Synthesis

Synthesis Tools

Xilinx XST

Synplify Premier

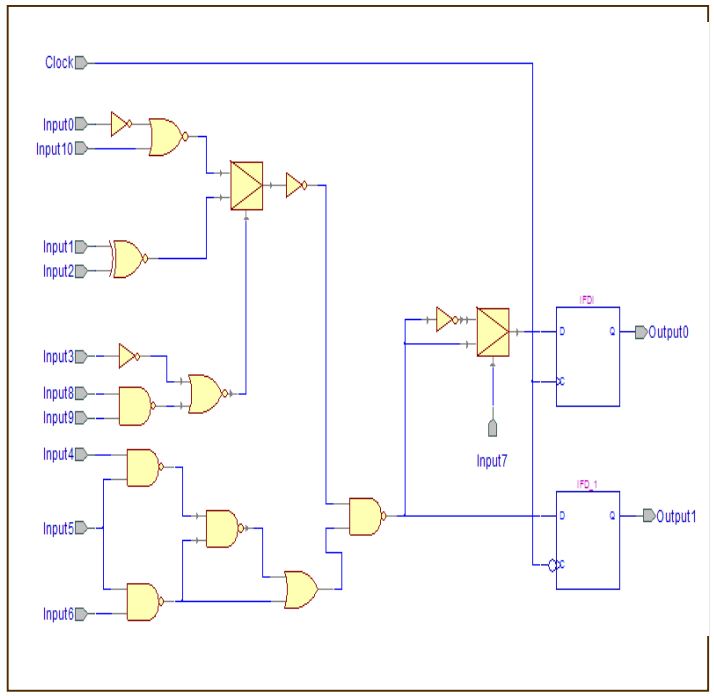
… and others

Logic Synthesis

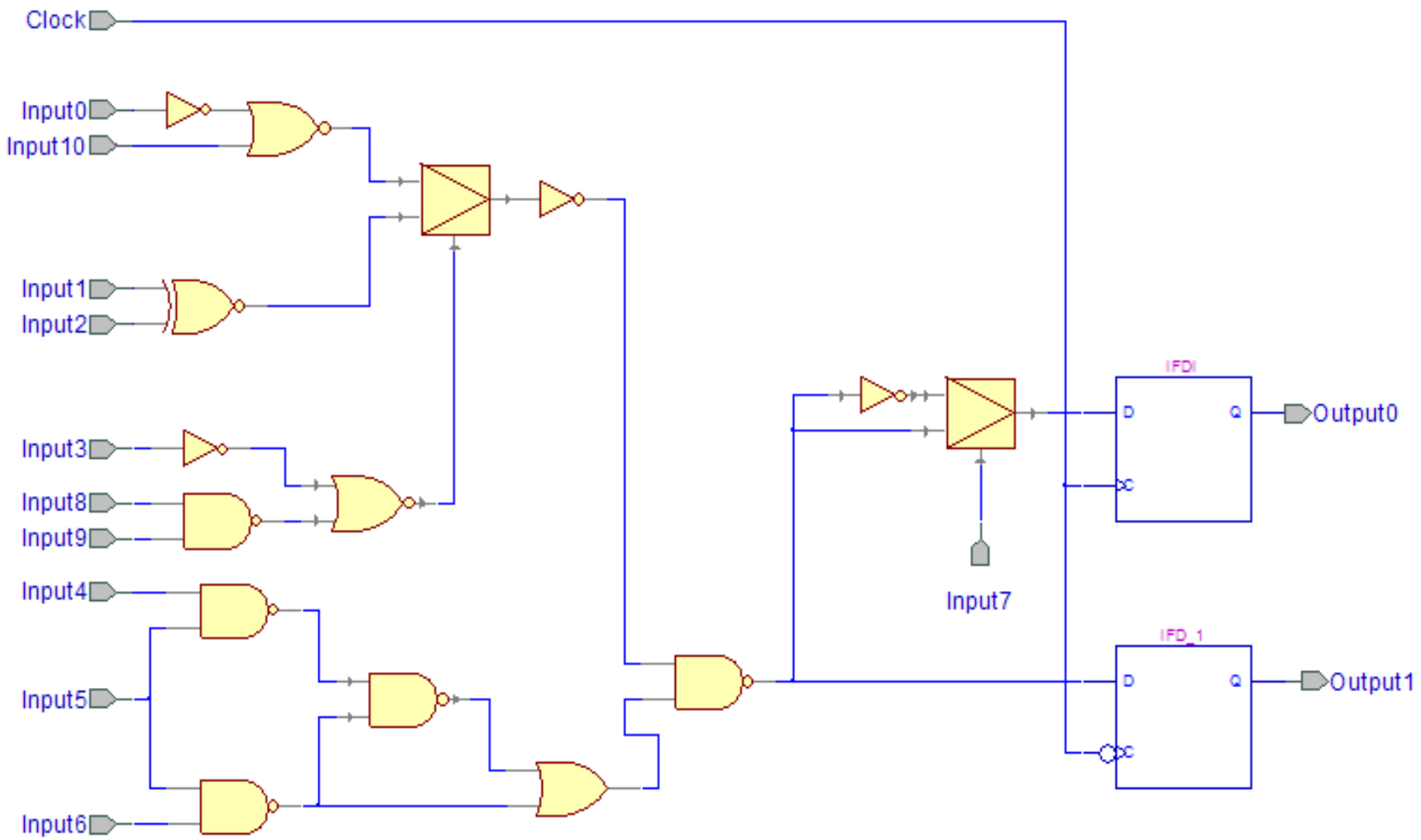
VHDL description

```
architecture MLU_DATAFLOW of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
                A1<=A when (NEG_A='0') else
                                not A;
                B1<=B when (NEG_B='0') else
                                not B;
                Y<=Y1 when (NEG_Y='0') else
                                not Y1;

                MUX_0<=A1 and B1;
                MUX_1<=A1 or B1;
                MUX_2<=A1 xor B1;
                MUX_3<=A1 xnor B1;

                with (L1 & L0) select
                                Y1<=MUX_0 when "00",
                                                MUX_1 when "01",
                                                MUX_2 when "10",
                                                MUX_3 when others;

end MLU_DATAFLOW;
```

