The Abdus Salam
International Centre for Theoretical Physics

United Nations
Educational, Scientific and
Cultural Organization

IAEA
International Atomic Energy Agency

2177-21

ICTP Latin-American Basic Course on FPGA Design for Scientific Instrumentation

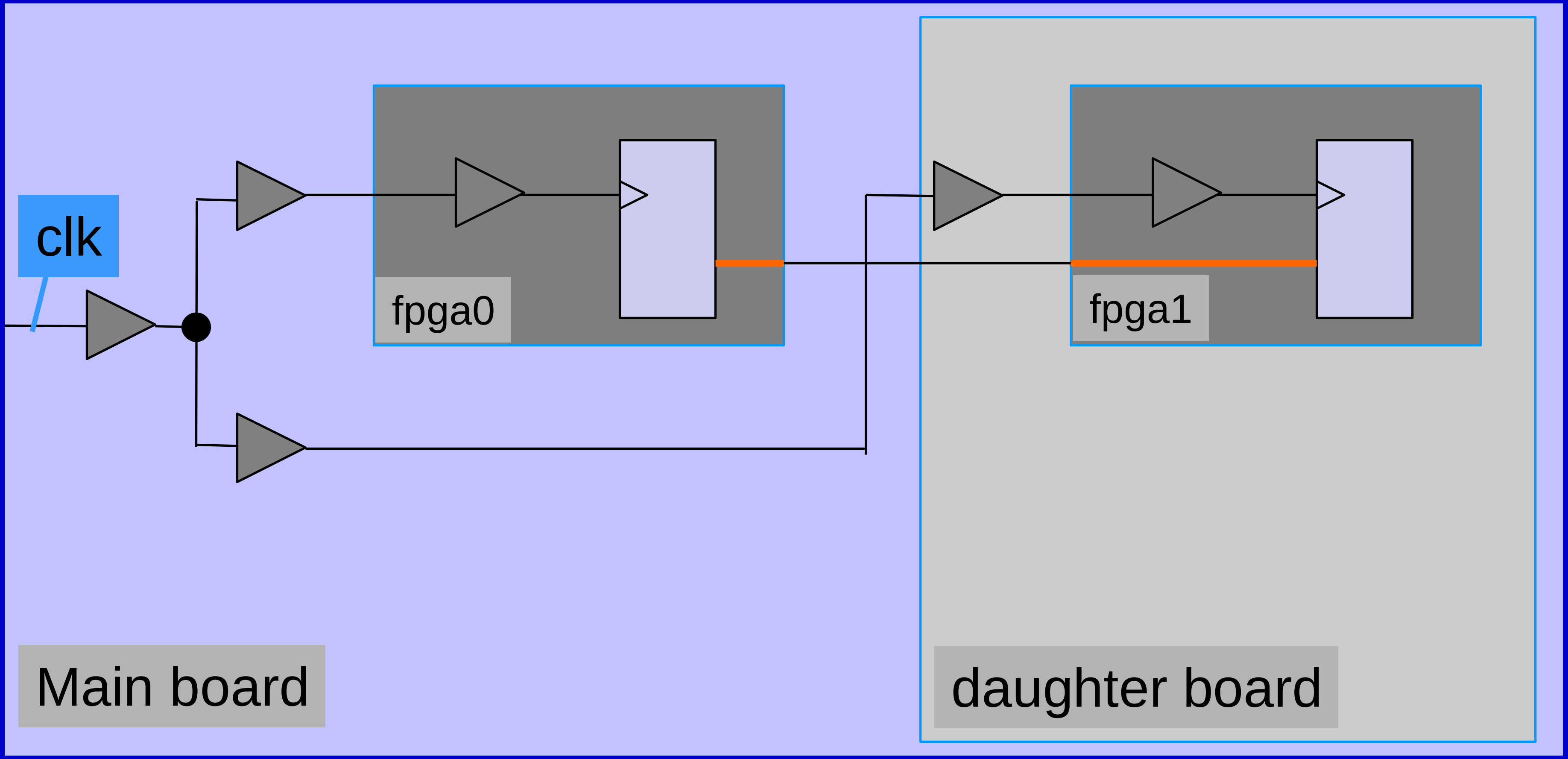
15 - 31 March 2010

Clock domains – multiple FPGA design

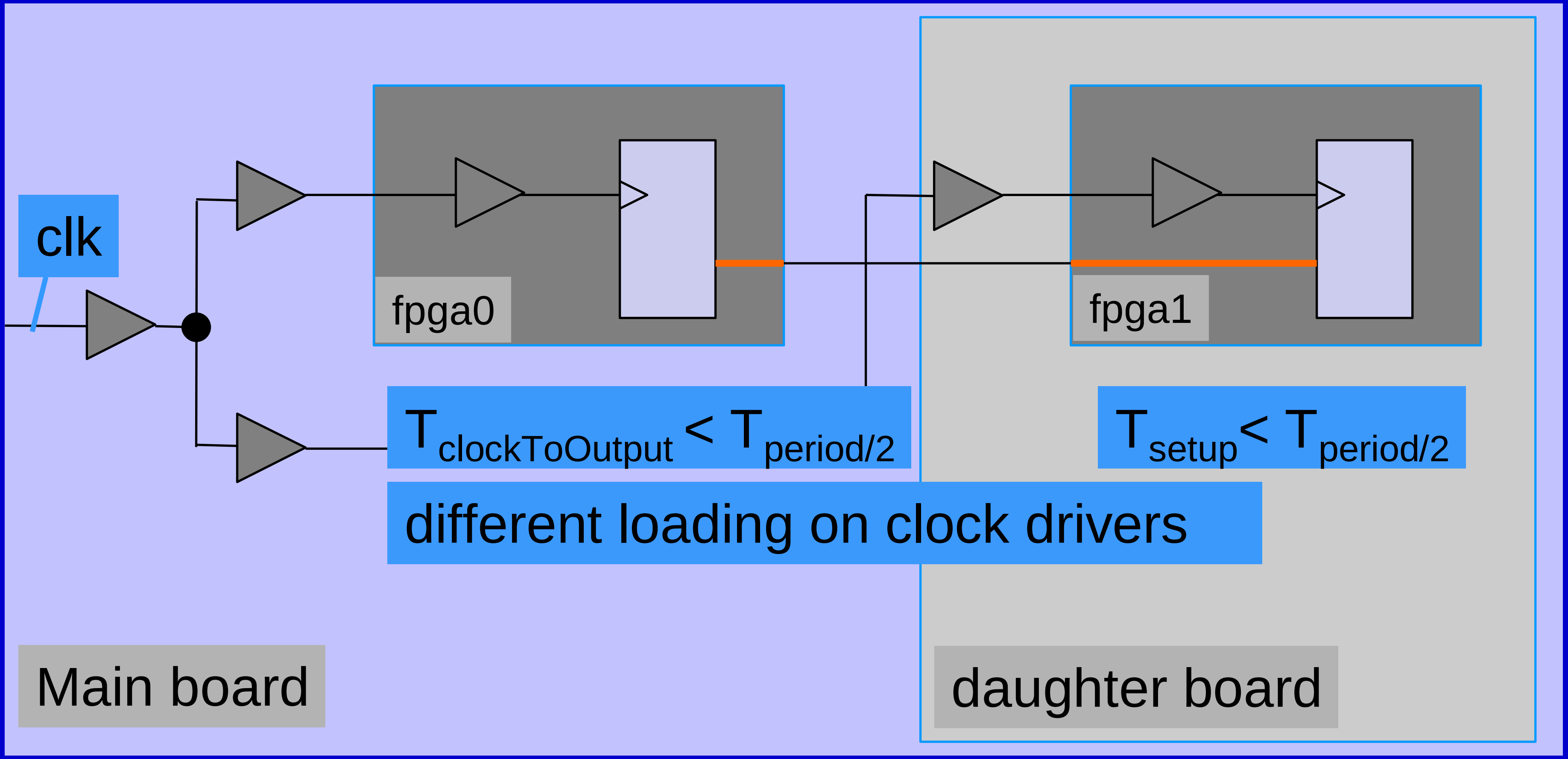
KLUGE Alexander
CERN
Geneva
Switzerland

Clock domains – multiple FPGA design

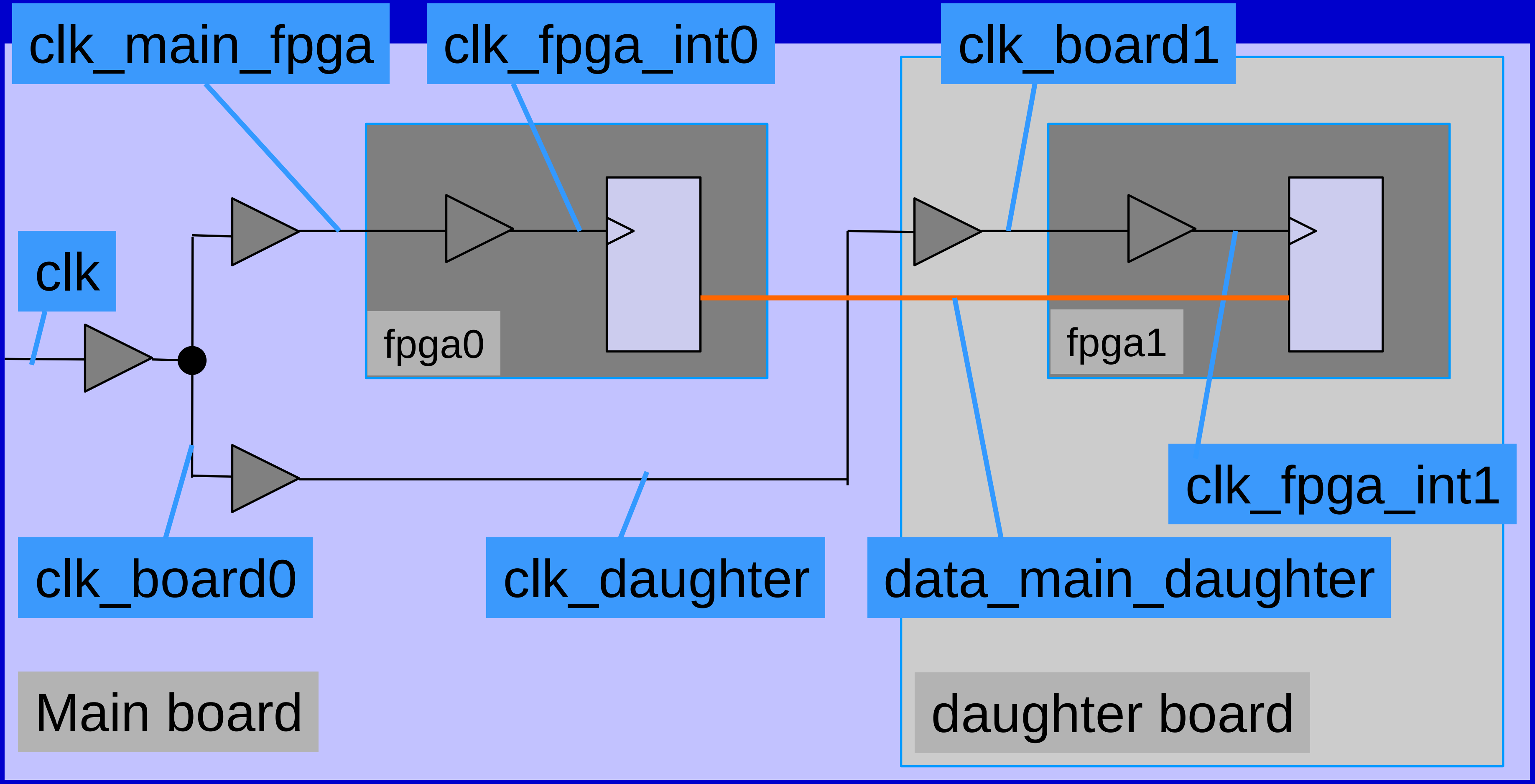
Clock distribution: multiple FPGAs



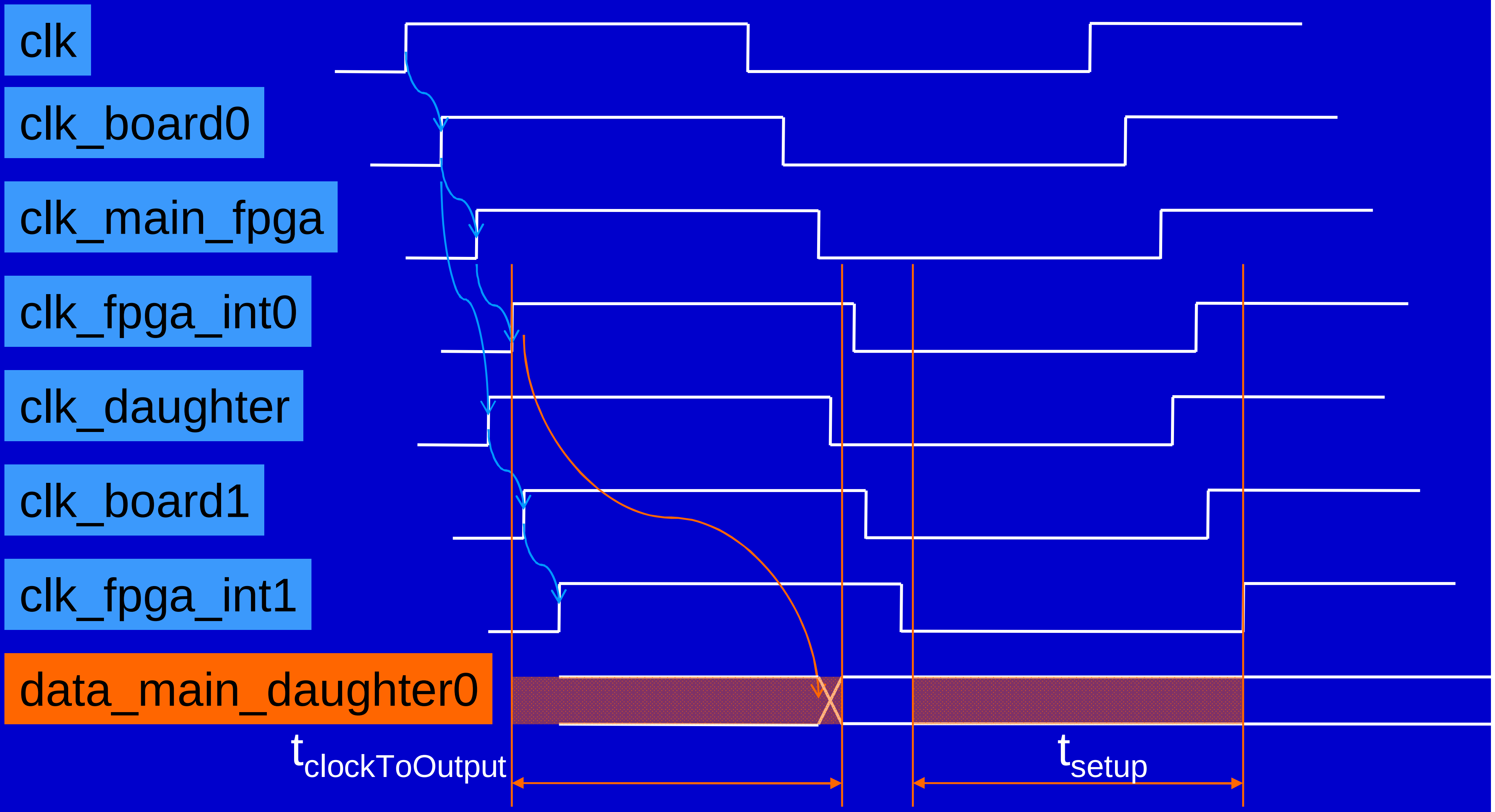
Clock distribution: multiple FPGAs


clk
fpga0
fpga1
$T_{clockToOutput} < T_{period/2}$
$T_{setup} < T_{period/2}$
different loading on clock drivers
Main board
daughter board

