom Instructions |
| -Instruction Master                           |                 | n Data Master  |
| Instruction Cache: 4 Kbytes                   |                 | Data Cache: 2 Kbytes Omt date mester port  |
| Enable Bursts (Burst Size: 32 bytes)          | Help            | Data Cache Line Size: 32 Bytes   |
|   |                 | Enable Bursta (Burst Size: 32 bytes) Help  |
| Number of ports: 1                            |                 | Include tightly coupled data master port(s).   |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 |  |
|   |                 | Cancel.   < Back.   [Next > ] [-Fini   |

→ Dans la fenêtre « caches and memory interfaces », on fixera 4K Octets de mémoire cache puis Next 3 fois

|   |  | Nios II Process  | sor - cpu  |  |                     |
|---|--|--|--|--|---------------------|
| Mios  | s II Processor   |  |  | Abo  | ut Documentation    |
| Core Nios II Cac  | hes and Memory Interfaces >  | Advanced Features  | MMU and MPU Settings $>$   | JTAG Debug Module  | Custom Instructions |
| TAG Debug Module-   | avel   |  |  |  |                     |
| No Debugger   | Level 1  | O Level 2  | O Level 3  | O Level 4  | l.                  |
|   | JTAG Target Connection<br>Download Software<br>Software Breakpoints                      | JTAG Target Connection<br>Download Software<br>Software Breakpoints<br>2 Hardware Breakpoints<br>2 Data Triggers | JTAG Target Connection<br>Download Software<br>Software Breakpoints<br>2 Hardware Breakpoints<br>2 Data Triggers<br>Instruction Trace<br>On-Chip Trace | JTAG Target Connection<br>Download Software<br>Software Breakpoints<br>4 Hardware Breakpoints<br>4 Data Triggers<br>Instruction Trace<br>Data Trace<br>On-Chip Trace<br>Off-Chip Trace | •                   |
| No LEs  | 300-400 LEs  | 800-900 LEs  | 2400-2700 LEs  | 3100-3700 LEs  |                     |
| -Break Vector<br>Memory: cpu  | off  | set: 0x20  | 0x00q008   |  |                     |
| Advanced Debug S<br>OCI Onchip Trace: 11<br>OCI Onchip Trace: 12<br>Advanced debug lice | ettings<br>28 Frames<br>needs of end 20 clock alonel H<br>nses can be purchased from FS2 | ein<br>. www.fs2.com   |  |  |                     |
|   |  |  |  |  |                     |

- → Level 1 sera choisi pour le « JTAG Debug module »
- → On peut valider le tout par Finish
- → Dans le champ Modul name, renommer cpu\_o ( sélection puis clic droit > rename) en cpu

#### 2.3) Le Timer

- → Ajouter un composant Peripherals>Microcontroller peripherals>interval Timer
- → Garder les valeurs par défaut, puis Finish
- → Renommer le composant *timer\_0* en <u>system\_timer</u>

| 3                     | Interval Timer                  | - system_t | timer 🧲       |
|-----------------------|---------------------------------|------------|---------------|
| MegaCore'             | Interval Time                   | r          | About         |
| Parameter<br>Settings |                                 |            |               |
|                       |                                 |            |               |
| [Timeout p            | eriod                           |            |               |
| Period: 1             |                                 | ms         | -             |
| -<br>Timer cou        | nter size                       |            |               |
| Counter Siz           | e: 32 🚽 bits                    |            |               |
|                       |                                 |            |               |
| Presets:              | ustom                           |            |               |
| Registe               | rs                              |            |               |
| Vvri                  | able period                     |            |               |
| 🔽 Rea                 | dable snapshot                  |            |               |
| Star                  | t/Stop control bits             |            |               |
| Coutput s             | ignals                          |            |               |
| Time                  | eout pulse (1 clock wide)       |            |               |
| Sys                   | tem reset on timeout (Watchdog) | )          |               |
|                       |                                 |            |               |
|                       |                                 |            |               |
|                       |                                 |            |               |
|                       |                                 |            |               |
| -                     |                                 |            | Capcel Finish |
|                       |                                 |            |               |