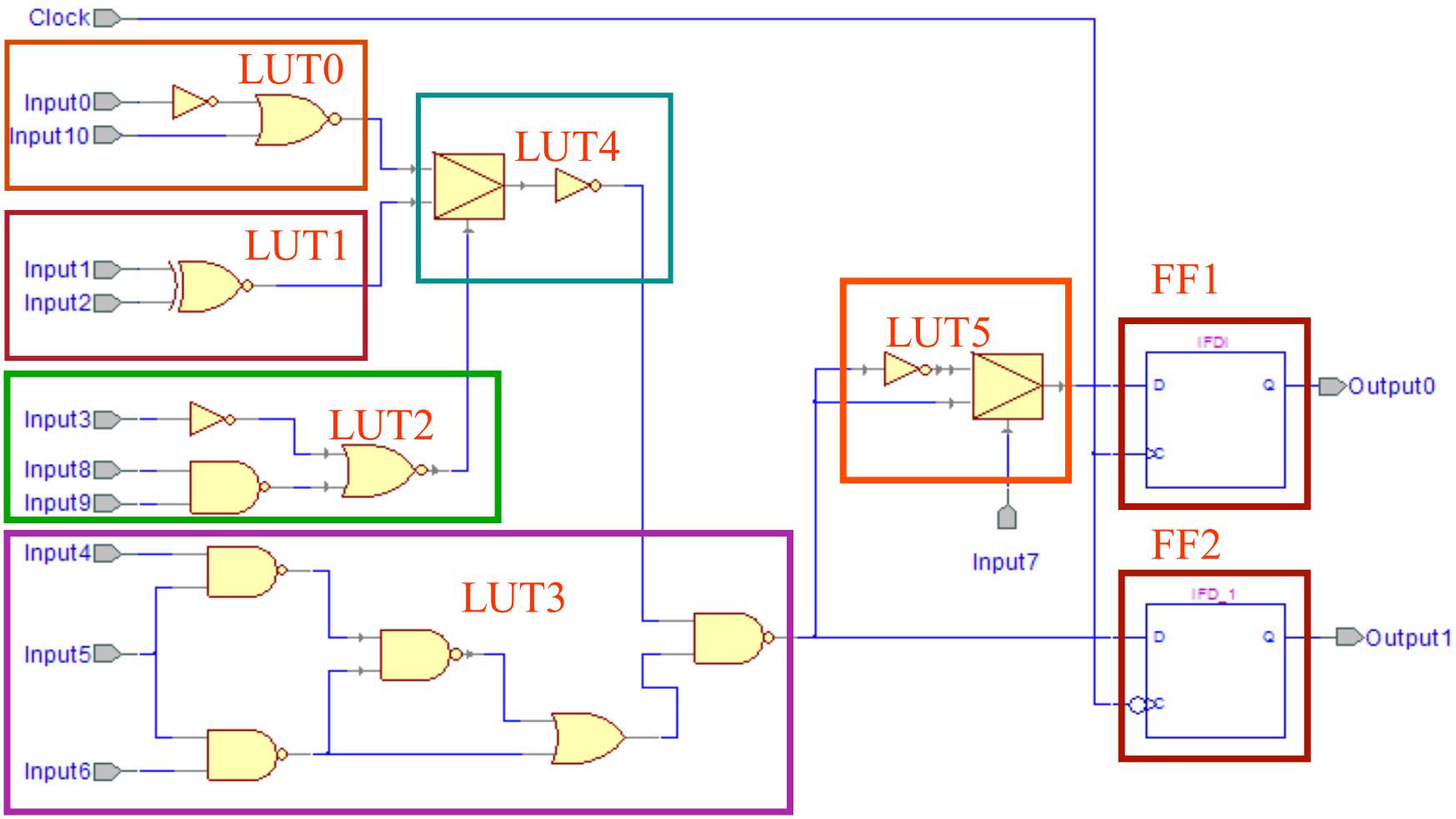
Circuit netlist