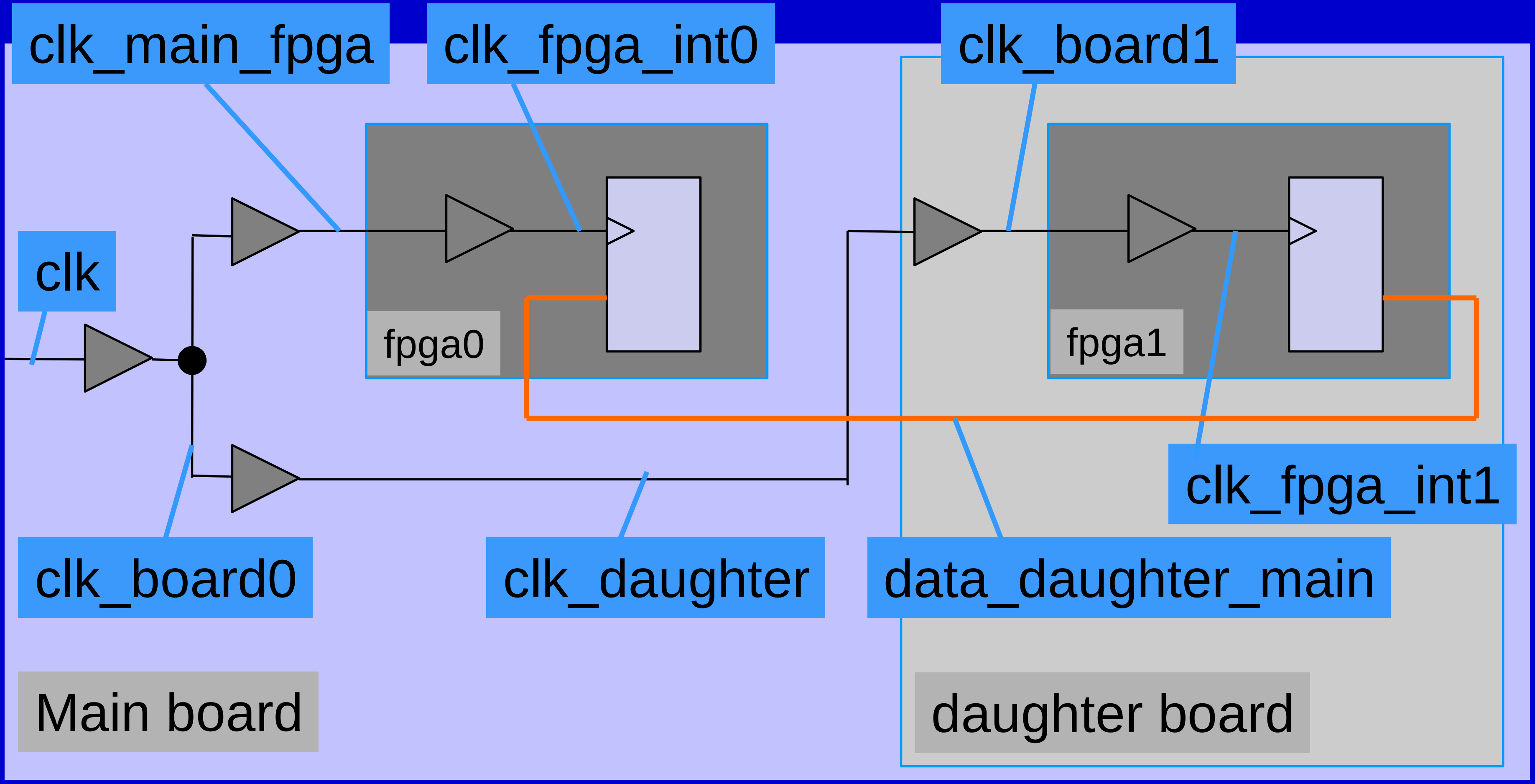
Clock distribution



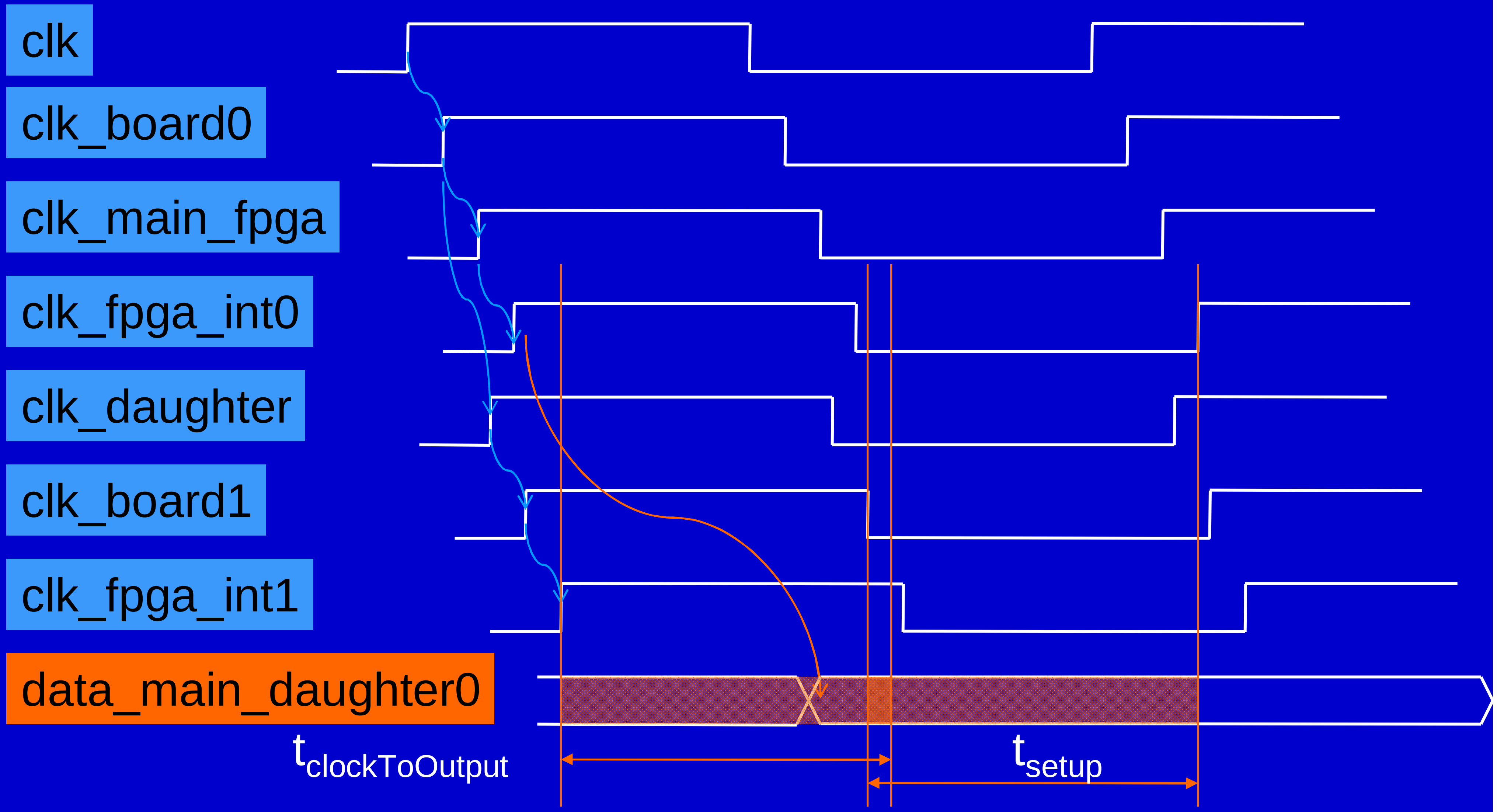
clock distribution/t_{co} & t_s /board 0-> 1



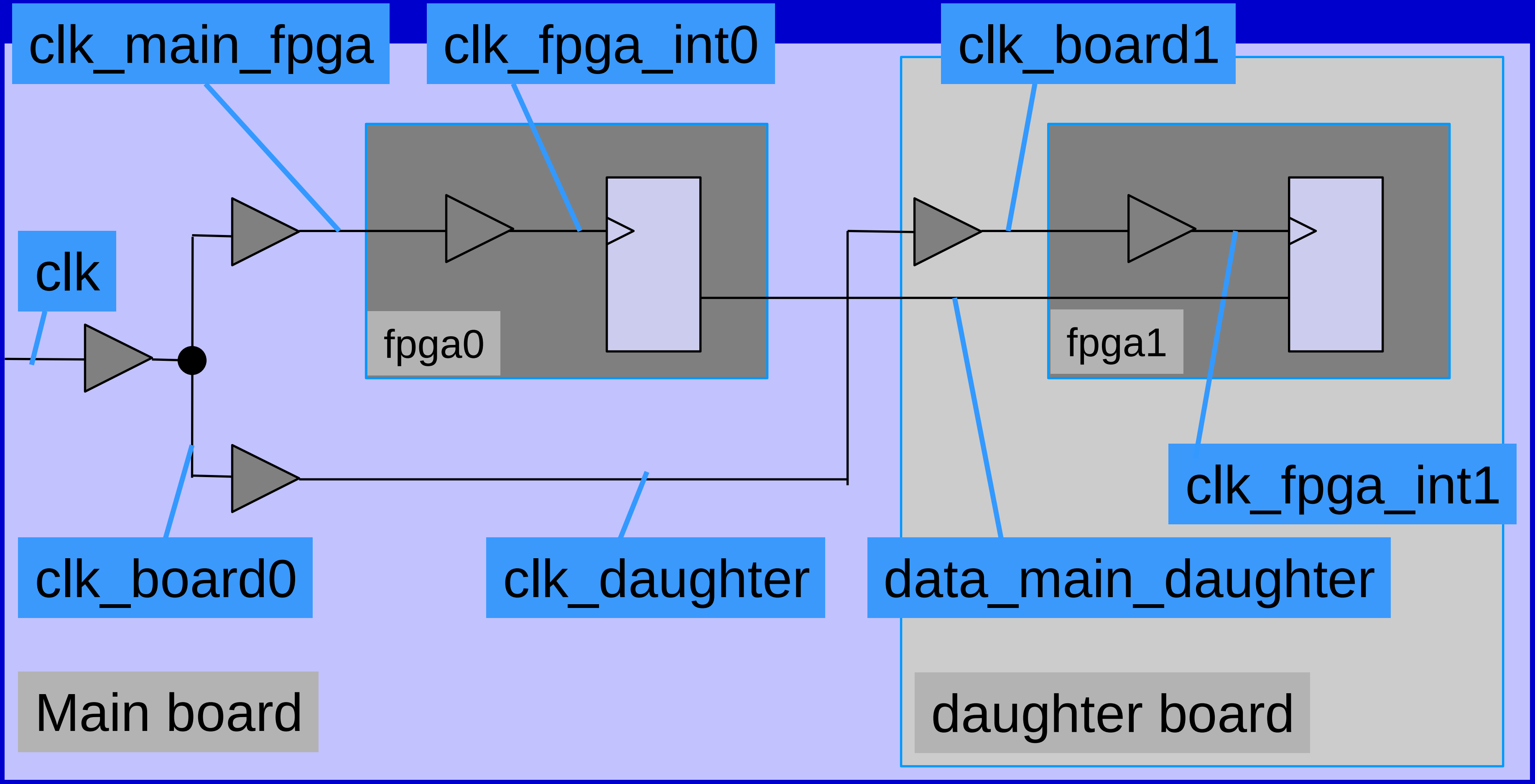
Clock distribution



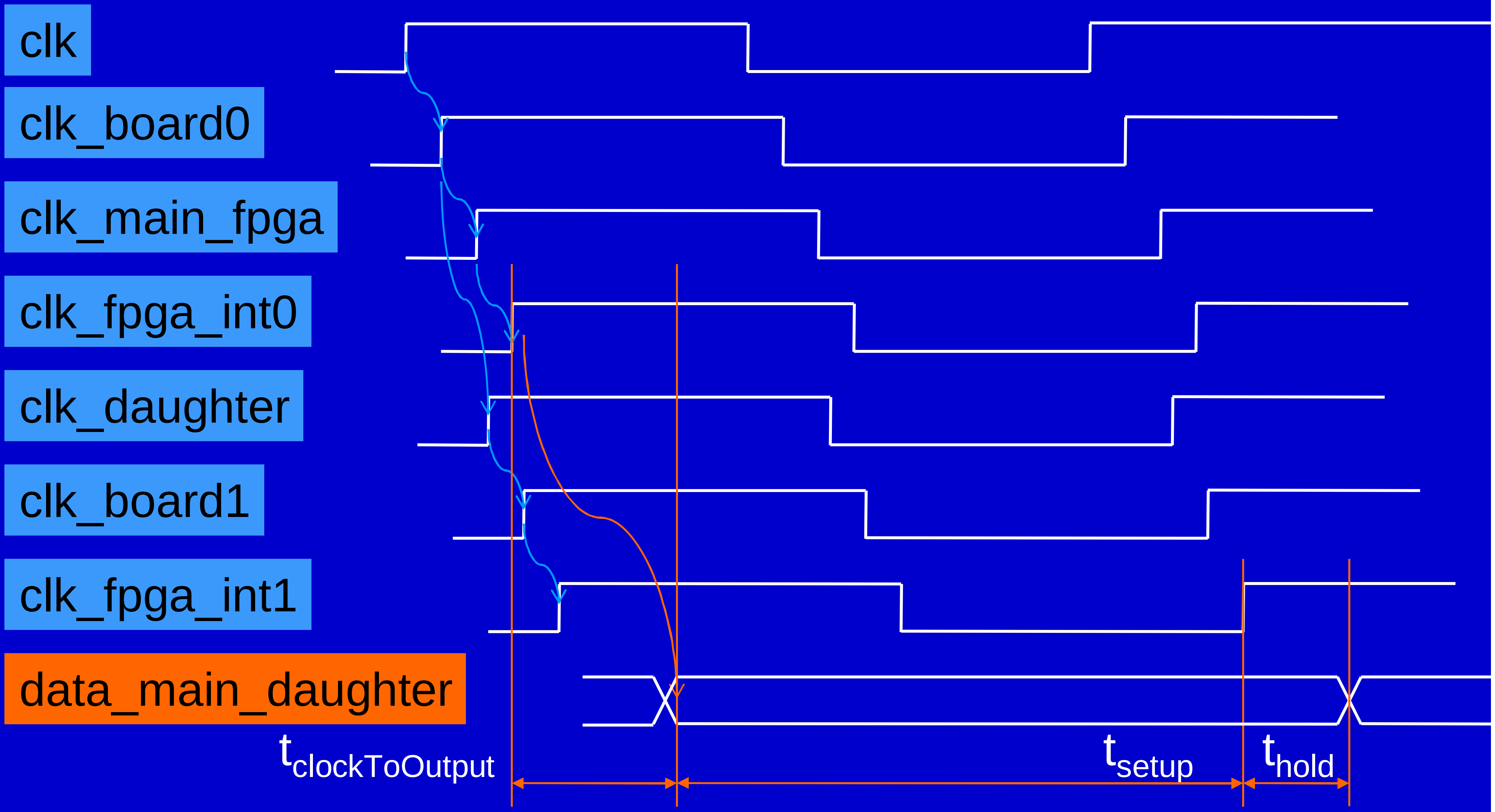
clock distribution/t_{co} & t_s /board 1-> 0