## 2.4) Le contrôleur de mémoire externe Flash

- → Ajouter le composant Memories and memory controller> Flash Memory(CFI)
- → Fixer le type de mémoire à: AMD29LV065120R puis garder les valeurs par défaut et Finish
- → Renommer le composant *cfi\_flash\_0* en *ext\_flash*

| a   |   | Flash Memory  | (CFI) - ext_fla:                                     | sh \varTheta   |
|---|---|---|--|--|
| MegaCore  | Flash   | Memory ((   | CFI)   | About Documentation  |
| Parameter<br>Settings   |   |   |  |  |
| Attributes  | Timing  | >   |  |  |
| Presets: AM   | D29LV065D120  | R   |  | <b>_</b>   |
| Address W<br>Data Width<br>Create an inte<br>flash memory<br>and timing inf | Vidth (bits): 23<br>(bits): 8<br>erface to any in<br>device. Select<br>ormation for a C | Justry-standard CFI (C<br>from a list of tested fla<br>FI memory device which | ommon Flash Ini<br>sh memories or<br>sh does not app | terface)-compliant<br>provide interface<br>lear on the list. |
| Info: Flas  | sh memory capa  | city: 8,0 MBytes (8388  | 608 bytes).  |  |
|   |   |   |  | Cancel < Back Next > Finish                                  |

### 2.5) Le contrôleur de mémoire externe SRAM

- → Ajouter le composant Memories and memory controller>SRAM >IDT71V416
- → Garder toutes les valeurs par défaut et Finish
- → Renommer le composant *sram\_0* en *ext\_ram*

|                     | IDT71V416 SRA                        | M              |
|---------------------|--------------------------------------|----------------|
| arameter<br>attings |                                      | [:]            |
|                     |                                      |                |
| Static RAM          |                                      | D              |
| 101101 9 012        | KBytes                               |                |
| Generic m           | emory model (simulation only)—       |                |
| V Include           | a functional memory model in the sys | stem testbench |
|                     |                                      |                |
|                     |                                      |                |
|                     |                                      |                |
|                     |                                      |                |

#### 2.6) Le contrôleur de mémoire externe SDRAM

- → Ajouter le composant Memories and memory controller>SDRAM>Sdram Controller
- → Choisir le type (presets) : single Micron MT48LC4M32B2-7 chip
- → Garder toutes les valeurs par défaut et Finish
- $\rightarrow$  Renommer le composant *sdram* 0 en sd*ram*

| SDRAM Controller - sdram                             |
|--|
| SDRAM Controller                                     |
| Parameter<br>Settings                                |
| Memory Profile Timing                                |
| Presets: single Micron MT48LC4M32B2-7 chip           |
| Data width Bits: 32                                  |
| Architecture<br>Chip select: 1 Banks: 4              |
| Address widths Row: 12 Column: 8                     |
| Share pins via tristate bridge                       |
| Tristate bridge selection:                           |
| Generic memory model (simulation only)               |
| Memory size = 16 MBytes<br>4194304 × 32<br>128 MBits |
|  |
| Cancel < Back Next > Finish                          |

#### 2.7) L'interface JTAG UART

- → Ajouter le composant : Interface Protocols> Serial> JtagUart
- → Garder les valeurs par défaut et Finish

http://vhdl33.free.fr

 $\rightarrow$  Renommer le composant *jtag uart 0* en *jtag uart* 

| 12                    |                             | JTAG UART - jta   | g_uart        | 0                   |
|-----------------------|-----------------------------|-------------------|---------------|---------------------|
| MegaCare              | JTAG UAR                    | т                 |               | About Documentation |
| Parameter<br>Settings |                             |                   |               |                     |
| Configuratio          | on Simulation               | IT I O            |               |                     |
| Buffer de             | pth (bytes): 64             |                   | IRQ threshold | t 8                 |
| Const                 | ruct using registers instea | ad of memory bloc | ks            |                     |
| Read FIFC             | ) (Data from JTAG to A      | valon)            |               |                     |
| Buffer de             | pth (bytes): 64             | -                 | IRQ threshold | t 8                 |
| Const                 | ruct using registers instea | ad of memory bloc | ks            |                     |
|                       |                             |                   |               |                     |
|                       |                             |                   |               |                     |
|                       |                             |                   |               |                     |
|                       |                             |                   |               |                     |
|                       |                             |                   |               |                     |
|                       |                             |                   |               |                     |
|                       |                             |                   | Cancel        | <pre></pre>         |

#### 2.8) L' interface de Bus externe

- → Ajouter le composant : Briges and Adapters>Memory Mapped>Avalon-MM tristate bridge
- → Garder les valeurs par défaut et Finish
- → Renommer le composant *tri\_state\_bridge\_0* en *ext\_ram\_bus*
- → Il faut alors connecter au bus les deux composants ext\_ram et ext\_flash. Pour réaliser cela, dans la partie « connection » du SOPC Builder, on déplace la souris sur le fil ext\_ram\_bus.tristate\_master. Une connexion étant repérée par un point noir, on y connecte les deux esclaves (s1) ext\_ram et ext\_flash en cliquant les points d'intersection.