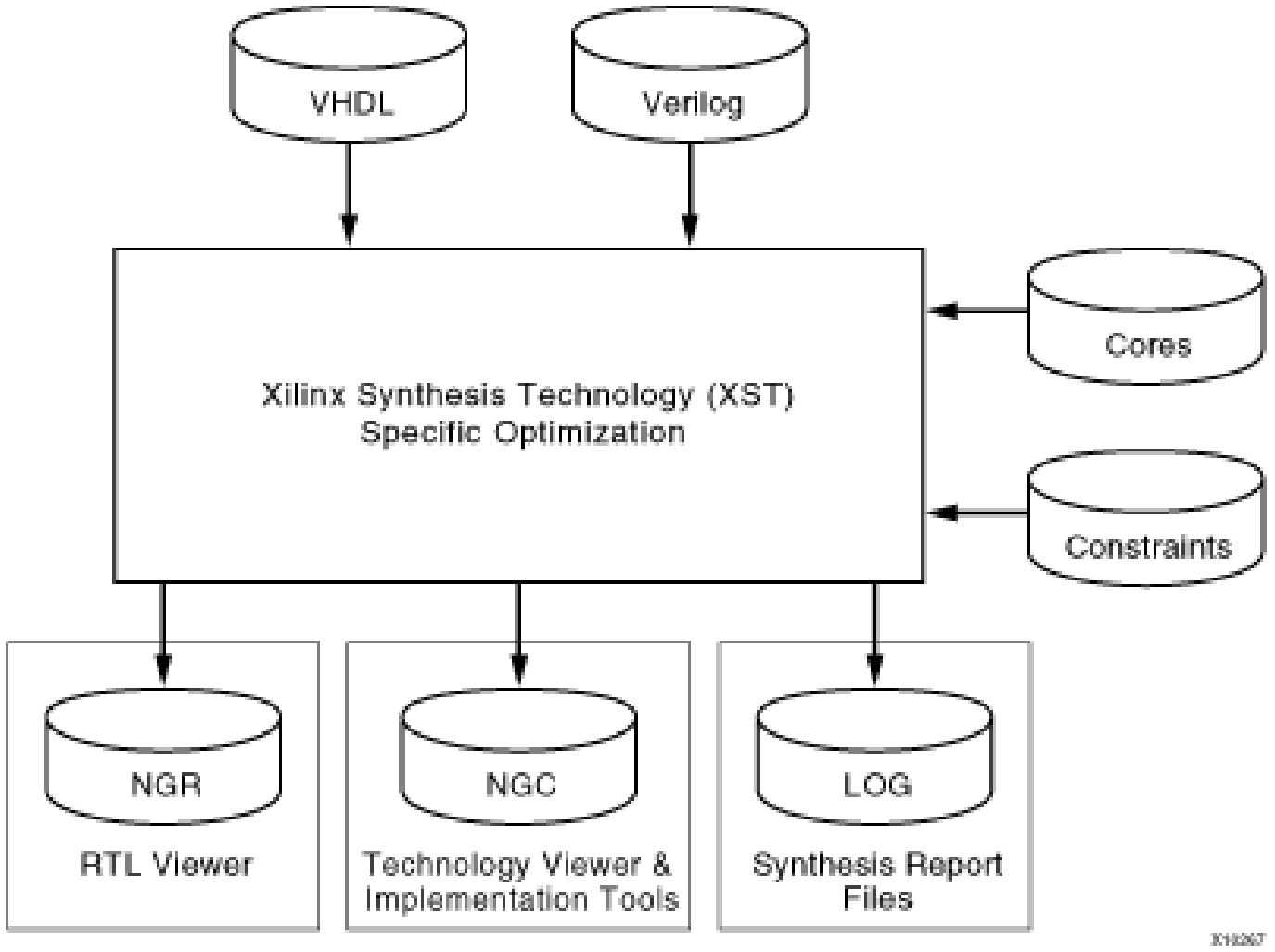
Circuit netlist (RTL view)