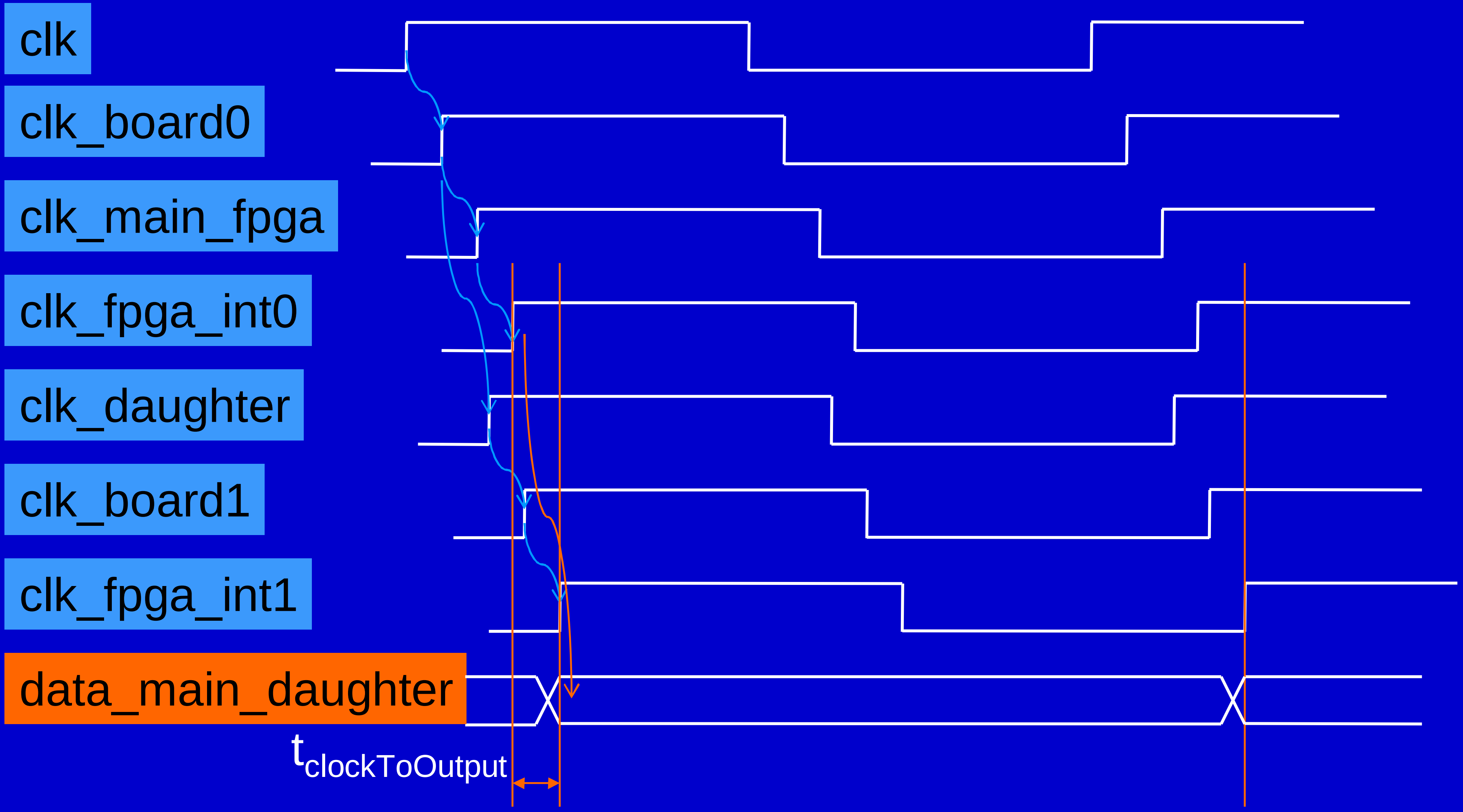
Clock distribution



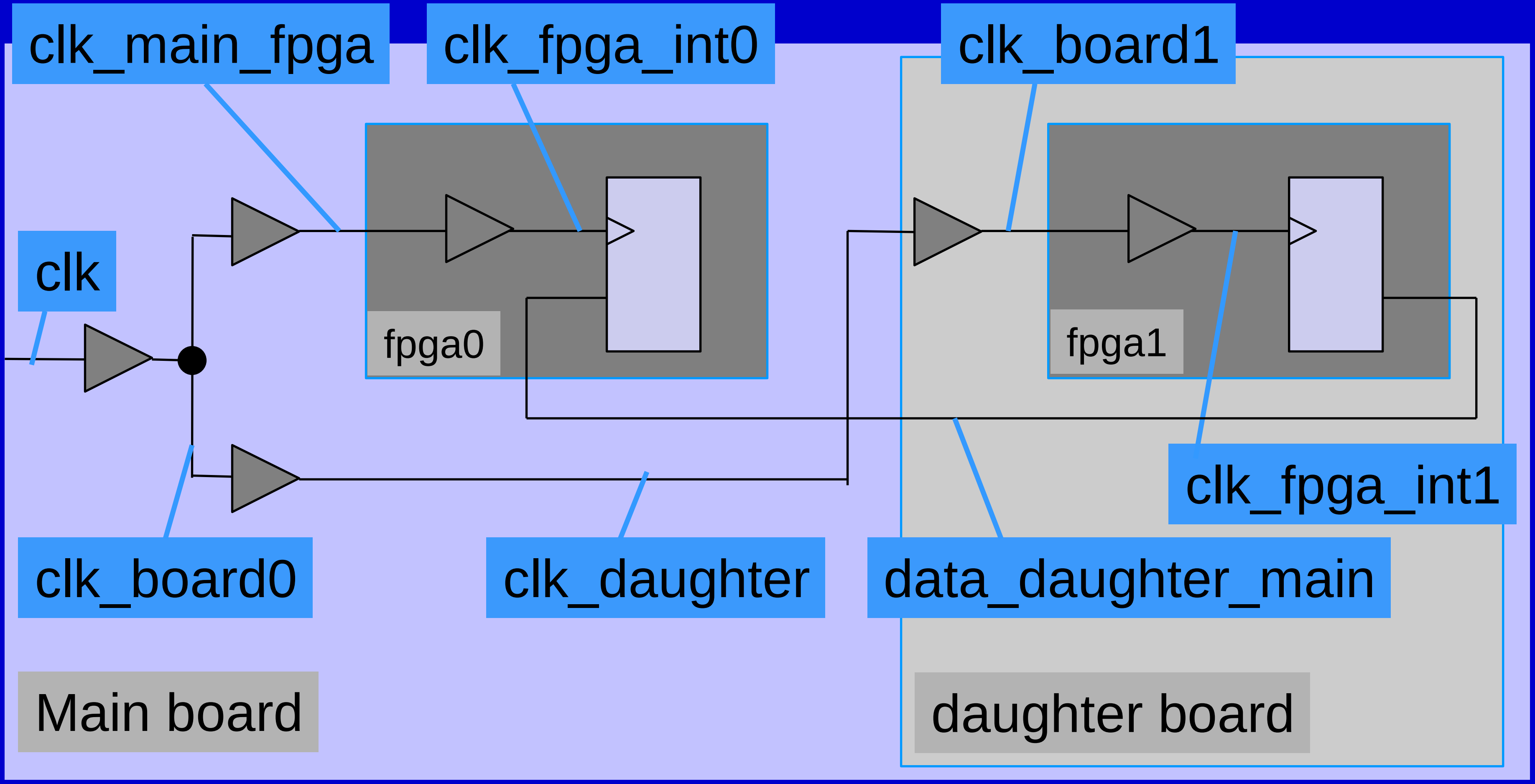
clock distribution/slow output board 0->1



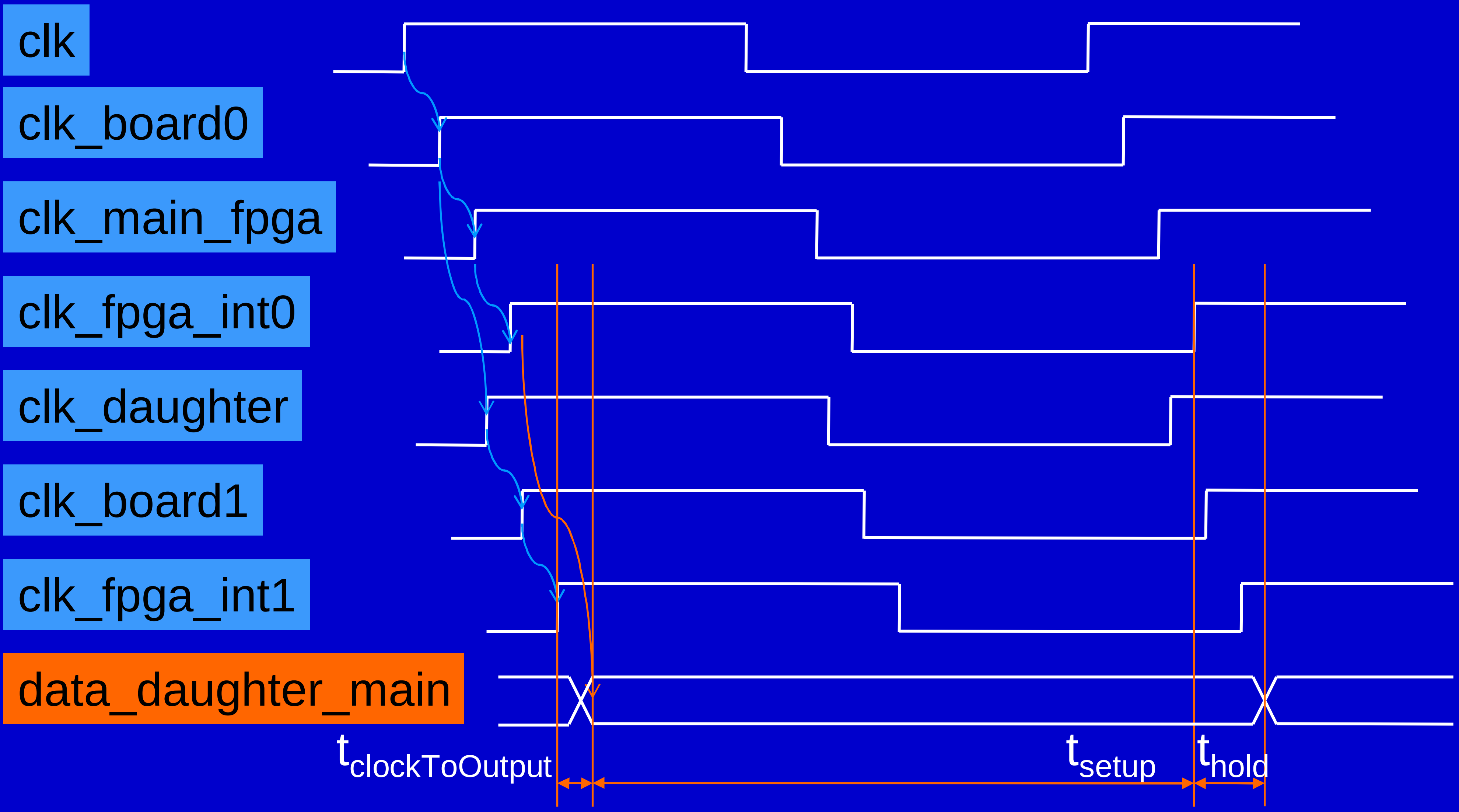
clock distribution/fast output board 0->1



