VHDLib

h1. +*VHDLIB*+

Child pages:

- Contribution Guide
- IP documentation
- Mini-LFR
- Mini LFR - Bitstream Generation
- Releases
  - Release 1
- Setup

h1. Contribution Guide

You are very welcome to contribute to the VHDLib project. For any contribution you should respect the following rules.

h2. Signal naming convention

Please use meaningful names for all your signals and modules names.
Active low signals should be named with a trailing 'n', for example an active low reset should be named 'resetn'.

h2. Library structure

Any new module should be placed first in the staging folder and integrated only when it's interfaces are locked.

h1. IP Documentation

---

h2. Sample Type

Sample TypeSample Type

```
TYPE Samples IS ARRAY(NATURAL RANGE <>) OF STD_LOGIC_VECTOR(15 DOWNTO 0);

SUBTYPE Samples24 IS STD_LOGIC_VECTOR(23 DOWNTO 0);
SUBTYPE Samples16 IS STD_LOGIC_VECTOR(15 DOWNTO 0);
SUBTYPE Samples14 IS STD_LOGIC_VECTOR(13 DOWNTO 0);
SUBTYPE Samples12 IS STD_LOGIC_VECTOR(11 DOWNTO 0);
SUBTYPE Samples10 IS STD_LOGIC_VECTOR( 9 DOWNTO 0);
SUBTYPE Samples8  IS STD_LOGIC_VECTOR( 7 DOWNTO 0);

TYPE Samples24v IS ARRAY(NATURAL RANGE <>) OF Samples24;
TYPE Samples16v IS ARRAY(NATURAL RANGE <>) OF Samples16;
TYPE Samples14v IS ARRAY(NATURAL RANGE <>) OF Samples14;
TYPE Samples12v IS ARRAY(NATURAL RANGE <>) OF Samples12;
TYPE Samples10v IS ARRAY(NATURAL RANGE <>) OF Samples10;
TYPE Samples8v  IS ARRAY(NATURAL RANGE <>) OF Samples8;
```

---

h2. General Purpose

h3. Edge Detection

Edge_DetectionEdge_Detection

EdgeDetection permit to detect the edge of the sin signal.

p=. !{width: 20%}edge_detection.png!

```
COMPONENT lpp_edge_detection
PORT (
clk  : IN  STD_LOGIC;
rstn : IN  STD_LOGIC;
sin  : IN  STD_LOGIC;
sout : OUT STD_LOGIC);
END COMPONENT;
```

.Signal	.Direction	.Size	.Function	_. Active
clk	input	1	clock domain 1	rising edge
rstn	input	1	reset	low
sin	input	1	signal in	
sout	ouput	1	signal out	

Edge_to_levelEdge_to_level

EdgeToLevel permit to transform the positive edge information into a level information.

p=. !{width: 20%}edge_to_level.png!

```
COMPONENT lpp_edge_to_level
PORT (
clk  : IN  STD_LOGIC;
rstn : IN  STD_LOGIC;
sin  : IN  STD_LOGIC;
sout : OUT STD_LOGIC);
END COMPONENT;
```

.Signal	.Direction	.Size	.Function	_. Active

clk	input	1	clock domain 1	rising edge
rstn	input	1	reset	low
sin	input	1	signal in	
sout	ouput	1	signal out	

h3. Synchronizer

SYNC_FFSYNC_FF

Sync_FF permit to synchronize a signal A in the clock domain clk. Normally, A signal should be the output of a FF cloked in an other domain. You shouldtn't have "logic" between the 2 domain.
You can configure the number FF use to synchronize (NB_FF_OF_SYNC). This number is depending of the MTBF(Mean Time Between Failure).

p=. !{width: 15%}SYNC_FF.png!

```
COMPONENT SYNC_FF_LPP_JCP
GENERIC (
NB_FF_OF_SYNC : INTEGER);
PORT (
clk    : IN  STD_LOGIC;
rstn   : IN  STD_LOGIC;
A      : IN  STD_LOGIC;
A_sync : OUT STD_LOGIC);
END COMPONENT;
```

.Parameter	.Type	.Size	.Description	.Default
NB_FF_OF_SYNC	Integer		Number of FF	2

|\5.|

.Signal	.Direction	.Size	.Function	. Active
clk	input	1	clock	rising edge
rstn	input	1	reset	low
sin	input	1	signal in	
sout	ouput	1	signal synchronized	