Mapping


Clock
LUT0
Input0
Input10
LUT4
LUT1
Input1
Input2
LUT5
FF1
IFDI
D
Q
C
Output0
Input3
LUT2
Input8
Input9
Input7
FF2
IFD_1
Input4
LUT3
Input5
Input6
Output1

Xilinx XST Inputs/Outputs



Xilinx XST Inputs

- RTL VHDL and/or Verilog files
- Core files

  These files can be in either NGC or EDIF format. XST does not modify cores. It uses them to inform area and timing optimization surrounding the cores.
- Constraints – XCF

  Xilinx constraints file in which you can specify synthesis, timing, and specific implementation constraints that can be propagated to the NGC file.

Xilinx XST Outputs

- NGC

  Netlist file with constraint information

- NGR

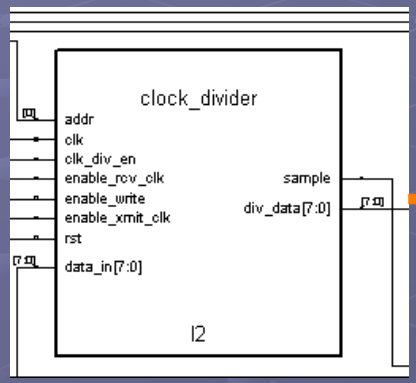
  This is a schematic representation of the pre-optimized design shown at the Register Transfer Level (RTL). This representation is in terms of generic symbols, such as adders, multipliers, counters, AND gates, and OR gates, and is generated after the HDL synthesis phase of the synthesis process.