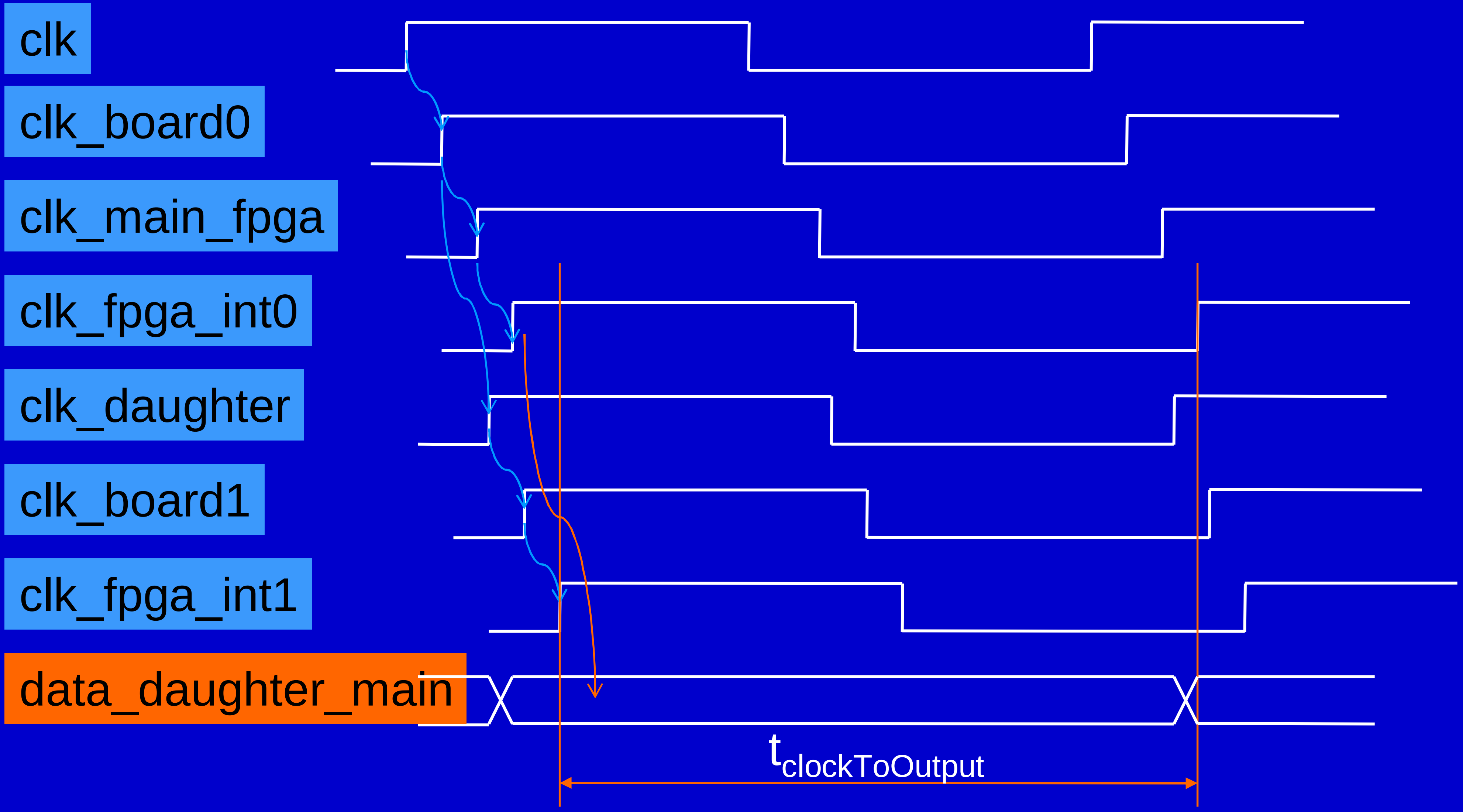
Clock distribution



clock distribution/fast output board 1-> 0



clock distribution/slow output board 1-> 0

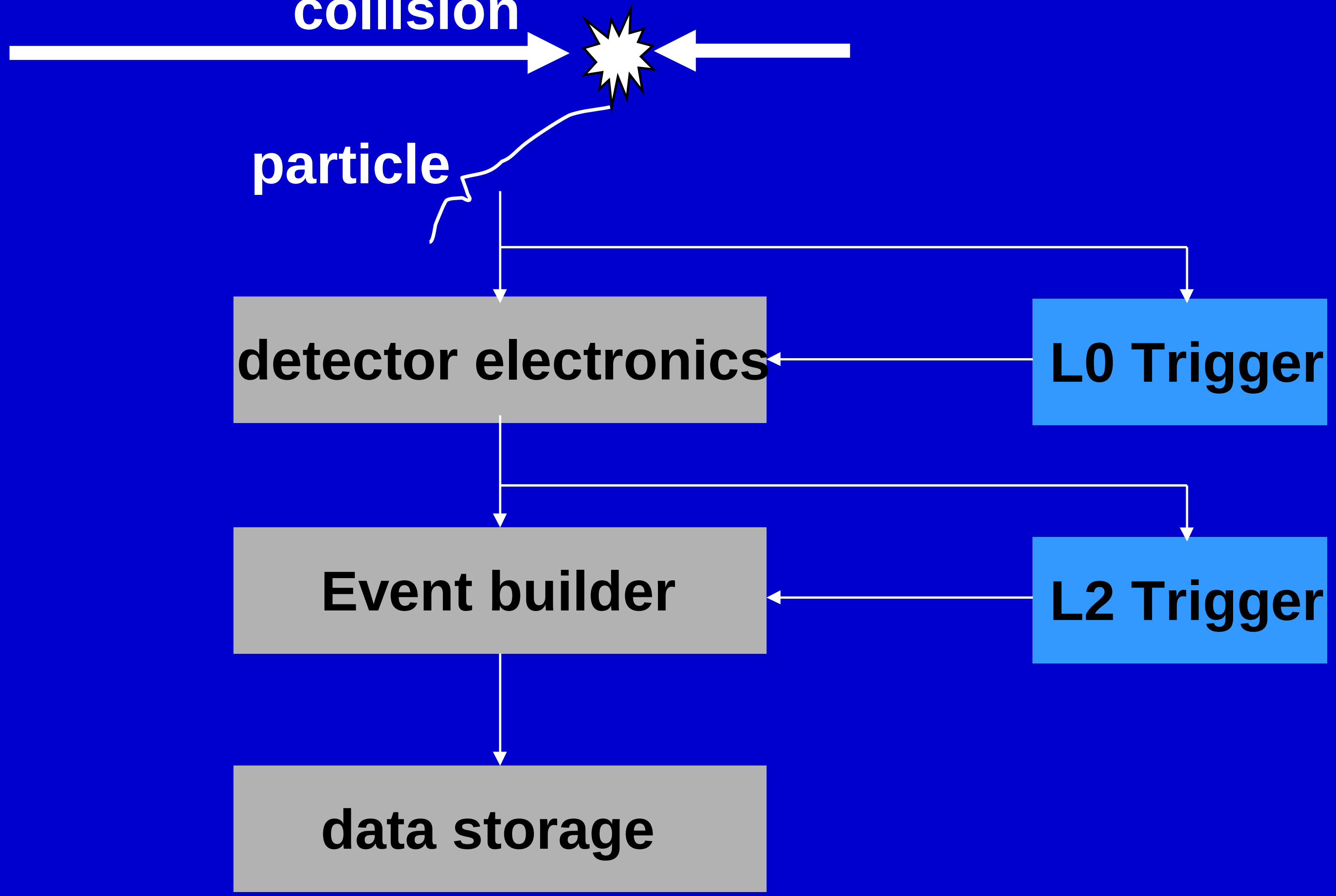


Constraints

- Fulfilling FPGA internal constraints is not sufficient.
- Perform system simulations
- Logic can be too fast

Data selection & delay

Data selection and delay



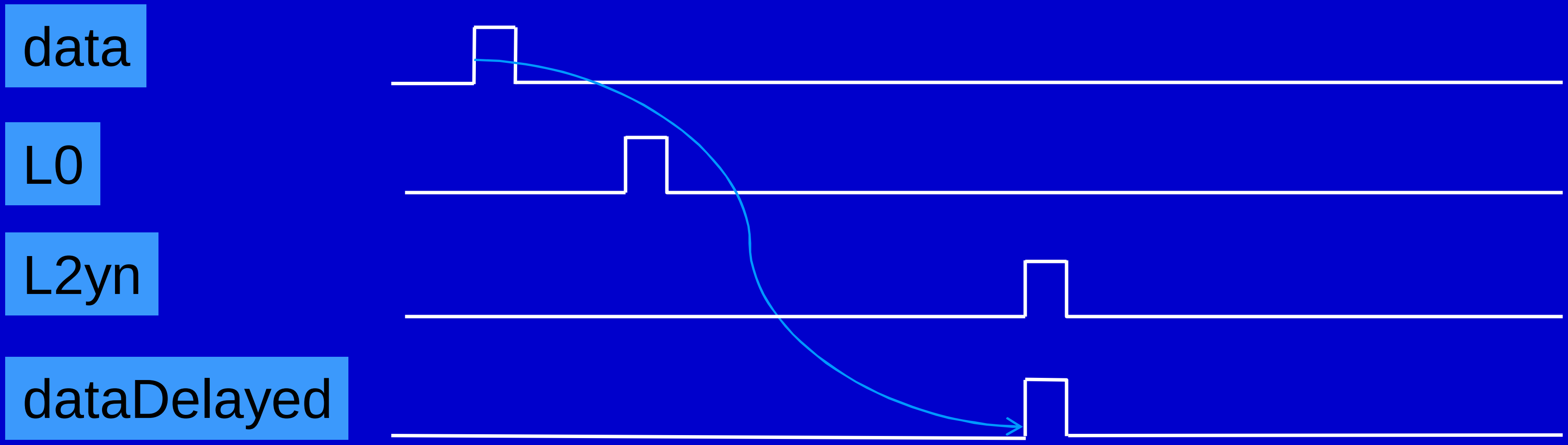
100 Tbyte/s

100 Gbyte/s

100 Mbyte/s

Data selection and delay

- Data (20 bits) every * 100 ns
- collision -> L0 (1µs)
- collision -> L2y or L2n (100 µs)



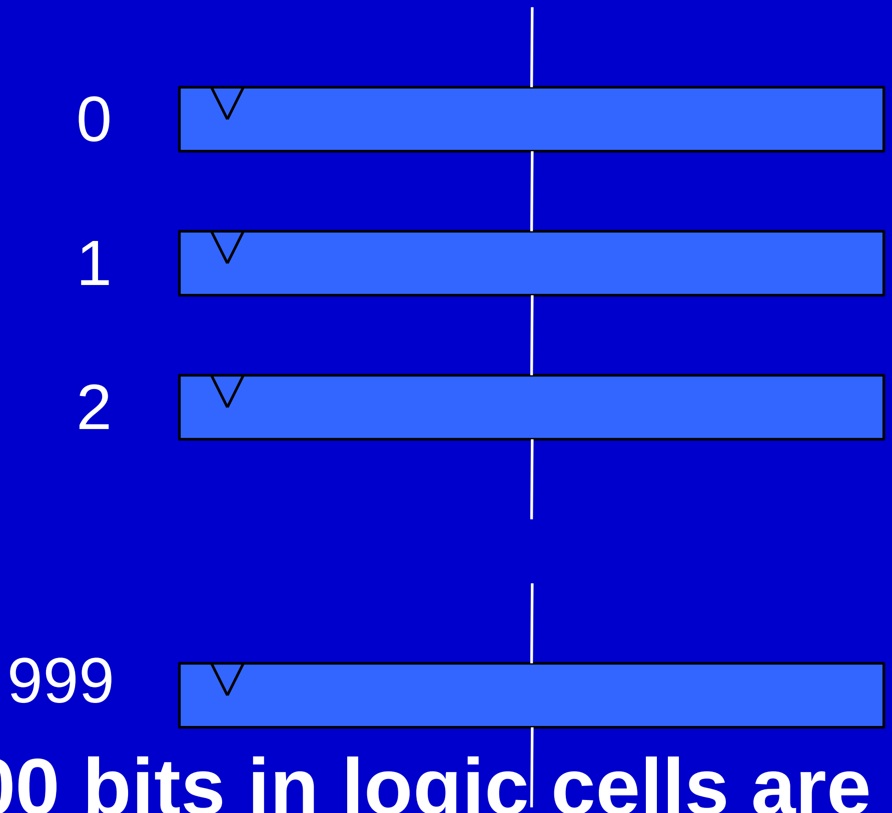
Data selection and delay

- **Data (20 bits) every * 100 ns**
- **collision -> L0 (1µs)**
- **collision -> L2y or L2n (100 µs)**

- **Options:**
  - Data pipeline until L2 with FIFO based on shift registers @ 10 MHz
    20 bits * 100 µs / 100 ns
    20 bits * 1000
    = 20 000 bits

Data selection and delay

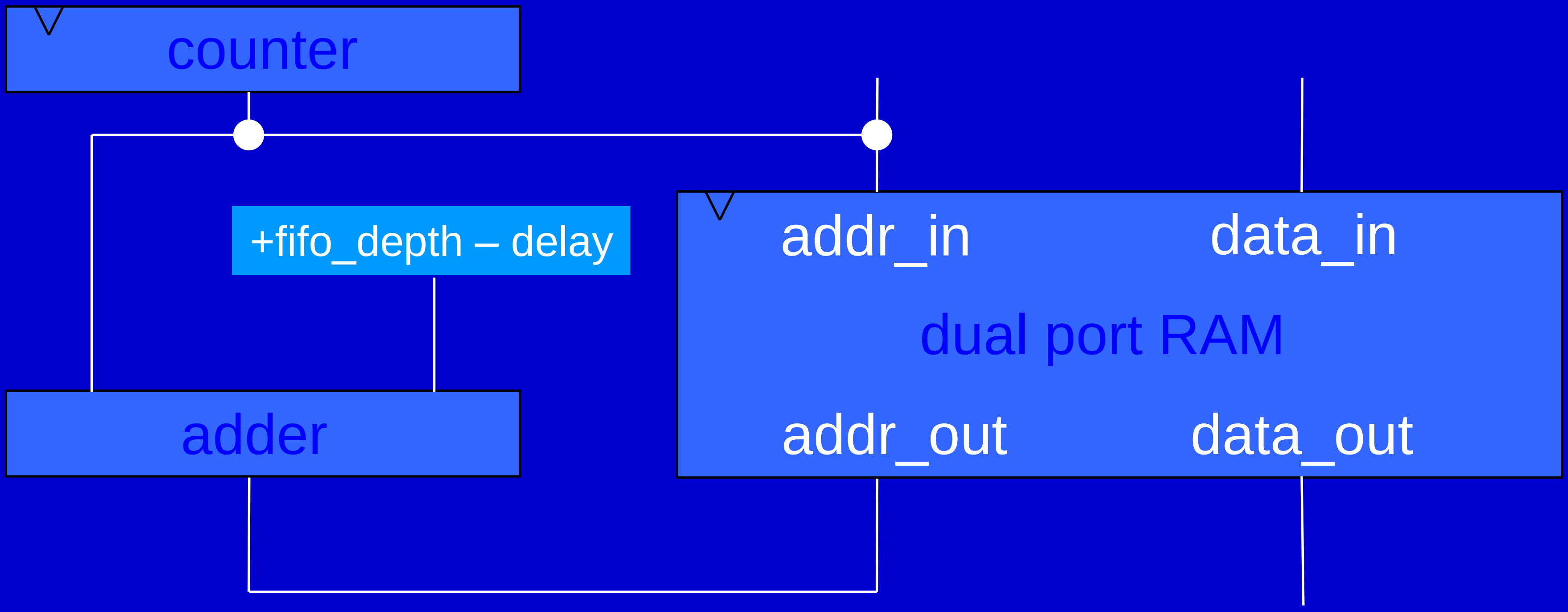
- **Data pipeline with FIFO with shift registers @ 10 MHz**
  20 bits * 1000 = 20 000 bits



20 000 bits in logic cells are used

Data selection and delay

- **Data pipeline with FIFO based on dual port RAM @ 10 MHz**
  20 bits * 1000 = 20 000 bits