|  |  | Avalor   | n-MM Tristat   | e Bridge - ext_ram_bus   |   |
|--|--|--|--|--|---|
| MegaCore                                       | Avalo  | on-MM  | Trista   | te Bridge  | Documentation   |
| Parameter<br>Settings                          |  |  |  |  |   |
| Incoming Si                                    | gnals St   | hared Signals  | $\geq$   |  |   |
| Register                                       | ed   |  |  |  |   |
| ncreases off                                   | f-chip fmax, bu                                      | t also increas   | ses latency.   |  |   |
| O Not regis                                    | stered   |  |  |  |   |
| Reduces late<br>Note: Check t<br>to be sure yo | ncy, but also n<br>he input setup<br>ur bus inputs r | educes fmax<br>times analys<br>neet system-  | is in the Quart<br>level timing rec                    | us compilation report<br>quirements.   |   |
| Outgoing add                                   | ress and contr                                       | rol signals are  | e always regis   | stered.  |   |
|  |  |  |  |  |   |
|  |  |  |  |  |   |
|  |  |  |  | Cancel < Ba  | ick Next > Finish   |
|  | ◊-♦>   | . s1   |  | Cancel < Ba  | ck Next > Finish  |
|  |  | s1<br>E ext_flas   |  | Cancel < Ba<br>Avaion Memory Mapped Slave<br>Flash Memory (CFI)  | ck Next > Finish  |
|  |  | S1<br>⊟ext_flaes<br>s1<br>⊟ext_ram   | h  | Cancel < 8a<br>Avaion Memory Mapped Slave<br>Flash Memory (CFI)<br>Avaion Memory Mapped Tristate Slave<br>(DT71V418 SRAM   | ck Next > Finish  |
|  |  | s1<br>⊟ ext_flass<br>s1<br>⊟ ext_ram<br>s1   | h  | Cancel < 8a<br>Avaion Memory Mapped Slave<br>Flash Memory (CFI)<br>Avaion Memory Mapped Tristate Slave<br>DT71V418 SRAM<br>Avaion Memory Mapped Tristate Slave   | ck Next > Finish  |
| I<br>I<br>I<br>I<br>I<br>I<br>I<br>I<br>I      |  | s1<br>ext_flass<br>s1<br>ext_ram<br>s1<br>jtag_uar<br>sraion,  | h<br>h<br>t<br>Jag_slave                               | Cancelin < Ba<br>Avaion Memory Mapped Slave<br>Flash Memory (CPI)<br>Avaion Memory Mapped Tristate Slave<br>DT71V416 SR/M<br>Avaion Memory Mapped Tristate Slave<br>JTAG UART<br>Avaion Memory Mapped Slave  | ck Next > Finish<br>clk_0<br>clk_0<br>clk_0<br>clk_0          |
|  |  | s1<br>ext_flas<br>s1<br>ext_ram<br>s1<br>jtag_uar<br>avaion<br>ext_ram                                 | h<br>t<br>tag_starre<br>_bus                           | Cancelin < Ba<br>Avalon Memory Magged Slave<br>Flash Memory (CFI)<br>Avalon Memory Magged Tristate Slave<br>DTTIV418 SRAM<br>Avalon Memory Magged Tristate Slave<br>JTAG UART<br>Avalon Memory Magged Slave<br>Avalon Memory Magged Slave<br>Avalon -MM Tristate Bridge                              | clk_0<br>clk_0<br>clk_0<br>clk_0<br>clk_0                     |
|  |  | s1<br>ext_flas<br>s1<br>ext_ram<br>s1<br>jtag_uar<br>avsion<br>ext_ram<br>ext_ram<br>ext_ram<br>svalon | h<br>t<br>jag_stave<br>_bus<br>_stave<br>_ristate_mast | Cancel < 84<br>Avaion Memory Mapped Slave<br>Flash Memory (CFI)<br>Avaion Memory Mapped Tristate Slave<br>DTT1V418 SRAM<br>Avaion Memory Mapped Tristate Slave<br>JTAG UART<br>Avaion Memory Mapped Slave<br>Avaion Memory Mapped Slave<br>Et Vivaion Memory Mapped Slave                            | clk_0<br>clk_0<br>clk_0<br>clk_0<br>clk_0<br>clk_0            |
|  |  | s1<br>= ext_flas<br>s1<br>= ext_ram<br>sraton,<br>= ext_ram<br>avelon,<br>ram bus.t<br>= ted_diag      | h<br>t<br>Jag_slave<br>j_bus<br>slave<br>ristatc_mast  | Cancel < 84<br>Avaion Memory Mapped Slave<br>Flash Memory (CFI)<br>Avaion Memory Mapped Tristate Slave<br>DT71V418 SRAM<br>Avaion Memory Mapped Slave<br>JTAG UART<br>Avaion Memory Mapped Slave<br>Avaion Memory Mapped Slave<br>Eter Ivaion Memory Mapped Slave<br>Eter Ivaion Memory Mapped Slave | ck Next > Finish<br>clk_0<br>clk_0<br>clk_0<br>clk_0<br>clk_0 |