SYNC_VALID_BITSYNC_VALID_BIT

SYNC_VALID_BIT permit to synchronize a signal of validity from clock domain clk_in to clock domain clk_out. A validity bit is a signal set "high" only one cycle and zero others. To Synchronize this type of signal, a first stage detect the positive edge, a second synchronizes this signal, and a last transform the edge information to a validity bit.
You can configure the FF number of the second stage (NB_FF_OF_SYNC).

p=. !{width: 20%}SYNC_VALID_BIT.png!

```
COMPONENT SYNC_VALID_BIT_LPP_JCP
GENERIC (
NB_FF_OF_SYNC : INTEGER);
PORT (
clk_in  : IN  STD_LOGIC;
clk_out : IN  STD_LOGIC;
rstn    : IN  STD_LOGIC;
sin     : IN  STD_LOGIC;
sout    : OUT STD_LOGIC);
END COMPONENT;
```

.Parameter	.Type	.Size	.Description	.Default
NB_FF_OF_SYNC	Integer		Number of FF	2

|\5.|

.Signal	.Direction	.Size	.Function	. Active
clk_in	input	1	clock domain 1	rising edge
clk_out	input	1	clock domain 1	rising edge
rstn	input	1	reset	low
sin	input	1	valid bit clocked in domain 1	
sout	output	1	valid bit clocked in domain 2	

h2. SoC (System On Chip)

Leon3_SocLeon3_Soc

Leon3_SoC is an IP which integrate all the basic IP for using a Leon3 System. This System is configurable :

- activate the DSU, AHB uart, APB UART, IRQ manager and timer manager
- activate the FPU and select the type of IP using for (netlist or rtl)

You can connect easily external AMBA IP. For example, if you have only one Leon3 and you want to add an AHB Master My_AHB_MST. You set NB_AHB_MASTER to 1 and connect My_AHB_MST_ahbmi signal to the input ahbi_m_ext and My_AHB_MST_ahbmo to ahbo_m_ext(1).

```
|.Number          |.NAME     |.Enable Parameter |.Address  |_.IRQ|
|\5=.AHB Master |
| 0 to NCPU-1        |leon3s     |                   |||
| NCPU+NB_AHBMASTER  |ahbuart    |ENABLE_AHB_UART    |||
|\5.|
|\5=.AHB Slave |
| 0                  |mctrl      |                   |0x00000000 |     |
| 1                  |apbctrl    |                   |0x80000000 |     |
| 2                  |dsu3       |ENABLE_DSU         |0x90000000 |0    |
|\5.|
|\5=.APB Slave |
| 0                  |mctrl      |                   |0x80000000 |     |
| 1                  |apbuart    |ENABLE_APB_UART    |0x80000100 |2    |
| 2                  |irqmp      |ENABLE_IRQMP       |0x80000200 |     |
| 3                  |gptimer    |ENABLE_GPT         |0x80000300 |8    |
| 4                  |ahbuart    |ENABLE_APB_UART    |0x80000400 |     |
```

p=. !{width: 40%}leon3_SoC.png!

```
COMPONENT leon3_soc_LPP_JCP
GENERIC (
fabtech         : INTEGER;
memtech         : INTEGER;
padtech         : INTEGER;
clktech         : INTEGER;
disas           : INTEGER;
dbguart         : INTEGER;
pclow           : INTEGER;
clk_freq        : INTEGER;
NB_CPU          : INTEGER;
ENABLE_FPU      : INTEGER;
FPU_NETLIST     : INTEGER;
ENABLE_DSU      : INTEGER;
ENABLE_AHB_UART : INTEGER;
ENABLE_APB_UART : INTEGER;
ENABLE_IRQMP    : INTEGER;
ENABLE_GPT      : INTEGER;
NB_AHB_MASTER   : INTEGER;
NB_AHB_SLAVE    : INTEGER;
NB_APB_SLAVE    : INTEGER);
PORT (
clk        : IN    STD_ULOGIC;
rstn       : IN    STD_ULOGIC;
errorn     : OUT   STD_ULOGIC;
ahbrxd     : IN    STD_ULOGIC;
ahbtxd     : OUT   STD_ULOGIC;
urxd1      : IN    STD_ULOGIC;
utxd1      : OUT   STD_ULOGIC;
address    : OUT   STD_LOGIC_VECTOR(19 DOWNTO 0);
data       : INOUT STD_LOGIC_VECTOR(31 DOWNTO 0);
nSRAM_BE0  : OUT   STD_LOGIC;
nSRAM_BE1  : OUT   STD_LOGIC;
nSRAM_BE2  : OUT   STD_LOGIC;
nSRAM_BE3  : OUT   STD_LOGIC;
nSRAM_WE   : OUT   STD_LOGIC;
nSRAM_CE   : OUT   STD_LOGIC;
nSRAM_OE   : OUT   STD_LOGIC;
apbi_ext   : OUT   apb_slv_in_type;
apbo_ext   : IN    soc_apb_slv_out_vector(NB_APB_SLAVE-1+5 DOWNTO 5);
ahbi_s_ext : OUT   ahb_slv_in_type;
ahbo_s_ext : IN    soc_ahb_slv_out_vector(NB_AHB_SLAVE-1+3 DOWNTO 3);
ahbi_m_ext : OUT   AHB_Mst_In_Type;
ahbo_m_ext : IN    soc_ahb_mst_out_vector(NB_AHB_MASTER-1+NB_CPU DOWNTO NB_CPU));
END COMPONENT;
```

