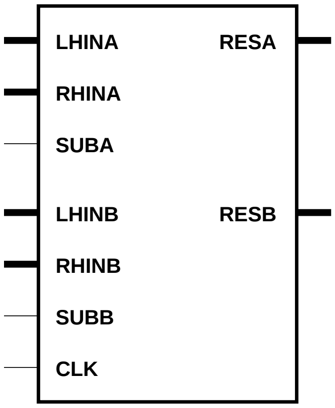
The LNS ALU
A Brief HW Design Description
version 0.0.1a, pre-release

Milan Tichý

June 23, 2003

LADDSUB_32, LADDSUB_19



Component Name:	laddsub_32 laddsub_19
Latency:	8 clock cycles
LHINA	32/19
RHINA	32/19
SUBA	1
RESA	32/19
LHINB	32/19
RHINB	32/19
SUBB	1
RESB	32/19
CLK	1

VHDL

```
-- COMPONENT declaration
component laddsub_19
  port (
    lhina : in unsigned(18 downto 0);
    rhina : in unsigned(18 downto 0);
    suba  : in std_logic;
    resa  : out unsigned(18 downto 0);
    lhinb : in unsigned(18 downto 0);
    rhinb : in unsigned(18 downto 0);
    subb  : in std_logic;
    resb  : out unsigned(18 downto 0);
    clk   : in std_logic
  );
end component;

-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of laddsub_19 : component is true;

-- INSTANTIATION Template
your_instance_name : laddsub_19
  port map (
    lhina => lhina,
    rhina => rhina,
    suba  => suba,
    resa  => resa,
    lhinb => lhinb,
    rhinb => rhinb,
    subb  => subb,
    resb  => resb,
    clk   => clk
  );
```

Handel-C

INPUT PORTS

```
/* interface for ADD/SUB component */

static lns_t las_l_ina = LNS_ZERO;
static lns_t las_r_ina = LNS_ZERO;

static lns_t las_l_inb = LNS_ZERO;
static lns_t las_r_inb = LNS_ZERO;

static wire_t las_suba = ADD;
static wire_t las_subb = ADD;
```

INPUT PORTS, non-registered input

```
signal <lns_t> las_l_ina = LNS_ZERO;
signal <lns_t> las_r_ina = LNS_ZERO;

signal <lns_t> las_l_inb = LNS_ZERO;
signal <lns_t> las_r_inb = LNS_ZERO;

signal <wire_t> las_suba = ADD;
signal <wire_t> las_subb = ADD;
```

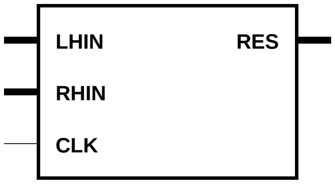
Target: VHDL, EDIF

```
interface LADDSUB (lns_t resa, lns_t resb)
     las (lns_t lhina = (lns_t) las_l_ina,
          lns_t rhina = (lns_t) las_r_ina,
          wire_t suba = (wire_t) las_suba,
          lns_t lhinb = (lns_t) las_l_inb,
          lns_t rhinb = (lns_t) las_r_inb,
          wire_t subb = (wire_t) las_subb,
          wire_t clk = __clock);
```

Target: SIMULATOR

```
interface LADDSUB (lns_t resa with {extfunc = "PlugInGet"},
                   lns_t resb with {extfunc = "PlugInGet"})
     las (lns_t lhina = (lns_t) las_l_ina,
          lns_t rhina = (lns_t) las_r_ina,
          wire_t suba = (wire_t) las_suba,
          lns_t lhinb = (lns_t) las_l_inb,
          lns_t rhinb = (lns_t) las_r_inb,
          wire_t subb = (wire_t) las_subb,
          wire_t clk = __clock with {extfunc = "PlugInClock"})
     with {extlib = LADDSUB_PLG,
             extinst = "las",
             extfunc = "PlugInSet"};
```

LMUL(E)_PREC, LDIV(E)_PREC



Component Name:	lmul(e)_PREC ldiv(e)_PREC
Latency:	1 clock cycle
LHIN	32/19
RHIN	32/19
RES	32/19
CLK	1

VHDL

```
-- COMPONENT declaration
component ldive_19
  port (
    lhin : in unsigned(18 downto 0);
    rhin : in unsigned(18 downto 0);
    res  : out unsigned(18 downto 0);
    clk  : in std_logic
  );
end component;

-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of ldive_19 : component is true;

-- INSTANTIATION Template
your_instance_name : ldive_19
  port map (
    lhin => lhin,
    rhin => rhin,
    res  => res,
    clk  => clk
  );
```

Handel-C

INPUT PORTS

```
/* interface for LOG DIVE component */

static lns_t ldive_l_in = LNS_ZERO;
static lns_t ldive_r_in = LNS_ZERO;
```