## 2.9) L' interface pour l'afficheur LCD

- → Ajouter le composant : Peripherals>Display>character LCD
- → Garder les valeurs par défaut et Finish
- $\rightarrow$  Renommer le composant lcd 0 en *lcd display*

|  | Character LCD - Icd  | L_0                        |
|--|--|----------------------------|
| MegaCore   | Character LCD  | About Documentation        |
| Parameter<br>Settings  |  |                            |
| or a Nios II pr<br>5x2-charact<br>evice driver:<br>here are no | ocessor to display characters on an Op<br>er LCD panel.<br>s are provided in the HAL system library<br>user-configurable settings. | rrex 16207 (or equivalent) |
|  |  |                            |
|  |  | Cancel                     |

## 2.10) Le PIO interface parallèle pour les LEDs

- → Ajouter un composant Peripherals>Microcontroller peripherals>PIO (Parallel I/O)
- → Garder les valeurs par défaut (8bits en sorties) puis Finish
- → Renommer le composant  $pio_0$  en  $led_pio$

| PIO (Parallel I/O) - led_pio 😝                |
|---|
| PIO (Parallel I/O) About Documentation        |
| Parameter<br>Settings                         |
| Basic Settings > Input Options > Simulation > |
| Width<br>Width (1-32 bits) : 8                |
| Direction                                     |
| O Bidirectional (tristate) ports              |
| O Input ports only                            |
| O Both input and output ports                 |
| <ul> <li>Output ports only</li> </ul>         |
| Output Port Reset Value<br>Reset Value: 0x0   |
|   |
|   |
| Cancel < Back Next > Finish                   |

#### 2.11) Le PIO interface parallèle pour les boutons poussoirs

- → Ajouter un composant Peripherals>Microcontroller peripherals>PIO (Parallel I/O)
- → Dans Basic Settings, choisir 4 bits et Inputs Ports Only
- → Dans Inputs Options , faire les choix: Synchronously capture / Rising edge et de Generate IRQ / Edge, puis Finish
- → Renommer le composant *pio\_0* en *button\_pio*

| PIO (Parallel I/O) - button_pio 😝   |
|---|
| PIO (Parallel I/O) About Documentation  |
| Parameter<br>Settings   |
| Basic Settings / Input Options / Simulation /   |
| Vidth<br>Vidth (1-32 bits): 4   |
| Direction   |
| O Bidirectional (tristate) ports  |
| <ul> <li>Input ports only</li> </ul>  |
| O Both input and output ports   |
| O Output ports only   |
| Output Port Reset Value Reset Value:  |
|   |
|   |
| 🕂 Warning: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO input |
| ٠   |
| Cancel < Back Next > Finish   |

## 2.12) Le PIO interface parallèle pour les afficheurs 7 segments

- → Ajouter un composant Peripherals>Microcontroller peripherals>PIO (Parallel I/O)
- → Dans Basic Settings, choisir 16 bits et Outputs Ports Only
- → Finish
- → Renommer le composant *pio\_0* en *seven\_seg\_pio*

| PIO (Parallel I/O) - seven_seg_pio   | θ                       |
|--|-------------------------|
| PIO (Parallel I/O)   | About Documentation     |
| Parameter<br>Settings<br>Basic Settings  |                         |
| Width<br>Width (1-32 bits): 16   |                         |
| Direction  |                         |
| <ul> <li>Input ports only</li> <li>Both input and output ports</li> <li>Output ports only</li> </ul> |                         |
| Output Port Reset Value<br>Reset Value: 0x0  |                         |
| <u>.</u>   |                         |
|  |                         |
|  |                         |
| Canc   | el < Back Next > Finish |

## 2.13) Le composant générateur d'identification du matériel

- → Ajouter un composant Peripherals>Debug and performance>System ID Peripheral
- → Finish
- → Renommer le composant sysid\_0 en *sysid*



