VHDLib - IP_documentation - # 38

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 |1 |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 |1 |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