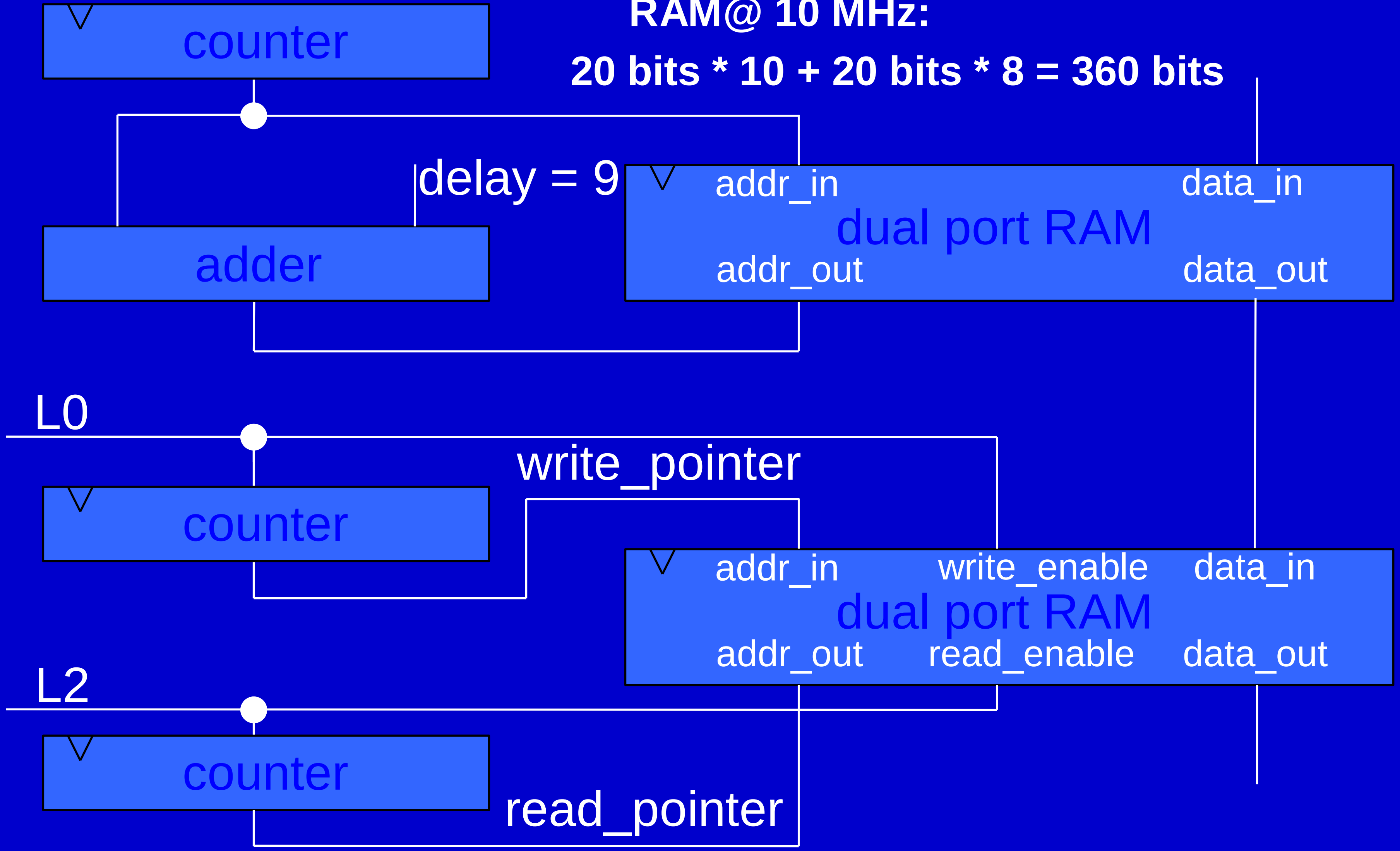


FPGAs have RAM cells in addition to logic blocks

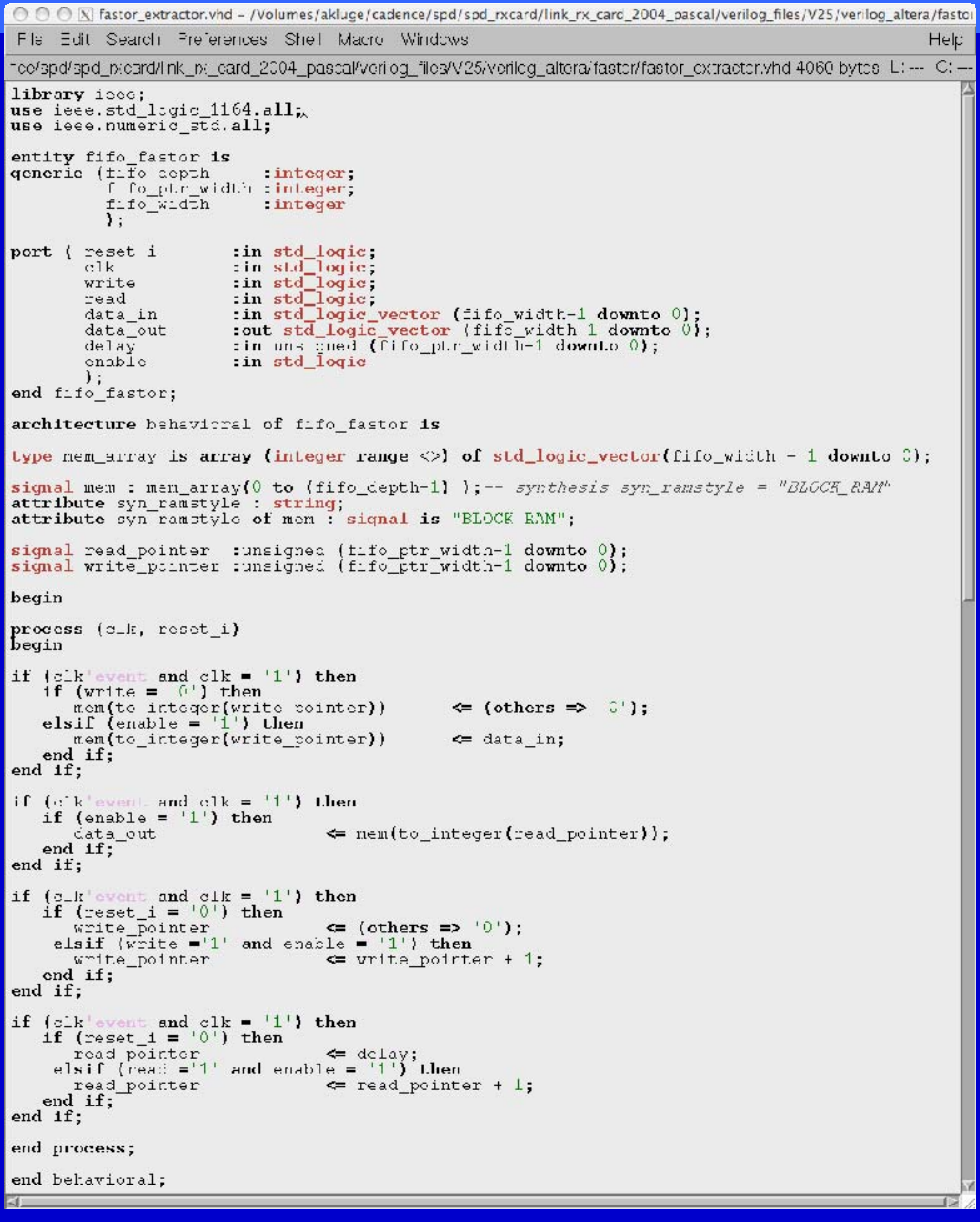
Data selection and delay

Data pipeline with 2 FIFOs based on dual port RAM@ 10 MHz:

20 bits * 10 + 20 bits * 8 = 360 bits



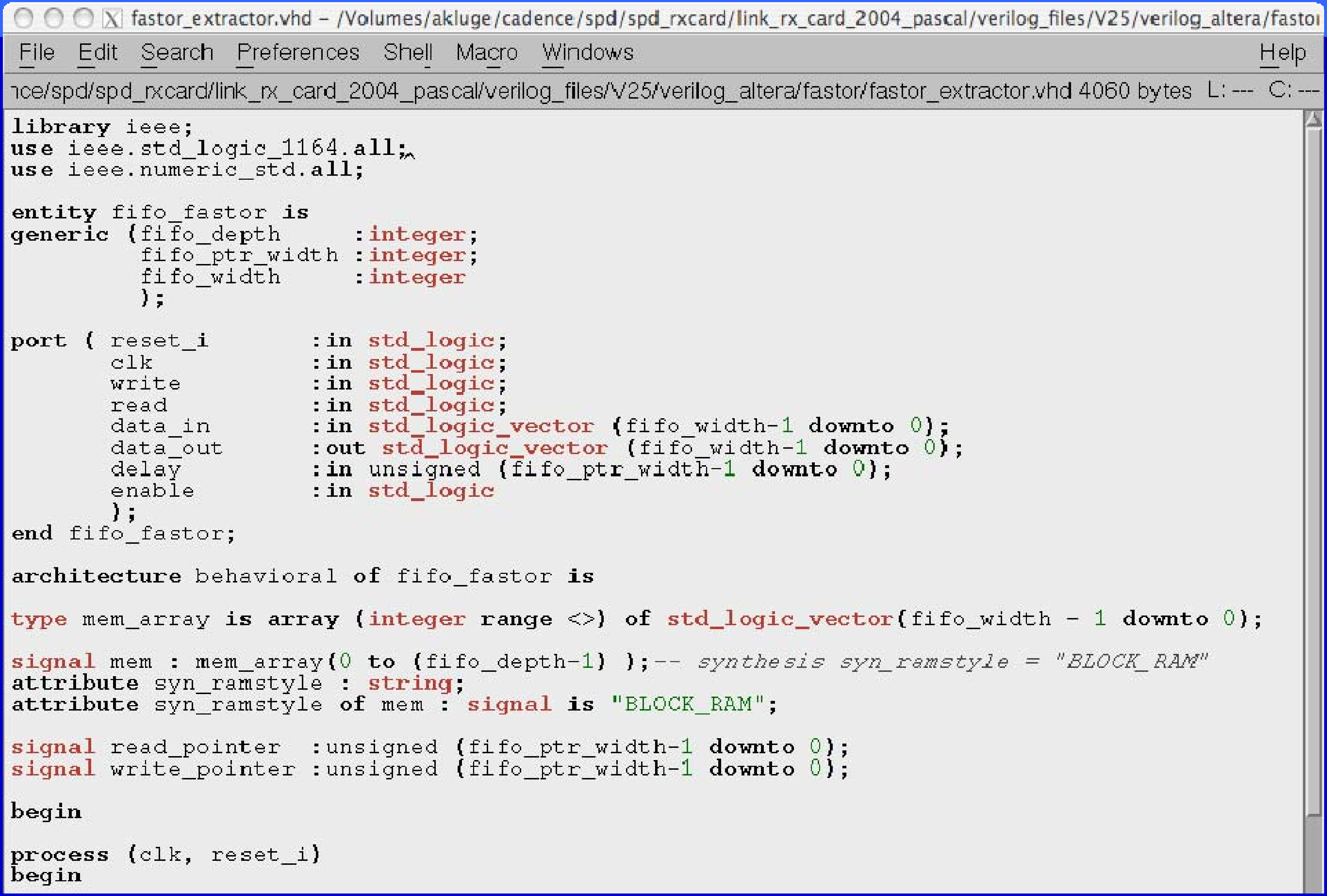
Data selection and delay



```
fastor_extractor.vhd - /Volumes/akluge/cadence/spd/spd_rxcard/link_rx_card_2004_pascal/verilog_files/V25/verilog_altera/fastor
~co/spd/spd_rxcard/link_rx_card_2004_pascal/verilog_files/V25/verilog_altera/fastor/fastor_extractor.vhd 4060 bytes L: --- C: ---
```

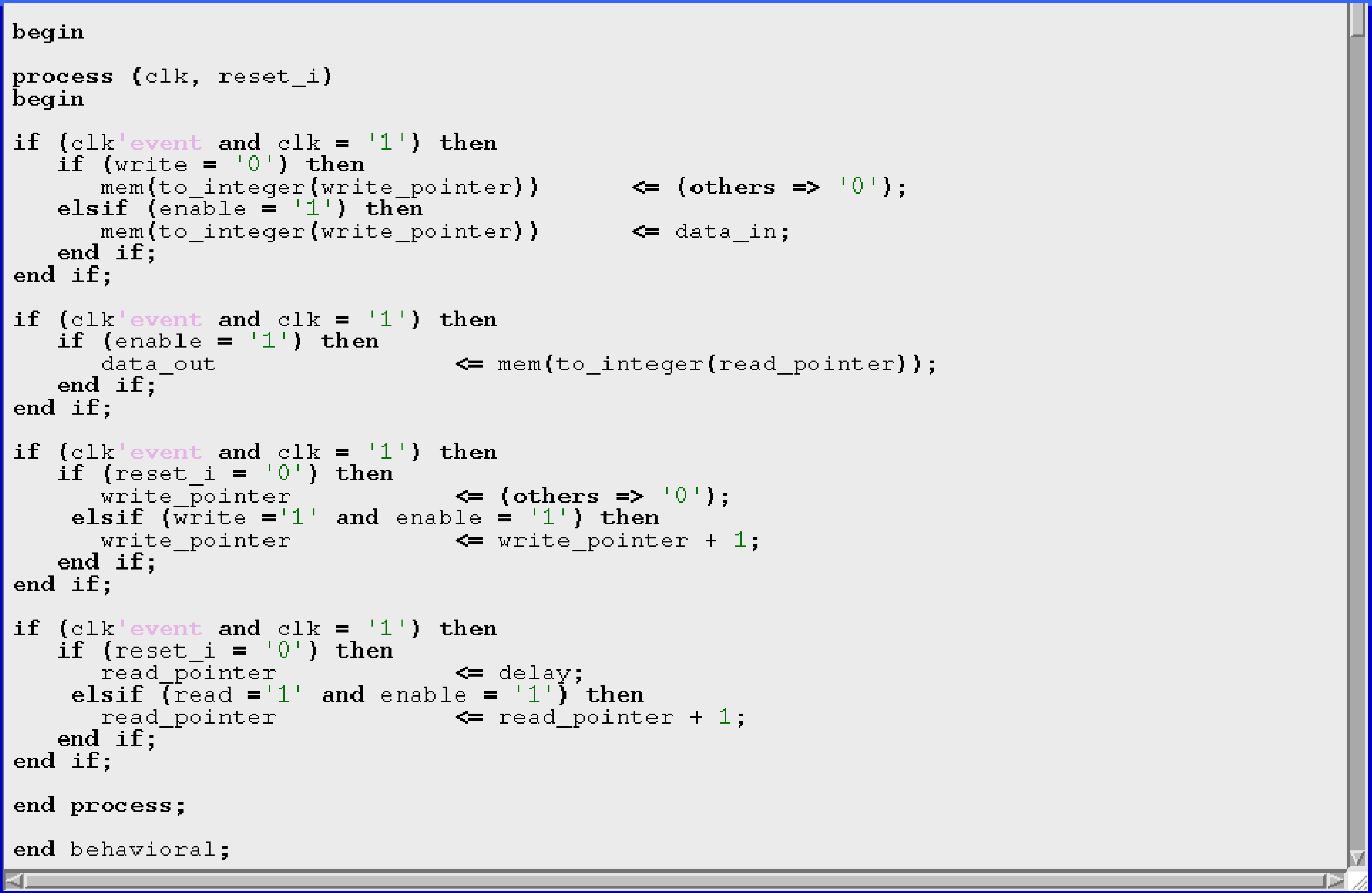
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fifo_fastor is
generic (fifo_depth        :integer;
         fifo_ptr_width    :integer;
         fifo_width        :integer
         );

port ( reset_i          :in std_logic;
       clk              :in std_logic;
       write            :in std_logic;
       read             :in std_logic;
       data_in          :in std_logic_vector (fifo_width-1 downto 0);
       data_out         :out std_logic_vector (fifo_width 1 downto 0);
       delay            :in unsigned (fifo_ptr_width-1 downto 0);
       enable           :in std_logic
       );
end fifo_fastor;

architecture behavioral of fifo_fastor is

type mem_array is array (integer range <>) of std_logic_vector(fifo_width - 1 downto 0);

signal mem : mem_array(0 to (fifo_depth-1) );-- synthesis syn_ramstyle = "BLOCK_RAM"
attribute syn_ramstyle : string;
attribute syn_ramstyle of mem : signal is "BLOCK RAM";

signal read_pointer  :unsigned (fifo_ptr_width-1 downto 0);
signal write_pointer :unsigned (fifo_ptr_width-1 downto 0);

begin

process (clk, reset_i)
begin

if (clk'event and clk = '1') then
   if (write = '0') then
      mem(to_integer(write_pointer))       <= (others =>  '0');
   elsif (enable = '1') then
      mem(to_integer(write_pointer))       <= data_in;
   end if;
end if;

if (clk'event and clk = '1') then
   if (enable = '1') then
      data_out                    <= mem(to_integer(read_pointer));
   end if;
end if;

if (clk'event and clk = '1') then
   if (reset_i = '0') then
      write_pointer               <= (others => '0');
    elsif (write ='1' and enable = '1') then
      write_pointer               <= write_pointer + 1;
   end if;
end if;

if (clk'event and clk = '1') then
   if (reset_i = '0') then
      read pointer                <= delay;
    elsif (read ='1' and enable = '1') then
      read_pointer                <= read_pointer + 1;
   end if;
end if;

end process;

end behavioral;
```

Data selection and delay


fastor_extractor.vhd - /Volumes/akluge/cadence/spd/spd_rxcard/link_rx_card_2004_pascal/verilog_files/V25/verilog_altera/fasto
File Edit Search Preferences Shell Macro Windows Help
nce/spd/spd_rxcard/link_rx_card_2004_pascal/verilog_files/V25/verilog_altera/fastor/fastor_extractor.vhd 4060 bytes L: --- C: ---

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fifo_fastor is
generic (fifo_depth     :integer;
         fifo_ptr_width :integer;
         fifo_width     :integer
         );

port ( reset_i          :in std_logic;
       clk              :in std_logic;
       write            :in std_logic;
       read             :in std_logic;
       data_in          :in std_logic_vector (fifo_width-1 downto 0);
       data_out         :out std_logic_vector (fifo_width-1 downto 0);
       delay            :in unsigned (fifo_ptr_width-1 downto 0);
       enable           :in std_logic
       );
end fifo_fastor;

architecture behavioral of fifo_fastor is

type mem_array is array (integer range <>) of std_logic_vector(fifo_width - 1 downto 0);

signal mem : mem_array(0 to (fifo_depth-1) );-- synthesis syn_ramstyle = "BLOCK_RAM"
attribute syn_ramstyle : string;
attribute syn_ramstyle of mem : signal is "BLOCK_RAM";

signal read_pointer  :unsigned (fifo_ptr_width-1 downto 0);
signal write_pointer :unsigned (fifo_ptr_width-1 downto 0);

begin

process (clk, reset_i)
begin
```