## 2.14) La PLL génératrice d'horloge pour la SDRAM

- → Ajouter un composant PLL>pll
- → Dans la fenêtre ouverte (PLL settings) cliquer sur Launch Altera's ALTPLL MegaWizard
- → Se déplacer dans l'onglet Output clocks > clock c0. On fera une copie de l'horloge système (50 Mhz) en cochant la case « use this clock », en fixant à 1 les paramètres « Clock multiplication factor » et « Clock division factor »
- → Se déplacer dans l'onglet Output clocks > extclk e0. On crée une horloge de 50 Mhz pour alimenter la SDRAM en cochant la case « use this clock », en fixant à 1 les paramètres « Clock Multiplication factor » et « Clock division factor ». La phase sera fixée à -3,50 ns.
- ➔ Finish
- → La pll s'appelle *pll*

|  | MegaWizard Plug-In Manager [page 5 of 16]                               | 000   |
|--|---|---|
|  |   |   |
| 2  |   | About Documentation                                 |
| 1 Parameter 2 Output 3 EDA 4 Sur         | nmary   |   |
| settings Liocks                          | clk r5extrik e1extrik e1extrik e2                                       | extrike3  |
|  | cB - Core Output Clock  |   |
| altplipli                                | Able to implement in Enhanced PLL                                       |   |
|  |   |   |
| inclk0 frequency: 60.000 MHz e0          | Use this clock  |   |
| Clk Ratio Ph (dg) DC (%)                 | Clock rap Seconds   | Requested settings Actual settings                  |
| o0 1/1 0.00 50.00<br>e0 1/1 -63.00 50.00 | <ul> <li>Enter output clock frequency:</li> </ul>                       | 100.0000000 MHz 🖌 50.000000                         |
|  | Inter output clock parameters:  |   |
| Stratix                                  | Clock multiplication factor   |   |
|  | Clock givision actor  |   |
|  | Clock p <u>h</u> ase shift  | 0.00 💭 ps 🔽 0.00                                    |
|  |   |   |
|  | Clock duty cycle (%)  | 50.00   |
|  | More Details >>   |   |
|  | Elece becaus x x  |   |
|  | Create a clock enable input   | Per Clock Feasibility Indicators                    |
|  | Mirror these settings on external<br>clock e0 (Required for PUL11, OUT) | <b>cO</b> c1 c2 c3 c4 c5                            |
|  | or PLL12_OUT output pins. This will disable                             | e0 e1 e2 e3   |
|  |   |   |
|  |   |   |
|  |   |   |
|  |   |   |
|  |   |   |
|  |   |   |
|  |   | Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish |
|  |   |   |

|  | MegaWizard Plug-In Manager [page 11 of 16]  |                                  |
|--|---|----------------------------------|
|  |   | About Documentation              |
| 1 Parameter 2 Output 3 EDA 4<br>Settings Clocks  | ] Summary   |                                  |
| $\operatorname{clk}\operatorname{c0}$ $\operatorname{clk}\operatorname{c1}$ $\operatorname{clk}\operatorname{c2}$ $\operatorname{clk}\operatorname{c3}$ $\operatorname{clk}$ | c4 🔪 clk c5 💙 extclk e0 💙 extclk e1 💙 extclk e2 💙 extclk e3                               | $\rangle$                        |
| altpllpll<br>inclk0 [noclk0 frequency: 60.000 MHz  | e0 - External Output Clock<br>Able to implement in Enhanced PLL<br>c0, V Lise this clock. |                                  |
| Ciki         Ratio         Ph (dg)         DC (%)           o0         1/1         0.00         50.00           e0         1/1         -63.00         50.00                  | Clock rap becomes     Reques     Enter output clock frequency:                            | Actual settings Actual settings  |
| Str  | Enter output clock parameters:     Clock multiplication factor     Clock division factor  | Copy 1 1                         |
|  | Clock phase shift -3.50   | ns v -3.50                       |
|  | Clock dgty cycle (%) 50.00<br>More Details >>   | 50.00                            |
|  | Create a clock enable input   | Per Clock Feasibility Indicators |
|  |   | c0 c1 c2 c3 c4 c5                |
|  |   | <b>eO</b> e1 e2 e3               |
|  |   | Cancel CBack Next > Finish       |

Après avoir créé le composant pll dans le système, les horloges apparraissent dans la partie « clock settings » du SOPC Builder. On peut les renommer en double-cliquant sur leur nom.

http://vhdl33.free.fr

→ Affecter la source pll.c0 du nom (name) sys\_clk et pll.e0 du nom (name) sdram\_clk\_out

| ock Settings  |          |      |   |     |
|---------------|----------|------|---|-----|
| Name          | Source   | MHz  |   | Add |
| clk_0         | External | 50,0 |   |     |
| sys_cik       | pll.c0   | 50,0 |   |     |
| sdram cik out | pll.e0   | 50,0 | - |     |

#### 2.15) Adresses et numéros d'interruptions

- → Selectionner le composant *ext\_flash* dans le champ « Modul Name »
- → Dans le champ « base » fixer 0x0 comme valeur après avoir double-cliquer.
- → Module > Lock Base Address permet de verouiller cette adresse pour la flash
- → System > Auto-Assign Base Addresses va faire de l'ordre dans la carte mémoire
- → System > Auto-Assign IRQs va organiser les numéros d'interruptions