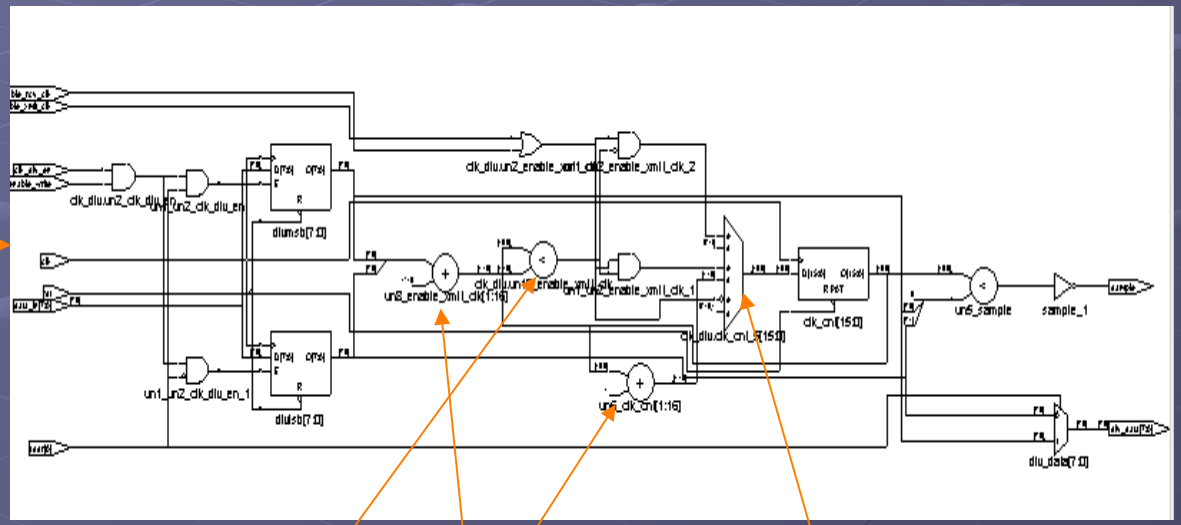
- LOG

  This report contains the results from the synthesis run, including area and timing estimation.

RTL view in Synplify Premier

- General logic structures can be recognized in RTL view





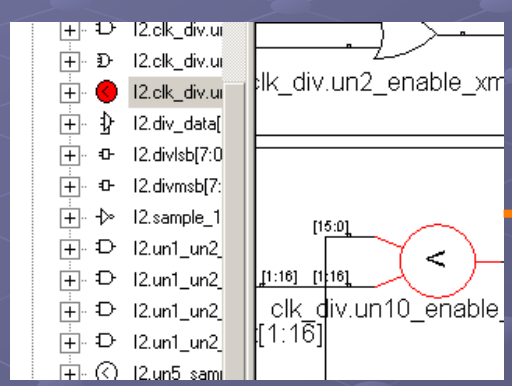
comparator

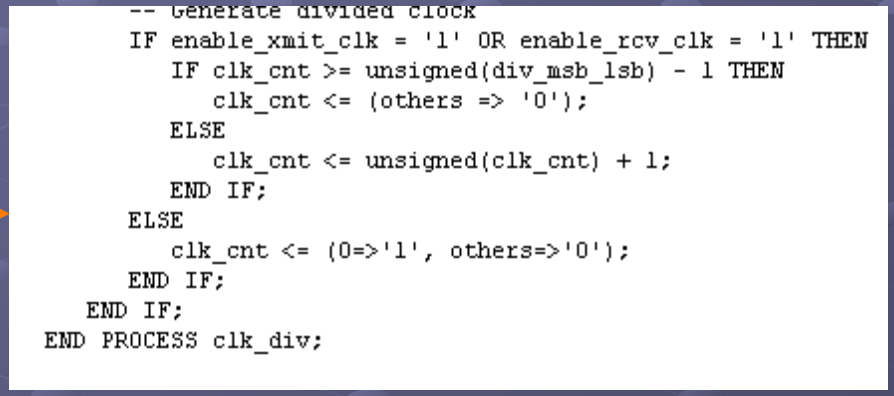
incrementer

MUX

Crossprobing between RTL view and code

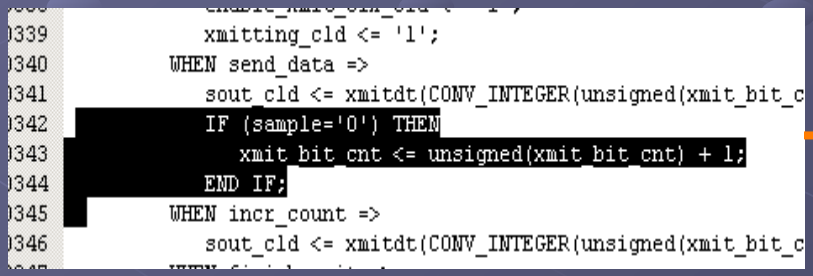
- Each port, net or block can be chosen by mouse click from the browser or directly from the RTL View
- By double-clicking on the element its source code can be seen:



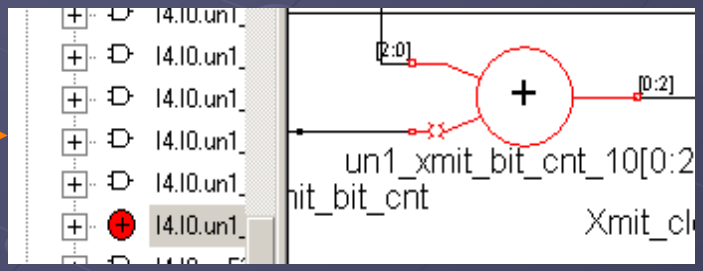


```
        -- Generate divided clock
        IF enable_xmit_clk = '1' OR enable_rcv_clk = '1' THEN
           IF clk_cnt >= unsigned(div_msb_lsb) - 1 THEN
              clk_cnt <= (others => '0');
           ELSE
              clk_cnt <= unsigned(clk_cnt) + 1;
           END IF;
        ELSE
           clk_cnt <= (0=>'1', others=>'0');
        END IF;
     END IF;
  END PROCESS clk_div;
```