INPUT PORTS, non-registered input

```
signal <lns_t> ldive_l_in = LNS_ZERO;
signal <lns_t> ldive_r_in = LNS_ZERO;
```

Target: VHDL, EDIF

```
interface LDIVE (lns_t res)
     ldive (lns_t lhin = (lns_t) ldive_l_in,
            lns_t rhin = (lns_t) ldive_r_in,
            wire_t clk = __clock);
```

Target: SIMULATOR

```
interface LDIVE (lns_t res with {extfunc = "PlugInGet"})
     ldive (lns_t lhin = (lns_t) ldive_l_in,
            lns_t rhin = (lns_t) ldive_r_in,
            wire_t clk = __clock with {extfunc = "PlugInClock"})
   with {extlib = LDIVE_PLG,
           extinst = "ldive",
           extfunc = "PlugInSet"};
```

LSQRT(E)_PREC, LDNOR_PREC

Component Name:	lsqrt(e)_PREC ldnor_PREC
Latency:	1 clock cycle
DIN	32/19
RES	32/19
CLK	1

VHDL

```
-- COMPONENT declaration
component lsqrt_32
  port (
    din : in unsigned(31 downto 0);
    res : out unsigned(31 downto 0);
    clk : in std_logic
  );
end component;

-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of lsqrt_32 : component is true;

-- INSTANTIATION Template
your_instance_name : lsqrt_32
  port map (
    din => din,
    res => res,
    clk => clk
  );
```

Handel-C

INPUT PORTS

```
/* interface for LOG SQRT component */

static lns_t lsqrt_in = LNS_ZERO;
```

INPUT PORTS, non-registered input

```
signal <lns_t> lsqrt_in = LNS_ZERO;
```

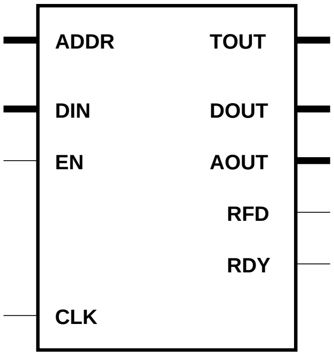
Target: VHDL, EDIF

```
interface LSQRT (lns_t res)
     lsqrt (lns_t din = (lns_t) lsqrt_in,
            wire_t clk = __clock);
```

Target: SIMULATOR

```
interface LSQRT (lns_t res with {extfunc = "PlugInGet"})
     lsqrt (lns_t din = (lns_t) lsqrt_in,
            wire_t clk = __clock with {extfunc = "PlugInClock"})
   with {extlib = LSQRT_PLG,
           extinst = "lsqrt",
           extfunc = "PlugInSet"};
```

ILCNV_TBSX_PREC



Component Name:	ilcnv_tbsX_PREC
Latency:	see below
ADDR	8
TOUT	32/19
DIN	32/19
EN	1
DOUT	32/19
AOUT	8
RFD	1
RDY	1
CLK	1

19-bit version

uses 2 conversion tables
components `ilcnv_tbs[01]_19` components

32-bit version

uses 3 conversion tables
components `ilcnv_tbs[012]_32` components

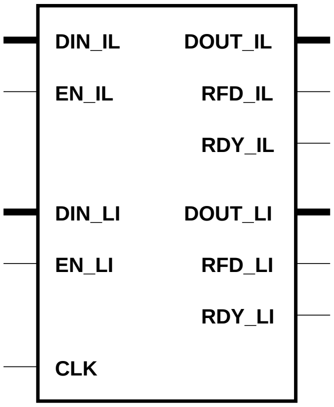
ilcnv_tbsX_PREC	ADDR	⇒	TOUT	Latency	**1** clock cycle
ilcnv_tbs0_PREC	DIN	⇒	[AD]OUT	Latency	**9** clock cycles
ilcnv_tbs1_19	DIN	⇒	[AD]OUT	Latency	**10** clock cycles
ilcnv_tbs[12]_32	DIN	⇒	[AD]OUT	Latency	**10** clock cycles

VHDL

```
-- COMPONENT declaration
component ilcnv_tbs1_32
  port (
    addr : in unsigned(7 downto 0);
    aout : out unsigned(7 downto 0);
    clk  : in std_logic;
    din  : in unsigned(31 downto 0);
    dout : out unsigned(31 downto 0);
    en   : in std_logic;
    rdy  : out std_logic;
    rfd  : out std_logic;
    tout : out unsigned(31 downto 0)
  );
end component;

-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of ilcnv_tbs1_32 : component is true;
```