.Parameter	.Type	.Size	.Description	.Default
fabtech	INTEGER		Target technologie	apa3e
memtech	INTEGER		Memory Target technologie	apa3e
padtech	INTEGER		Pad Target technologie	inferred
clktech	INTEGER		Clock target technologie	inferred
disas	INTEGER		Activate the disassembler	0
dbguart	INTEGER		Activate debug uart	0
pclow	INTEGER			2
clk_freq	INTEGER		Clock input frequency (in kHz)	25000
NB_CPU	INTEGER		Number of Leon3	1
ENABLE_FPU	INTEGER		Enable the FPU	1
FPU_NETLIST	INTEGER		Select the FPU Used (1=> NetList, 0=> RTL)	1
ENABLE_DSU	INTEGER		Enable the Debug System Unit	1
ENABLE_AHB_UART	INTEGER		Enable AHB UART	1
ENABLE_APB_UART	INTEGER		Enable APB UART	1
ENABLE_IRQMP	INTEGER		Enable irq manager	1
ENABLE_GPT	INTEGER		Enable the timer	1
NB_AHB_MASTER	INTEGER		Number of AHB Master outside the SoC	0
NB_AHB_SLAVE	INTEGER		Number of AHB Slave outside the SoC	0
NB_APB_SLAVE	INTEGER		Number of APB Slave outside the SoC	0

|\5.|

.Signal	.Direction	.Size or Type	.Function	. Active
clk	input	1	clock	rising edge
rstn	input	1	reset	low
errorn	output	1	leon 3 error signal	
ahbrxd	input	1	AHB uart Rx Signal	
ahbtxd	output	1	AHB uart Tx Signal	
urxd1	input	1	APB uart Rx Signal	
utxd1	output	1	APB uart Tx Signal	
address	output	20	SRam Address	
data	inout	32	SRam Data	
nSRAM_BE0	output	1	SRam bankEnable 0	
nSRAM_BE1	output	1	SRam bankEnable 1	
nSRAM_BE2	output	1	SRam bankEnable 2	
nSRAM_BE3	output	1	SRam bankEnable 3	
nSRAM_WE	output	1	SRam WriteEnable	
nSRAM_CE	output	1	SRam ChipEnable	
nSRAM_OE	output	1	SRam OutputEnable	
apbi_ext	output	apb_slv_in_type	APB Slave bus input signal	
apbo_ext	input	NB_APB_SLAVE of apb_slv_out_type	APB Slave bus output signal	
ahbi_s_ext	output	ahb_slv_in_type	AHB Slave bus input signal	
ahbo_s_ext	input	NB_AHB_SLAVE of ahb_slv_out_type	AHB Slave bus output signal	
ahbi_m_ext	output	ahb_mst_In_Type	AHB Master bus input signal	
ahbo_m_ext	input	NB_AHB_MASTER of ahb_mst_out_type	AHB Master bus output signal	

---

h2. AMBA Peripherals

APB LFR Time ManagmentAPB LFR Time Managment

LFR Time management is a real time clock. The time is give by the coarse_time and fine_time value. The Time managment is like a big counter of 48b (coarse_time + fine_time). This big counter count each nb_wait_period of period clk49_152MHz.
The bit 0 of coarse_time is the second.