Le système NIOS est entièrement construit, il n'y a plus d'erreurs ou de conflits signalés, presque plus de warning, on va pouvoir le générer.

| Use  | Connec                    | Module Name   | Description   | Clock | Base                | End         | IRQ |
|--|---------------------------|---|---|-------|---------------------|-------------|-----|
| <b>V</b>   |                           | cpu instruction_master data_master tag debug module | Nios II Processor<br>Avalon Memory Mapped Master<br>Avalon Memory Mapped Master<br>Avalon Memory Mapped Slave | cik_0 | IRQ 0               | IRQ 33      |     |
|  |                           | ⊟ system timer                                      | Interval Timer  |       | - 02000000          | CACCELOCITI |     |
| _  | $   \longrightarrow$      | s1  | Avalon Memory Mapped Slave  | clk_0 | ■ 0x00a01000        | 0x00a0101f  | 一向  |
|  |                           | E ext_flash<br>s1                                   | Flash Memory (CFI)<br>Avalon Memory Mapped Tristate Slave   | clk_0 | ▲ 0x00000000        | 0x007fffff  |     |
|  |                           | ext_ram<br>s1                                       | IDT71∨416 SRAM<br>Avalon Memory Mapped Tristate Slave   | clk_0 |                     | 0x009fffff  |     |
| <b>V</b>   |                           | jtag_uart<br>avalon_jtag_slave                      | JTAG UART<br>Avalon Memory Mapped Slave   | clk_0 |                     | 0x00a01067  | 一   |
|  | + <del>L</del>            | ext_ram_bus<br>avalon_slave<br>tristate_master      | Avalon-MM Tristate Bridge<br>Avalon Memory Mapped Slave<br>Avalon Memory Mapped Tristate Master               | clk_0 |                     |             |     |
|  | $   \longrightarrow$      | Icd_display<br>control_slave                        | Character LCD<br>Avalon Memory Mapped Slave   | cik_0 |                     | 0x00a0102f  |     |
| <b>V</b>   | $   \longrightarrow$      | led_pio<br>s1                                       | PIO (Parallel I/O)<br>Avalon Memory Mapped Slave  | cik_0 |                     | 0x00a0103f  |     |
| <b>V</b>   |                           | button_pio<br>s1                                    | PIO (Parallel I/O)<br>Avalon Memory Mapped Slave  | cik_0 |                     | 0x00a0104f  |     |
| <ul> <li>Image: Second sec</li></ul> | $\square \longrightarrow$ |   | PIO (Parallel I/O)<br>SDRAM Controller  | clk_0 |                     | 0x00a0105f  |     |
|  | $\rightarrow$             | s1<br>E sysid                                       | Avalon Memory Mapped Slave<br>System ID Peripheral  | cik_0 | <b>€ 0x02000000</b> | 0x02ffffff  |     |
|  | $  \rightarrow$           | control_slave                                       | Avalon Memory Mapped Slave  | clk_0 |                     | 0x00a0106f  |     |
|  |                           | 🖃 pli   | PLL   |       |                     |             |     |
|  | $  \longrightarrow$       | s1  | Avalon Memory Mapped Slave  | cik_0 | <b>■ 0x0080000</b>  | 0x0080001f  |     |

→ Faire Next puis Generate (sans simulation) . La génération est assez rapide.

→ En fin de génération réussie , on peut quitter le SOPC Builder par Exit

## 3) Synthèse et placement-routage

### 3.1) Compilation du projet Quartus

Dans Quartus , le projet au plus haut niveau est représenté par la feuille de schéma uPC1S10.bdf qui doit être ouverte. On y a déjà placé les différentes entrées-sorties avec leur noms conformes au fichier d'affectation des pins. Il ne manque que le système représenté pas son symbole.

- ➔ Dans la barre d'outil présente à gauche de la feuille de schéma, cliquer sur le symbole représentant une porte « ET » et marqué « symbol Tool »
- → Dans **Project** sélectionner *Nios2\_system* et **OK**
- ➔ On dépose le composant sur la feuille de schéma, terminant ainsi la partie construction du matériel
- → Ouvrir la fenêtre console TCL View>Utility windows>Tcl Console
- → Dans la console, vérifier par la commande pwd que vous êtes dans le bon répertoire de travail. Si c'est le cas , la commande dir vous montre la présence du fichier uPC1s10.tcl
- → Réaliser l'affectation des pins par la commande : source uPC1S10.tcl
- → Dans la barre d'outil du haut, choisir l'icône marqué « Start Compilation » (un triangle violet). Accepter de sauvegarder le projet . L'ensemble des opérations de synthèse VHDL, placement des fonctions sur des primitives Stratix1S10, et routage va s'enchainer donnant lieu à des rapports détaillés pour chacune des phases.