- Reverse crossprobing is also possible: if section of code is marked, appropriate element of RTL View is marked too:

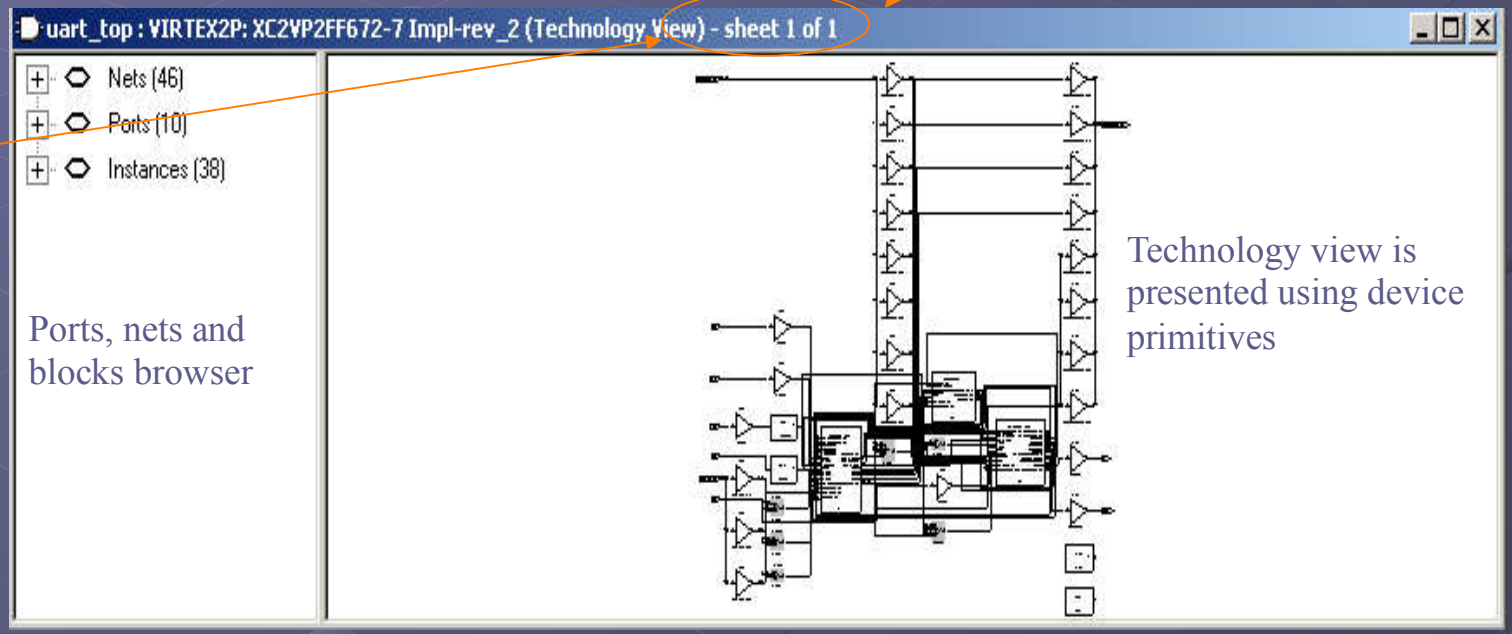


```
0339          xmitting_cld <= '1';
0340       WHEN send_data =>
0341          sout_cld <= xmitdt(CONV_INTEGER(unsigned(xmit_bit_c
0342          IF (sample='0') THEN
0343             xmit_bit_cnt <= unsigned(xmit_bit_cnt) + 1;
0344          END IF;
0345       WHEN incr_count =>
0346          sout_cld <= xmitdt(CONV_INTEGER(unsigned(xmit_bit_c
```