The Time mangment can be updated by register with the value into COARSE_TIME_LOAD. When a space_wire_tick or a CTRL.force_tick is asserted, the value into COARSE_TIME_LOAD is loaded in COARSE_TIME and the FINE_TIME is reset to 0.

|.VENDOR_ID |TBD |
|.DEVICE_ID |TBD |
|\2.|
.APB_ADDRESS	.Register Name	.Access	.Description	\2.*Field*	_.Reset Value
/2.0x00	/2. CTRL	/2.RW	/2.Control register	0	force_tick	/2.0x0
31-1	Unused
0x04	COARSE_TIME_LOAD	RW	CoarseTime to load after the next "tick"	31-0	Coarse_time	0x800000000
/2.0x08	/2. COARSE_TIME	/2.R	Current CoarseTime	30-0	Coarse_time	/2.
Indicates if the current coarse_time is not "in phase" with the SPW	31	Not_SPW

|/2.0x0C |/2. FINE_TIME |/2.R |/2.Current FineTime |15-0 | FineTime |/2. |
|31-16| Unused |

```
COMPONENT apb_lfr_time_management_LPP_JCP IS
GENERIC(
pindex        : INTEGER := 0;
paddr         : INTEGER := 0;
pmask         : INTEGER := 16#fff#;
pirq          : INTEGER := 0;
nb_wait_period : INTEGER := 375
);
PORT (
clk25MHz     : IN  STD_LOGIC;
clk49_152MHz : IN  STD_LOGIC;
resetn       : IN  STD_LOGIC;
grspw_tick   : IN  STD_LOGIC;
apbi         : IN  apb_slv_in_type;
apbo         : OUT apb_slv_out_type;
coarse_time  : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
fine_time    : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
);
END COMPONENT;
```

Files

SYNC_FF.png	11.8 KB	27/02/2014	Jean-Christophe Pellion
SYNC_VALID_BIT.png	16.9 KB	27/02/2014	Jean-Christophe Pellion
edge_detection.png	10.3 KB	27/02/2014	Jean-Christophe Pellion
edge_to_level.png	12.1 KB	27/02/2014	Jean-Christophe Pellion
SYNC_VALID_BIT.png	16.9 KB	27/02/2014	Jean-Christophe Pellion
leon3_SoC.png	36.8 KB	19/03/2014	Jean-Christophe Pellion

Mini LFR - Bitstream Generation Procedure

1 Generates scripts files

Clean the project directory, and run the makefile scripts command to generate all project's files.

```
# clean the project and re-generates the scripts
$> make distclean scripts
```

The files MINI_LFR_top_libero.prj should be created.

2 Launch Libero IDE

Open the project MINI_LFR_top_libero.prj with Libero IDE v9.1.
Capture1.PNG

2.1 Synthesis

Synthesis tools should be configured to use **Synplify** version **2012-03A-SP1-2**.

Launch the synthesis step (You must not add constraint file).
In the synplify Pro windows, click on the Run button.

The project status obtained should be :

Capture2.PNG

> You must verfify the block RAM's usage. The expected value is 100.
>
> If it's around 60/70, LEON3 processor is not mapped. In this case, you must clean your project and restart the generation procedure.

You can close Synplify and return to Libero.

If the Sythesis step is green like that :

You must activate the option Detect new files on disk automatically in the project settings.

Capture3.PNG
Capture4.PNG

2.2 Place&Route

Launch the Place&Route step. You must add the constraint files :

- default.pdc (Input/Output constraint)
- MINI-LFR_PlaceAndRoute.sdc (Timing constraint)

Capture5.PNG

2.2.1 Compile Step

Capture6.PNG
Run the Compile step.

2.2.2 Layout Step

Capture7.PNG
Run the Layout step.

In the Layout options windows, select Advanced Layout Options :

Capture8.PNG
And checked those options :

Capture9.PNG
Click Ok and wait...

Capture10.PNG

2.2.3 Timing Analyser

Launch theTiming Analyser.

Capture11.PNG

2.2.3.a SpaceWire Output

You must verify the SpaceWire output timing for the Max and Min delay.

Capture12.PNG
Capture19.PNG
The ouput skew is equal to Ouput_SOut timing - Ouput_DOut timing. This output skew must be positive.

In this exemple,

SPW_NOM skew max = 13.907 - 15.882 = -1.975 ns

SPW_NOM skew min = 6.367 - 7.317 = -0.950 ns

SPW_RED skew max = 17.937 - 13.510 = 4.427 ns

SPW_RED skew min = 8.304 - 6.175 = 2.129 ns

The SPW_NOM interface must be modified. In ChipPlanner, we will move element in the SPW_NOM path to have a positive skew in min and max delay.
Capture13.PNG
Capture14.PNG

2.2.3.b SpaceWire Input

You must also verify the SpaceWire input timing. For that, you must add a new set for SPW_INPUT:

Capture15.PNG
and configure it like that :
you
Capture16.PNG

You obtain those timing for min and max delay.