Data selection and delay



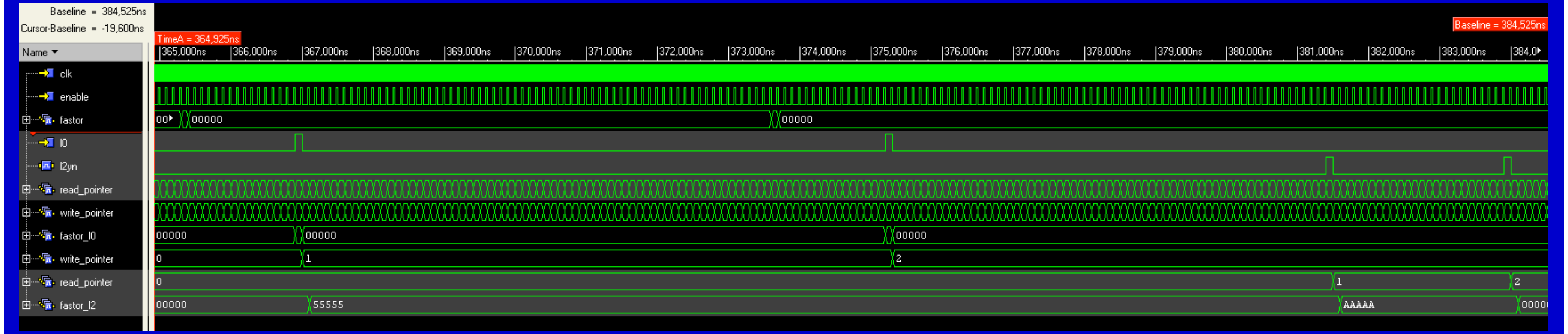
```
begin

process (clk, reset_i)
begin

if (clk'event and clk = '1') then
   if (write = '0') then
      mem(to_integer(write_pointer))      <= (others => '0');
   elsif (enable = '1') then
      mem(to_integer(write_pointer))      <= data_in;
   end if;
end if;

if (clk'event and clk = '1') then
   if (enable = '1') then
      data_out                      <= mem(to_integer(read_pointer));
   end if;
end if;

if (clk'event and clk = '1') then
   if (reset_i = '0') then
      write_pointer                 <= (others => '0');
    elsif (write ='1' and enable = '1') then
      write_pointer                 <= write_pointer + 1;
   end if;
end if;

if (clk'event and clk = '1') then
   if (reset_i = '0') then
      read_pointer                  <= delay;
    elsif (read ='1' and enable = '1') then
      read_pointer                  <= read_pointer + 1;
   end if;
end if;

end process;

end behavioral;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fastor_extractor is
generic (fifo_depth       :integer := 16;
         fifo_ptr_width   :integer := 4;
         fifo_width       :integer := 20
         );

port ( reset_i            :in std_logic;
       clk                :in std_logic;
       fastor0            :in std_logic_vector (9 downto 0);
       fastor1            :in std_logic_vector (9 downto 0);
       l0                 :in std_logic   :='0';
       l2y                :in std_logic   :='0';
       l2n                :in std_logic   :='0';
       delay_l0           :in unsigned (3 downto 0) := "1111";
       fastor_delayed0    :out std_logic_vector (9 downto 0);
       fastor_delayed1    :out std_logic_vector (9 downto 0);
       enable             :in std_logic
       );
end fastor_extractor;

architecture behavioral of fastor_extractor is

component fifo_fastor is
generic (fifo_depth       :integer;
         fifo_ptr_width   :integer;
         fifo_width       :integer
         );
port ( reset_i            :in std_logic;
       clk                :in std_logic;
       write              :in std_logic;
       read               :in std_logic;
       data_in            :in std_logic_vector (fifo_width-1 downto 0);
       data_out           :out std_logic_vector (fifo_width-1 downto 0);
       delay              :in unsigned (fifo_ptr_width-1 downto 0);
       enable             :in std_logic
       );
end component;

signal fastor             :std_logic_vector (fifo_width-1 downto 0);
signal fastor_l0          :std_logic_vector (fifo_width-1 downto 0);
signal fastor_l2          :std_logic_vector (fifo_width-1 downto 0);
signal l2yn               :std_logic;
```

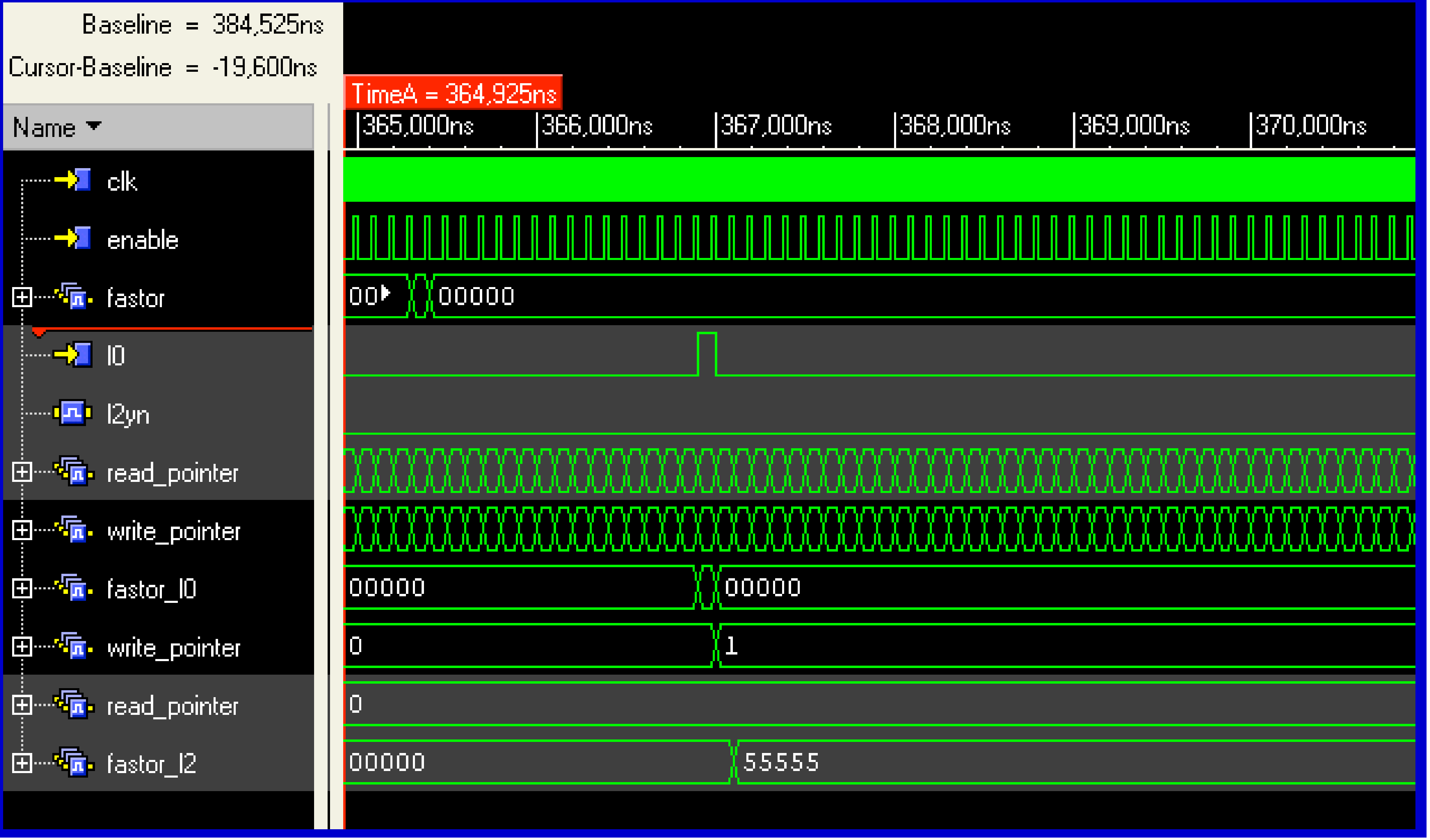
Data selection and delay

```vhdl
begin

fastor (19 downto 10)   <= fastor1;
fastor (9 downto 0)     <= fastor0;
l2yn                    <= l2y or l2n;
fastor_delayed1         <= fastor_l2(19 downto 10);
fastor_delayed0         <= fastor_l2(9 downto 0);

fifo_fastor_l0:   fifo_fastor generic map(fifo_depth,fifo_ptr_width,fifo_width)
                              port map (reset_i,
                                       clk      => clk,
                                       write    => '1',
                                       read     => '1',
                                       data_in  => fastor,
                                       data_out => fastor_l0,
                                       delay    => delay_l0,
                                       enable   => enable
                                       );

fifo_fastor_l2:   fifo_fastor generic map(4,2,20)
                              port map (reset_i,
                                       clk      => clk,
                                       write    => l0,
                                       read     => l2yn,
                                       data_in  => fastor_l0,
                                       data_out => fastor_l2,
                                       delay    => (others => '0'),
                                       enable   => enable
                                       );
end behavioral;
```