## 3.2) Configuration du circuit Stratix1S10

Cette opération de configuration du FPGA peut être réalisée en dehors du contexte **Quartus**, ce qui est expliqué ici est donc tout à fait optionnel. En effet, l'outil de programmation peut aussi être invoqué depuis l'**ide** outil de développement logiciel. Cette deuxième méthode sera, en pratique la plus utilisée mais la façon d'opérer est la même dans les deux cas.

- → Dans les icônes de la barre du haut, cliquer sur le troisième en partant de la droite marqué « Programmer ». L'interface de programmation apparaît.
- → Vérifier la présence de l'indication USB-Blaster à coté de Hardware Setup sinon réparer à l'aide de cette dernière commande.
- → Add File... permet de choisir le fichier de programmation uPC1S10.sof
- → Start lance le chargement de ce fichier dans le circuit Stratix1S10

| P Qu                | artus II - D:/patrice/t | ravail/TpNios81/Tp_sop    | c/uPC1s10/uPC1 | S10 - uPC1S10 - | [uPC1S10.cdf          | ]      | 0               | 00     |
|---------------------|-------------------------|---------------------------|----------------|-----------------|-----------------------|--------|-----------------|--------|
| File Edit Processin | g Tools Window          |                           |                |                 |                       |        |                 |        |
| 🔔 Hardware Setup.   | USB-Blaster [USB-       | 0]                        |                |                 |                       |        |                 |        |
| Mode:               | JTAG                    |                           |                |                 |                       |        |                 | -      |
| Progress:           |                         |                           | 100            | 1%              |                       |        |                 |        |
| Enable real-time IS | SP to allow background  | l programming (for MAX II | devices)       |                 |                       |        |                 |        |
| Mart Start          | File                    | Device                    | Checksum       | Usercode        | Program/<br>Configure | Verify | Blank-<br>Check | Examin |
| 🖬 Stop              | uPC1S10.sof             | EP1S10F780                | 00434CA9       | FFFFFFF         |                       |        |                 |        |
| Auto Detect         |                         |                           |                |                 |                       |        |                 |        |
| X Delete            |                         |                           |                |                 |                       |        |                 |        |
| 🍰 Add File          |                         |                           |                |                 |                       |        |                 |        |
| Change File         |                         |                           |                |                 |                       |        |                 |        |
| Save File           |                         |                           |                |                 |                       |        |                 |        |
| 😂 Add Device        |                         |                           |                |                 |                       |        |                 |        |
| 📲 Up                | •                       |                           |                |                 |                       |        |                 |        |
| For Help, press F1  |                         |                           |                |                 |                       |        | NUM             | 1.     |

#### 3.3) Analyse et conclusions sur la partie matérielle du projet

A ce stade, il est important d'exercer notre esprit critique et de se poser quelques questions sur les caractéristiques, qualités et défauts du matériel créé. Des réponses nous sont données dans Quartus, soit dans la fenêtre hierarchie du projet, dans le navigateur mais surtout dans la fenêtre « compilation report ».

#### Questionnaire

- ✓ Que contient le fichier Nios2\_system.ptf?
- ✔ Quel est le rôle exact de ce fichier ?
- ✓ La séquence « Compile design » ( à gauche dans le navigateur) contient 5 phases. A quoi correspond chacune de ces phases ?
- ✔ Quelles sont les ressources Stratix utilisées en terme de :
  - Nombre de pins ?
  - Nombre total d'éléments logiques ?
  - Nombre total d'éléments mémoires ?
  - Classer par ordre décroissant en termes de surface les différentes parties du NIOS en estimant un pourcentage pour chacune d'entre elles
- ✔ Quelle est la fréquence maximale théorique de fonctionnement ?
- ✔ Quel est l'élément qui limite cette fréquence ?
- Pourrait-on améliorer cette fréquence sur cette même plateforme ? Si oui, par quel moyen ?

#### 3.4) Option

A ce stade, le matériel est au point, on peut parfaitement quitter Quartus.

## 4) Logiciel associé au NIOS

Tous les programmes sont écrits en langage C ou C++. L'outil de développement est une suite Eclipse adaptée par Altera et nommée ici **NIOS II IDE.** On va créer un premier projet basé sur un exemple déjà écrit par Altera.