Capture17.PNG
Capture18.PNG
You must also add a new set for the FF setup time :

Capture20.PNG
Capture21.PNG
In resume, for the input SPW_NOM interface :

```
        <tr>
        <th colspan=2> Signal Type </th>        <th> max delay</th> <th>min delay</th>
        </tr>
    </thead>
    <thead>
        <tr>
        <th colspan=4 align="center" bgcolor="#C0C0C0"> SPW_NOM_IN </th>
        </tr>
    </thead>
    <tbody>
        <tr>    <td rowspan=4>r_FF</td>     <td>Strobe to CLK</td>  <td>7.230 ns</td>   <td>3.497 ns</td>   </
tr>
    <tr>                    <td>Data   to CLK</td>  <td>8.046 ns</td>   <td>3.807 ns</td>   </tr>
    <tr>                    <td>Data   to D  </td>  <td>1.700 ns</td>   <td>0.692 ns</td>   </tr>
    <tr>                    <td>D to Q  </td>   <td>0.888 ns</td>   <td>0.413 ns</td>   </tr>

    <tr>    <td rowspan=4>nr_FF</td>    <td>Strobe  to CLK</td> <td>7.315 ns</td>   <td>3.543 ns</td>   </tr>
    <tr>                    <td>Data    to CLK</td> <td>8.131 ns</td>   <td>3.853 ns</td>   </tr>
    <tr>                    <td>Data    to D  </td> <td>1.767 ns</td>   <td>0.724 ns</td>   </tr>
    <tr>                    <td>D to Q  </td>   <td>0.888 ns</td>   <td>0.413 ns</td>   </tr>
    </tbody>
    <thead>
        <tr>
        <th colspan=4 align="center" bgcolor="#C0C0C0"> SPW_RED_IN </th>
        </tr>
    </thead>
    <tbody>
        <tr>    <td rowspan=4>r_FF</td>     <td>Strobe to CLK</td>  <td>11.601 ns</td>  <td>6.217 ns</td>   </
tr>
    <tr>                    <td>Data   to CLK</td>  <td>12.845 ns</td>  <td>5.488 ns</td>   </tr>
    <tr>                    <td>Data   to D  </td>  <td>2.799 ns</td>   <td>1.246 ns</td>   </tr>
    <tr>                    <td>D to Q  </td>   <td>0.888 ns</td>   <td>0.413 ns</td>   </tr>

    <tr>    <td rowspan=4>nr_FF</td>    <td>Strobe  to CLK</td> <td>11.628 ns</td>  <td>6.236 ns</td>   </tr>
    <tr>                    <td>Data    to CLK</td> <td>12.872 ns</td>  <td>5.507 ns</td>   </tr>
    <tr>                    <td>Data    to D  </td> <td>2.798 ns</td>   <td>1.246 ns</td>   </tr>
    <tr>                    <td>D to Q  </td>   <td>0.888 ns</td>   <td>0.413 ns</td>   </tr>
    </tbody>
```

The input skew is equal to : min(Time from strobe to CLK; Time from data to CLK) - Time Data to FF - FF Setup Time. This input skew must be positive.

```
        <tr>
        <th> skew </th>         <th colspan=2>max delay</th>    <th colspan=2>min delay</th>
        </tr>
    </thead>
    <thead>
        <tr>
        <th colspan=5 align="center" bgcolor="#C0C0C0"> SPW_NOM_IN </th>
        </tr>
    </thead>
    <tbody>
        <tr>    <td>skew r_FF</td>  <td>4.642 ns</td>   <td bgcolor="#00FF00">OK</td>   <td>2.392 ns</td>   <t
d bgcolor="#00FF00">OK</td>   </tr>
        <tr>    <td>skew nr_FF</td>     <td>4.660 ns</td>   <td bgcolor="#00FF00">OK</td>   <td>2.406 ns</td>
  <td bgcolor="#00FF00">OK</td>   </tr>
    </tbody>
    <thead>
        <tr>
        <th colspan=5 align="center" bgcolor="#C0C0C0"> SPW_RED_IN </th>
        </tr>
    </thead>
    <tbody>
        <tr>    <td>skew r_FF</td>  <td>7.914 ns</td>   <td bgcolor="#00FF00">OK</td>   <td>3.829 ns</td>   <t
d bgcolor="#00FF00">OK</td>   </tr>
        <tr>    <td>skew nr_FF</td>     <td>7.942 ns</td>   <td bgcolor="#00FF00">OK</td>   <td>3.848 ns</td>
  <td bgcolor="#00FF00">OK</td>   </tr>
    </tbody>
```

In our case, all SPW input skew are positive.
If MIN or MAX skew delay are not positive, you must launch the ChipPlanner tool to move r_FF and nr_FF closest to the CLK source and move XOR gate far away from input pads (SIN, DIN), nrFF:D and r_FF:D.