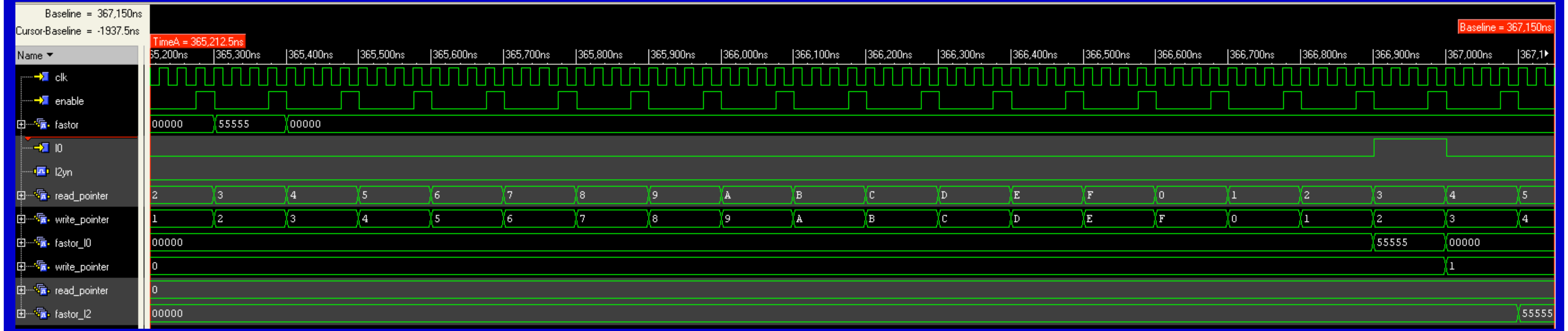
Data selection and delay



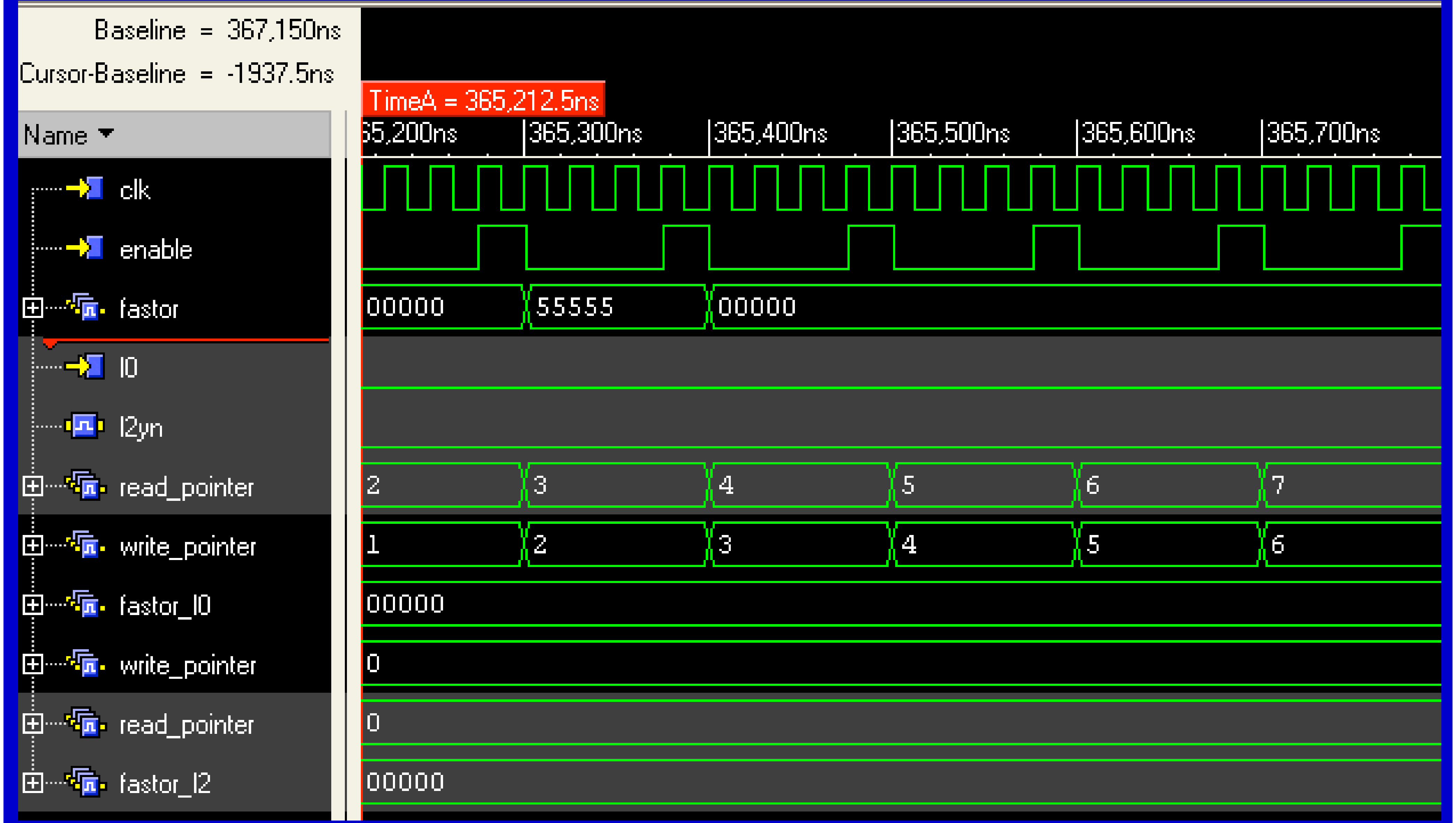
Data selection and delay


Baseline = 384,525ns
Cursor-Baseline = -19,600ns
TimeA = 364,925ns

Data selection and delay



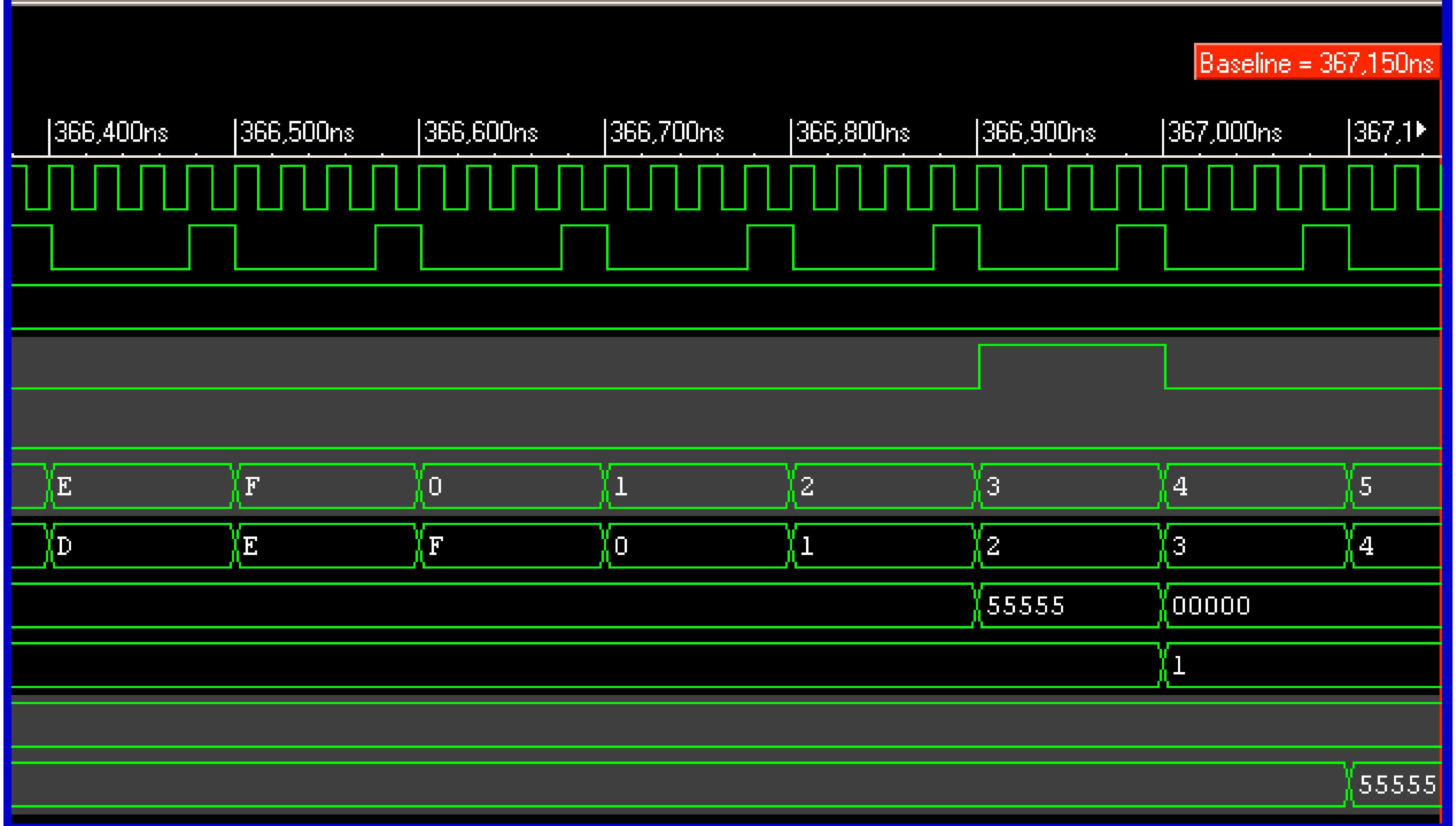
Data selection and delay



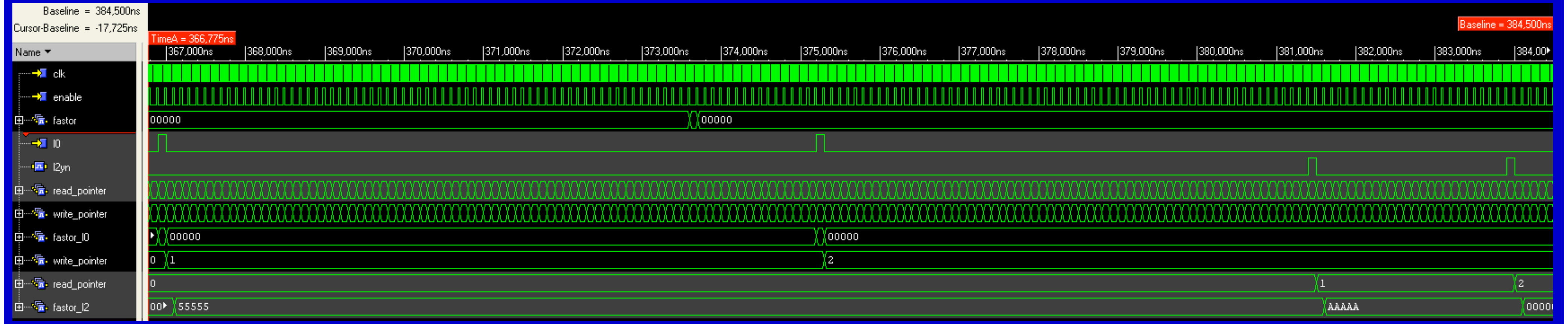
Baseline = 367,150ns
Cursor-Baseline = -1937.5ns
TimeA = 365,212.5ns

```
Name            55,200ns   |365,300ns  |365,400ns  |365,500ns  |365,600ns  |365,700ns
clk
enable
fastor          00000      55555       00000
I0
I2yn
read_pointer    2          3           4           5           6           7
write_pointer   1          2           3           4           5           6
fastor_I0       00000
write_pointer   0
read_pointer    0
fastor_I2       00000
```