Attention ! Toujours vérifier que votre espace de travail logiciel est le répertoire software présent à l'intérieur de votre projet uPC1S10

#### 4.1) Construction de l'application logicielle

- → Altera >NIOS II EDS 8.1> NIOS II IDE invoque l'environnement de travail logiciel.
- → File > Switch workspace permet de vérifier que le répertoire de travail est celui souhaité (*Tp\_sopc/software*)
- → File>New Nios II C/C++ Application ouvre une fenêtre de construction du projet logiciel
- → Dans select Project template , choisir count\_binary
- → Renommer count\_binary\_0 en *count\_binary* et rechercher le fichier <u>Nios2\_system.ptf</u> comme cible pour le SOPC Buider Ptf File.
- → Next. Dans la fenêtre suivante, choisir: Select or create a system library puis cliquer sur New System Library Project
- → Dans la fenêtre Nios II System library fixer name à : uPC1S10\_syslib . Ce sera le nom de la bibliothèque commune à tous les applications logicielles associées à ce même matériel
- → Finish. On revient sur la fenêtre précédente. Finish.

|                                     | New Project 😑   |
|-------------------------------------|---|
| Nios II C/C++ Application           |   |
| D:\patrice\travail\TpNios81\Tp_s    | vich a derault system library as<br>opc\uPC1s10\software\count_binary   |
|                                     |   |
| Name: count_binary                  |   |
| Specify Location                    |   |
| Location: D:\patrice\travail\7      | pNios81\Tp_sopc\uPC1s10\software Browse   |
| Select Target Hardware.             |   |
| SOPC Builder System PTF File:       | D:\patrice\travail\TpNios81\Tp_sopc\uPC1s10\Nios2_system.p  |
| CPU:                                | cpu 💌   |
| Select Project Template             |   |
| Blank Project<br>Board Diagnostics  | Displays a running count of 0x00 to 0xff  |
| Hello Freestanding                  | Details   |
| Hello World<br>Hello World Small    | Count Binary exercises the push-button, LCD, LED, and seven-segment display peripherals. Count Binary displays a              |
| Memory Test<br>Simple Socket Server | running count of 0x00 to 0xff on output peripherals, while<br>responding to input on the push-buttons. This example runs with |
| Web Server                          | or without the MicroC/OS-II RTOS and supports hardware<br>systems that do not include all the peripherals listed.             |
|                                     | · · · · · · · · · · · · · · · · · · ·   |
|                                     |   |
|                                     |   |
| (2)                                 |   |
| U                                   |   |

http://vhdl33.free.fr

#### 4.2) Compilation

→ Dans la fenêtre navigateur, sélectionner *count\_binary* puis clique-droit et **Build** project . Cette action lance tout le processus compilation / édition de lien / création de l'éxecutable.

Remarque : Si le FPGA est déjà configuré , on peut directement se servir de la commande **Run As Nios II Hardware** qui compile et procéde au démarrage du programme.

#### 4.3) Configuration du FPGA

Cette partie est optionnelle car la configuration a pu être faite à partir de Quartus . Si ce n'est le cas, ou si la configuration a été perdue ( carte éteinte par exemple), il faut bien entendu que le FPGA soit configuré avec un système NIOS avant de penser pouvoir y faire exécuter un programme.

→ Tools > Quartus II Programmer... invoque l'outil de programmation. La marche à suivre dés lors est la même que celle expliquée précédemment.

#### 4.4) Lancement du programme

→ Dans la fenêtre navigateur, sélectionner *count\_binary* puis clique-droit et Run As> Nios II Hardware. Cette action lance le processus d'exécution du programme sur la cible NIOS.

#### Questions

- → Dans quelle mémoire se trouve le programme exécutable ? Fonctionne t-il toujours si l'on le charge dans une autre mémoire ? Comment faites-vous ?
- → Quelle est la taille mémoire du programme exécutable ? Où voit-on cette information ?
- → Que contient le fichier system.h se trouvant sous uPC1S10\_syslib>Debug>Sysyem description ?
- → Quand et comment a été créé ce fichier ?

#### 4.5) Lancement du programme avec débugger

→ Dans la fenêtre navigateur, sélectionner *count\_binary* puis clique-droit et Debug As> Nios II Hardware . Cette action lance le processus d'exécution du programme sur la cible NIOS avec débugger. Un point d'arrêt est positionné en début de programme. On peut ensuite faire du pas à pas, visualiser le code assembleur, l'état des variables , poser d'autres point d'arrêt. Pour les détails du débugger, on se réfèrera à la documentation de l'ide se trouvant à l'adresse suivante : <u>http://127.0.0.1:60955/help/index.jsp</u> et on explorera la rubrique **Tutorials** > **Debugging the Project** 

### 4.6) Simulation de l'exécution du programme (ISS)

→ Dans la fenêtre navigateur, sélectionner *count\_binary* puis clique-droit et Run As> Nios II Instruction Set Simulator.