ILCNV_19



Component Name:	ilcnv_19
Latency:	see below
DIN_IL	32
EN_IL	1
DOUT_IL	19
RFD_IL	1
RDY_IL	1
DIN_LI	19
EN_LI	1
DOUT_LI	32
RFD_LI	1
RDY_LI	1
CLK	1

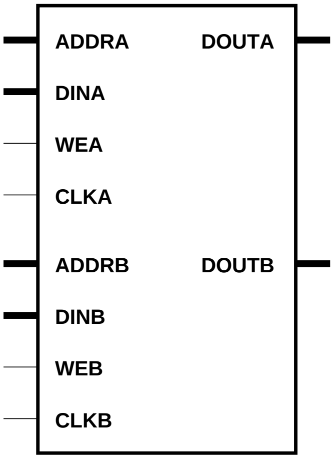
DIN_IL	⇒	**DOUT_IL**	fully pipelined	Latency	**10** clock cycles
DIN_LI	⇒	**DOUT_LI**	**8** cyc. input lat.	Latency	**28** clock cycles

VHDL

```
-- COMPONENT declaration
component ilcnv_19
  port (
    clk : in std_logic;
    din_il : in unsigned(31 downto 0);
    din_li : in unsigned(18 downto 0);
    dout_il : out unsigned(18 downto 0);
    dout_li : out unsigned(31 downto 0);
    en_il : in std_logic;
    en_li : in std_logic;
    rdy_il : out std_logic;
    rdy_li : out std_logic;
    rfd_il : out std_logic;
    rfd_li : out std_logic
  );
end component;

-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of ilcnv_19 : component is true;
```

Dual-Port Block Memory



Component Name:	dpbramWxD
Latency:	1 clock cycle
ADDRA	8/9/10/11/12
DINA	16/18/19/32
WEA	1
DOUTA	16/18/19/32
CLKA	1
ADDRB	8/9/10/11/12
DINB	16/18/19/32
WEB	1
DOUTB	16/18/19/32
CLKB	1

VHDL

See CORE Generator documentation for the configuration of Dual-Port Block Memory components.

```
-- COMPONENT declaration
component dpbram32x1024
  port (
    addra: IN std_logic_VECTOR(9 downto 0);
    addrb: IN std_logic_VECTOR(9 downto 0);
    clka: IN std_logic;
    clkb: IN std_logic;
    dina: IN std_logic_VECTOR(31 downto 0);
    dinb: IN std_logic_VECTOR(31 downto 0);
    douta: OUT std_logic_VECTOR(31 downto 0);
    doutb: OUT std_logic_VECTOR(31 downto 0);
    wea: IN std_logic;
    web: IN std_logic
  );
end component;

-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of dpbram32x1024: component is true;
-- INSTANTIATION Template
```

Handel-C

The example here introduces only instantiation of general Dual-Port Block Memory component. For more information on data types and other parameters see Handel-C header file bram.hch that can be found in examples.

INPUT PORTS

```
/* Dual Port Block Memory interface: u_in_bram component */

signal <bram_addr_t> sig_u_in_addra = 0;
signal <bram_addr_t> sig_u_in_addrb = 0;
signal <bram_data_t> sig_u_in_dina = 0;
signal <bram_data_t> sig_u_in_dinb = 0;
signal <wire_t> sig_u_in_wea = BRAM_WD;
signal <wire_t> sig_u_in_web = BRAM_WD;

/* rename u_in_bram component output registers */
#define u_in_douta         u_in_bram.douta
#define u_in_doutb         u_in_bram.doutb
```

Target: VHDL, EDIF

```
interface DPBRAM (bram_data_t douta, bram_data_t doutb)
     u_in_bram (bram_addr_t addra = sig_u_in_addra,
                bram_addr_t addrb = sig_u_in_addrb,
                bram_data_t dina = sig_u_in_dina,
                bram_data_t dinb = sig_u_in_dinb,
                wire_t wea = sig_u_in_wea,
                wire_t web = sig_u_in_web,
                wire_t clka = __clock,
                wire_t clkb = __clock);
```

Target: SIMULATOR

```
interface DPBRAM (bram_data_t douta with {extfunc = "PlugInGet"},
                  bram_data_t doutb with {extfunc = "PlugInGet"})
     u_in_bram (bram_addr_t addra = sig_u_in_addra,
                bram_addr_t addrb = sig_u_in_addrb,
                bram_data_t dina = sig_u_in_dina,
                bram_data_t dinb = sig_u_in_dinb,
                wire_t wea = sig_u_in_wea,
                wire_t web = sig_u_in_web,
                wire_t clka = __clock with {extfunc = "PlugInClock"},
                wire_t clkb = __clock with {extfunc = "PlugInClock"})
     with {extlib = DPBRAM_PLG,
             extinst = "u_in_bram",
             extfunc = "PlugInSet"};
```