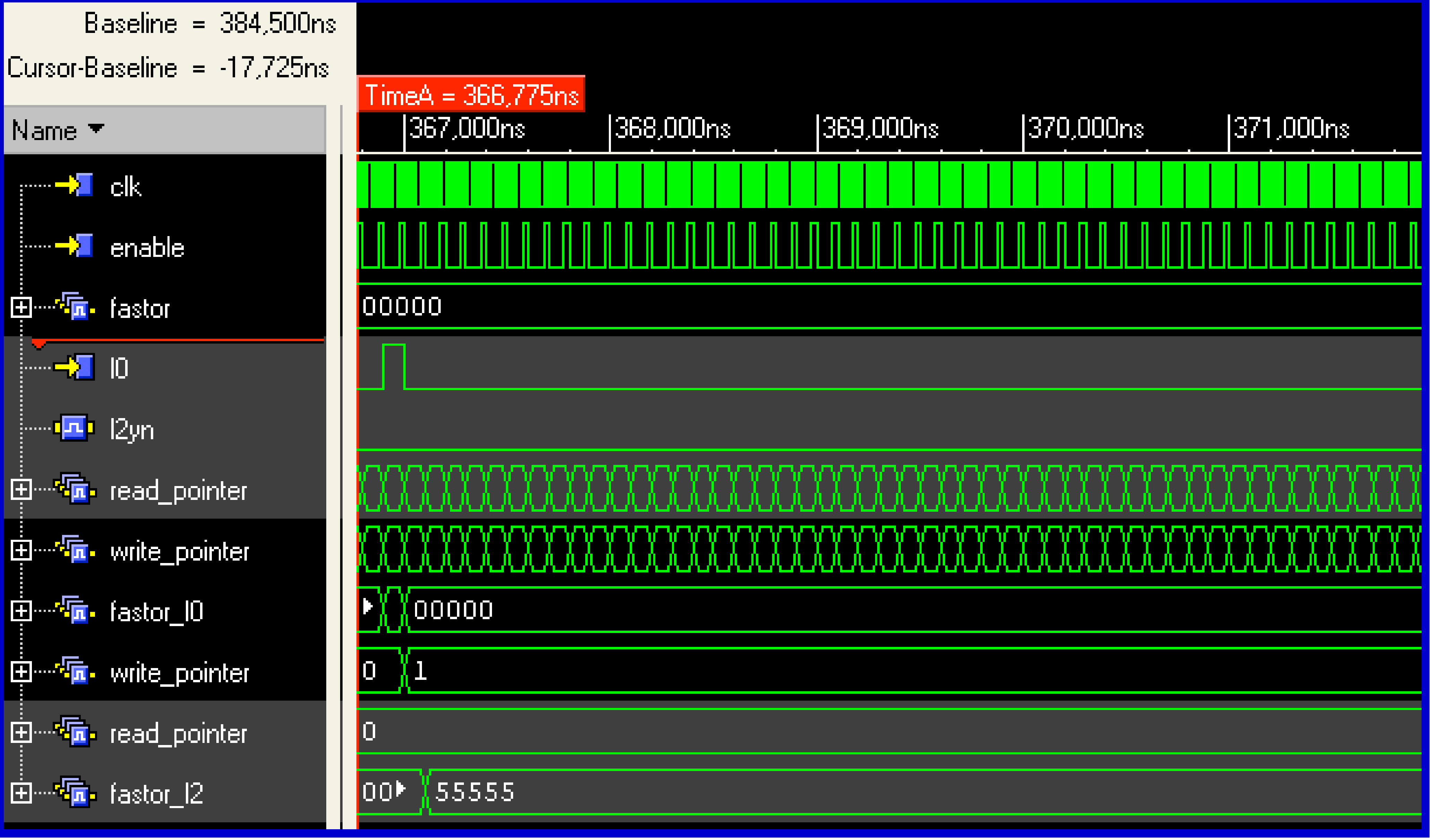
Data selection and delay



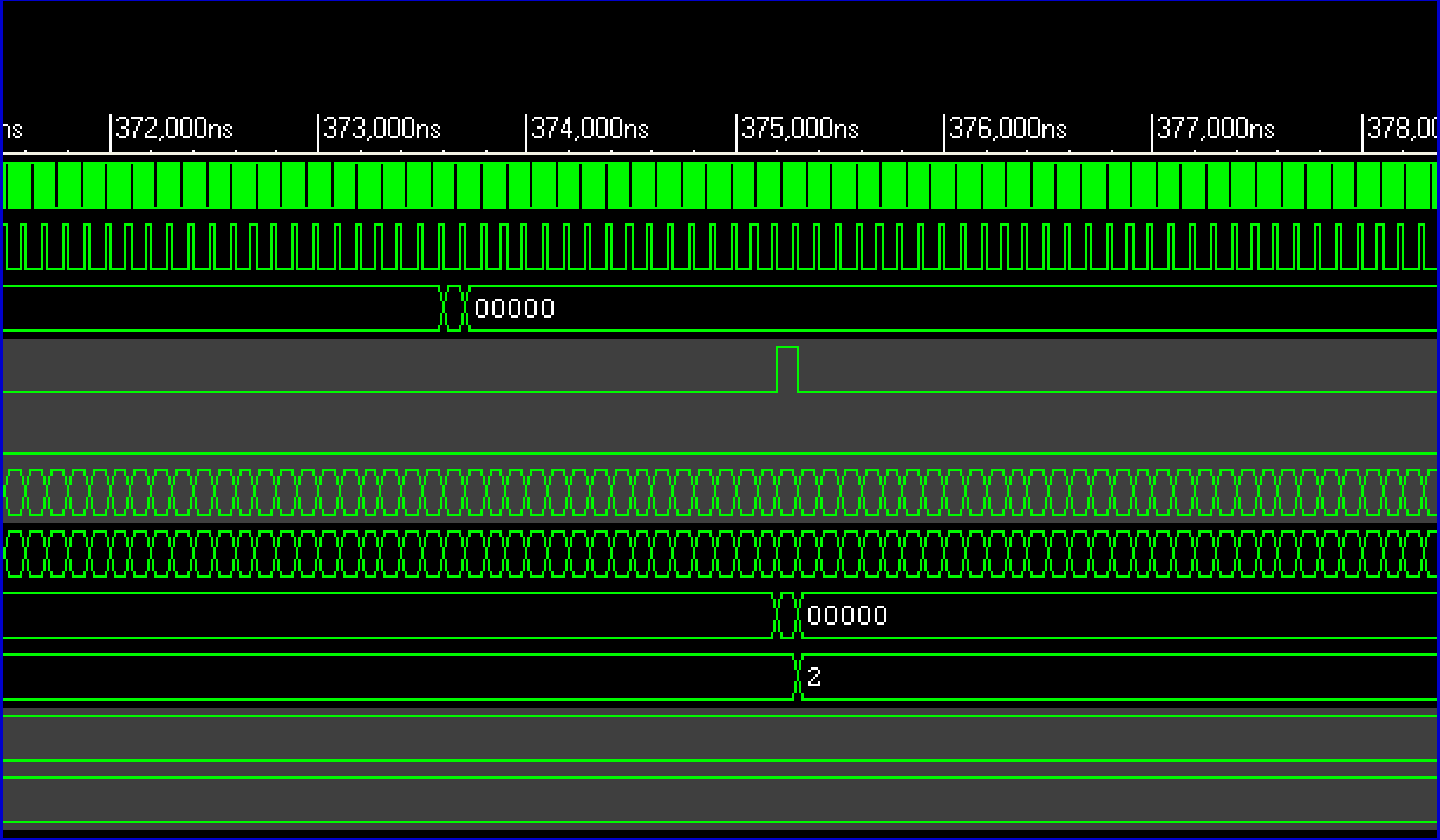
Data selection and delay



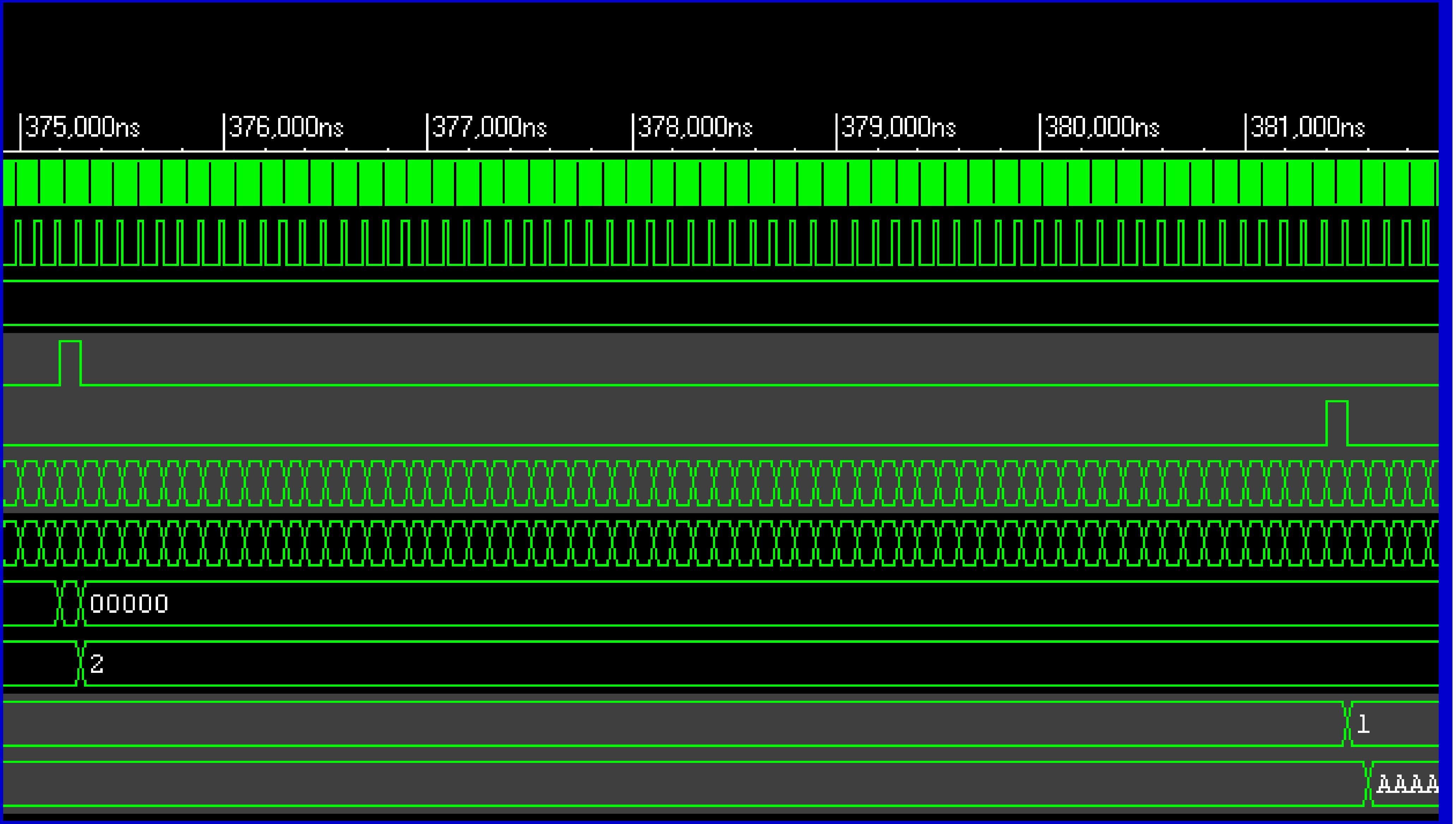
Data selection and delay



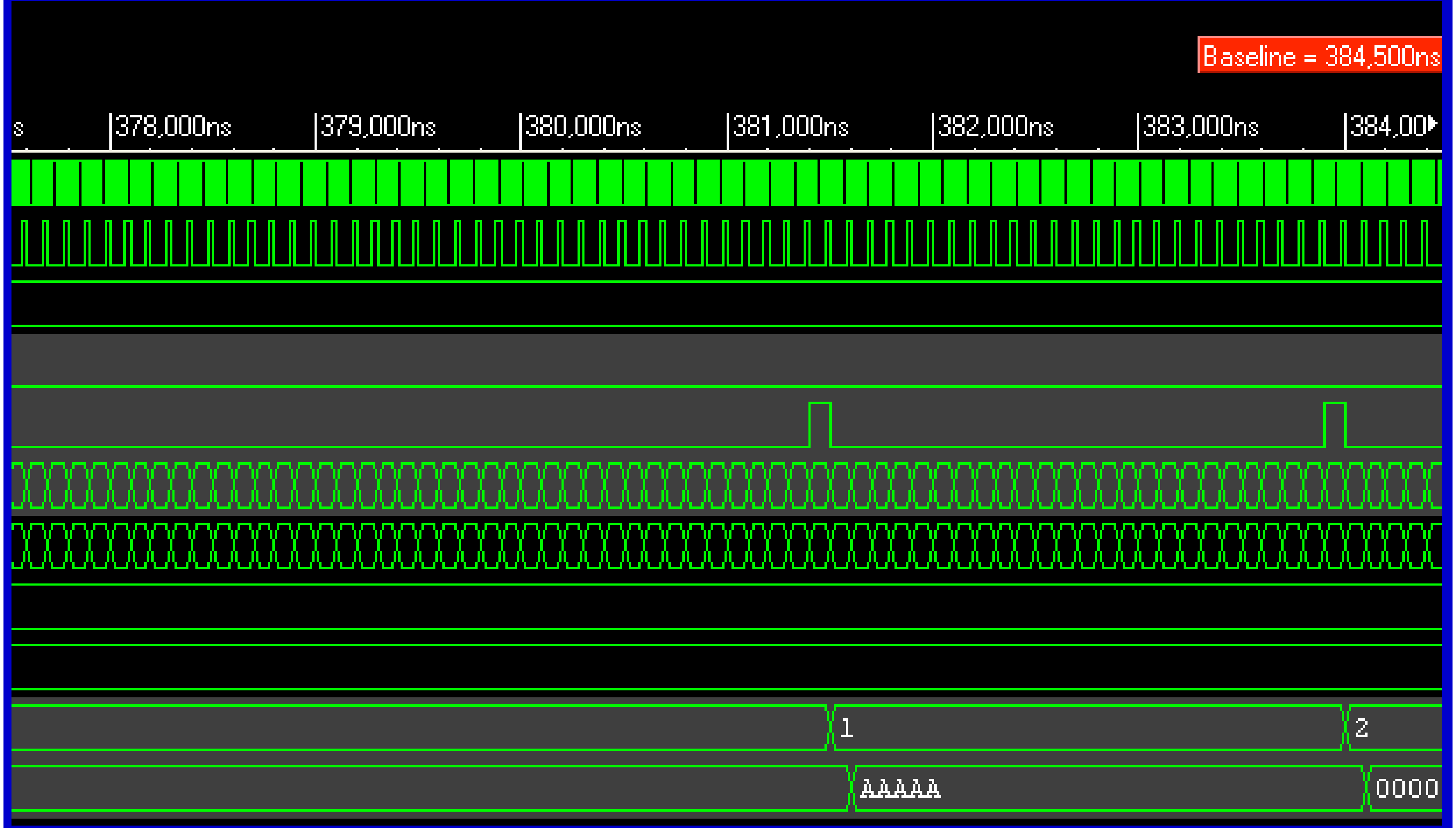
Data selection and delay



Data selection and delay

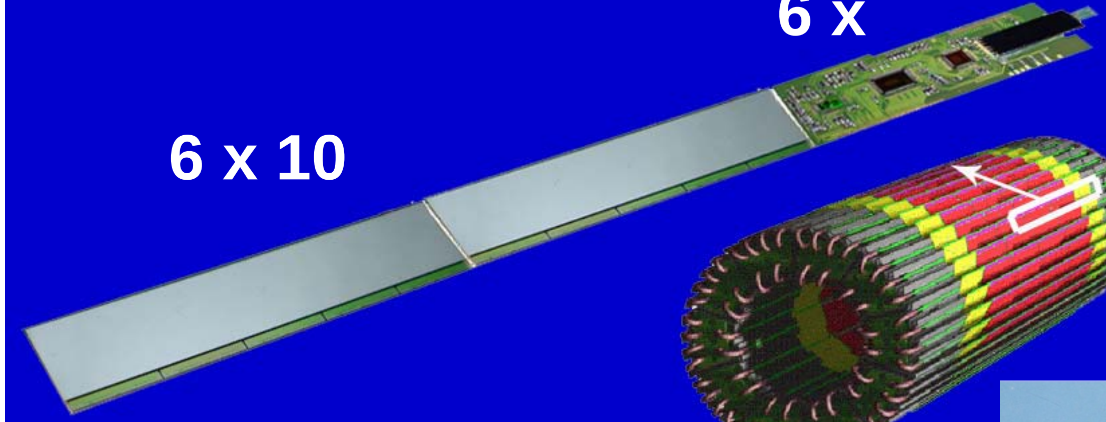


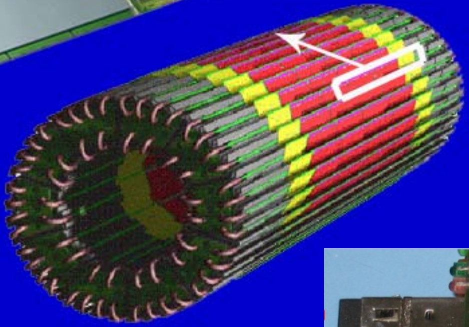
Data selection and delay



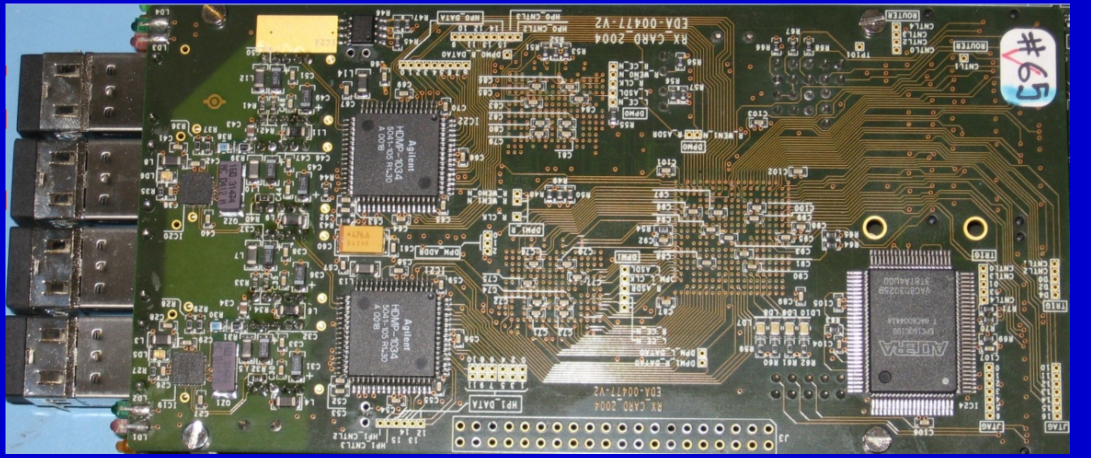
System level simulation

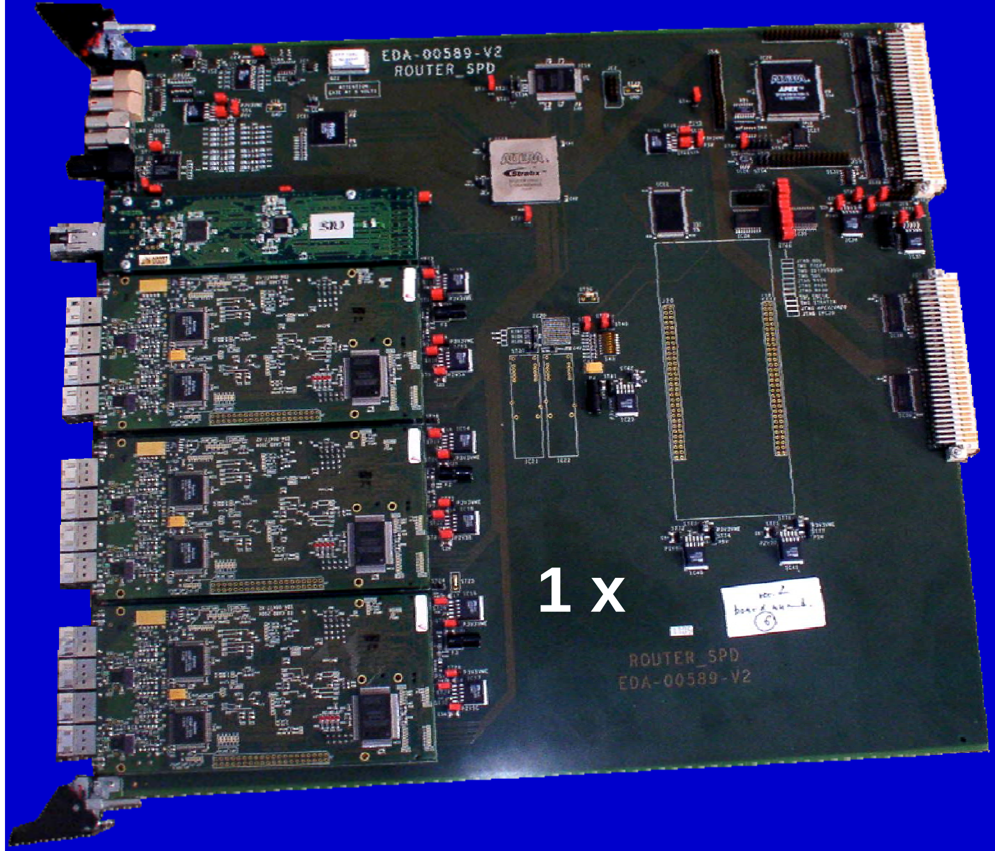
System level simulation





3 x





- 60 ASICs: simplified behavioral
- 40 ASICs: full behavioral
- 5 FPGA: full behavioral
- 7 SRAMs: full behavioral
- 4 PCBs

What happens if we have speed problems?

- **Often because of inadequate logic architecture/coding style**
  - evaluate logic architecture
  - rewrite HDL code to adapt structure to better data throughput
  - insert pipeline structure - often one clock cycle more latency does not matter
  - Understand the specifications
  - look for systematics which can help to simplify logic
  - adapt architecture and schematics/code
  - only then optimize placing & routing

What happens if we have speed problems?

- **Often because of components too small and routing congestion**
  - timing constraints
  - Routing constraint - placement constraint
  - Use bigger/faster component

Conclusion

- **FPGA application at CERN**
  - data selection/trigger (muon track finder trigger)
  - data processing (pixel detector)
- **Design cycle**
- **Defining Specifications**
- **Clock domains**
- **Data delay**

Additional slides

- Alexander.kluge@cern.ch
- http://akluge.web.cern.ch/akluge