You can close the Timing Analyser.

2.2.4 Chip Planner

Launch ChipPlanner to move element in the netlist and try to fit the timing requirements (skew delay for the SPW_NOM Output interface).

Capture22.PNG
In our case, we want to increase SPW_NOM\ output skew. For that, we must increase Ouput_SOut timing and reduce the Ouput_DOut timing.

To reduce the output dout delay, we will select the last FF before the SPW_NOM_OUTPUT :

Capture23.PNG
and move closest to the ouput PAD :

Capture24.PNG
To increase the output SOut delay, we will select the last FF before the SPW_NOM_OUTPUT :

Capture25.PNG
and move far away from the ouput PAD :

Capture26.PNG
Now, you can clicked on Commit and Check button and close ChipPlanner.

You must relaunch the Layout step with those selected options :

Capture27.PNG
And finally, checked the timing update with the Timing Analyser.

2.2 Bitstream generation

Click on the Programming File tool.

Capture28.PNG
It's done !

Files

Capture2.PNG	39.8 KB	01/12/2016	Jean-Christophe Pellion
Capture3.png	21.5 KB	01/12/2016	Jean-Christophe Pellion
Capture4.PNG	29.5 KB	01/12/2016	Jean-Christophe Pellion
Capture5.PNG	28.3 KB	01/12/2016	Jean-Christophe Pellion
Capture6.PNG	7.37 KB	01/12/2016	Jean-Christophe Pellion
Capture7.PNG	7.59 KB	01/12/2016	Jean-Christophe Pellion
Capture8.PNG	28.4 KB	01/12/2016	Jean-Christophe Pellion
Capture9.PNG	20.1 KB	01/12/2016	Jean-Christophe Pellion
Capture10.PNG	7.75 KB	01/12/2016	Jean-Christophe Pellion
Capture11.PNG	2.52 KB	01/12/2016	Jean-Christophe Pellion
Capture1.PNG	144 KB	01/12/2016	Jean-Christophe Pellion
Capture12.PNG	26.7 KB	01/12/2016	Jean-Christophe Pellion
Capture13.PNG	19 KB	01/12/2016	Jean-Christophe Pellion
Capture14.PNG	18.3 KB	01/12/2016	Jean-Christophe Pellion
Capture15.png	10.2 KB	01/12/2016	Jean-Christophe Pellion
Capture16.PNG	18.3 KB	01/12/2016	Jean-Christophe Pellion
Capture17.PNG	23 KB	01/12/2016	Jean-Christophe Pellion
Capture18.PNG	21.2 KB	01/12/2016	Jean-Christophe Pellion
Capture19.PNG	19.7 KB	01/12/2016	Jean-Christophe Pellion
Capture20.PNG	11.7 KB	01/12/2016	Jean-Christophe Pellion
Capture21.PNG	11.2 KB	01/12/2016	Jean-Christophe Pellion
Capture22.PNG	4.15 KB	01/12/2016	Jean-Christophe Pellion
Capture23.PNG	48.8 KB	01/12/2016	Jean-Christophe Pellion
Capture24.PNG	14.2 KB	01/12/2016	Jean-Christophe Pellion
Capture24.PNG	14.2 KB	02/12/2016	Jean-Christophe Pellion
Capture26.PNG	21.9 KB	02/12/2016	Jean-Christophe Pellion
Capture27.PNG	8.24 KB	02/12/2016	Jean-Christophe Pellion
Capture25.PNG	9.51 KB	02/12/2016	Jean-Christophe Pellion
Capture28.PNG	7.74 KB	02/12/2016	Jean-Christophe Pellion

h1. Releases

Child pages:

- Release 1

h1. Release 1

h2. Release requirements

h1. Setup

First you need to install the "Grlib":http://www.gaisler.com/index.php/downloads/leongrlib and follow the instruction in the grlib manual. Once you have a working Grlib, you can install and start to use the VHDlib.