Technology View in Synplify Pro

- Technology view is a mapped RTL view. It can be seen by pressing  button or by double-click on ".srm" file
- As in case of "RTL View", buttons  can be used here
- Two additional buttons are enabled:  - show critical path
   - open timing analyst



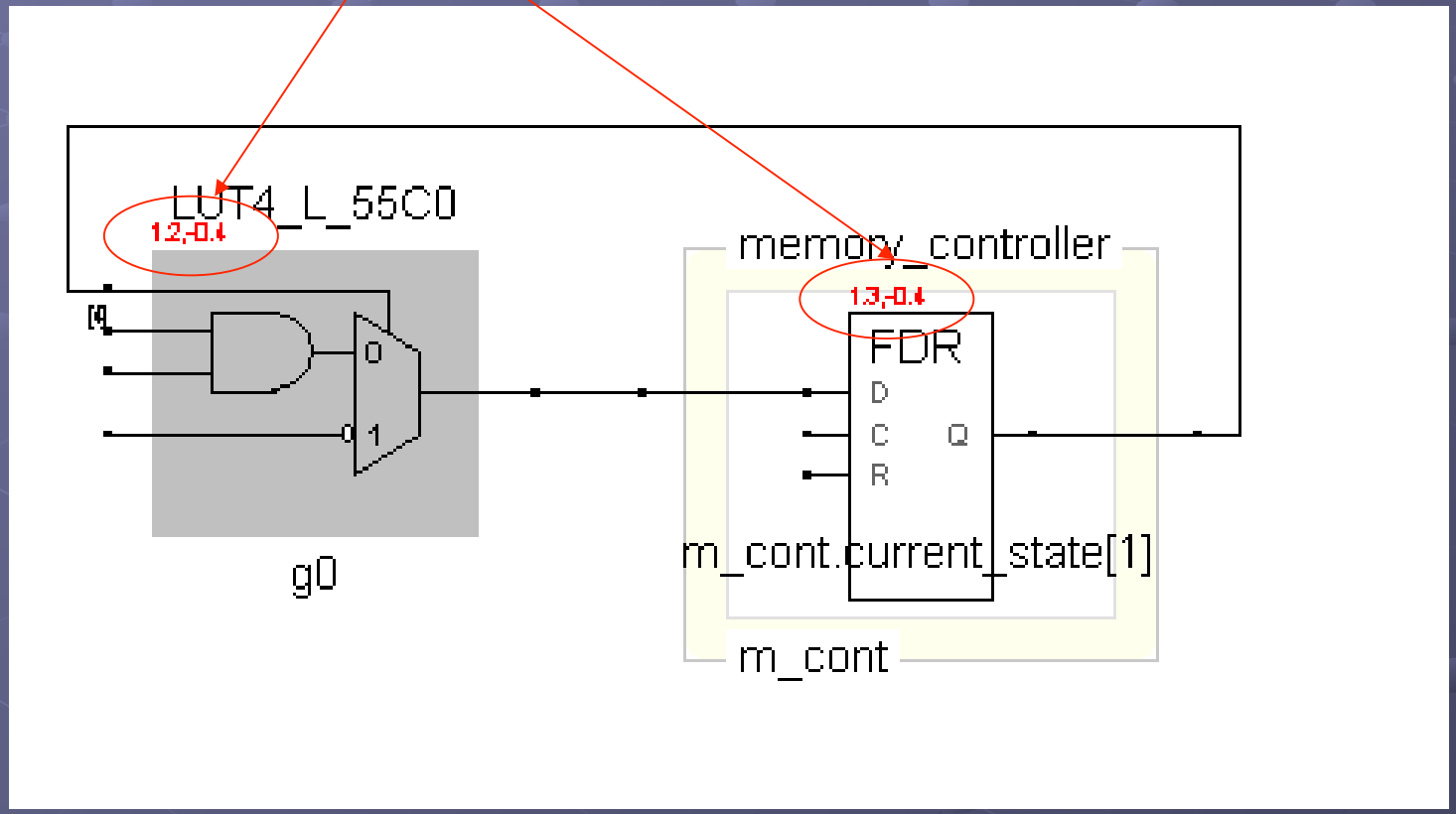
Pay attention: technology view is usually large and presented on number of sheets

Ports, nets and blocks browser

Technology view is presented using device primitives

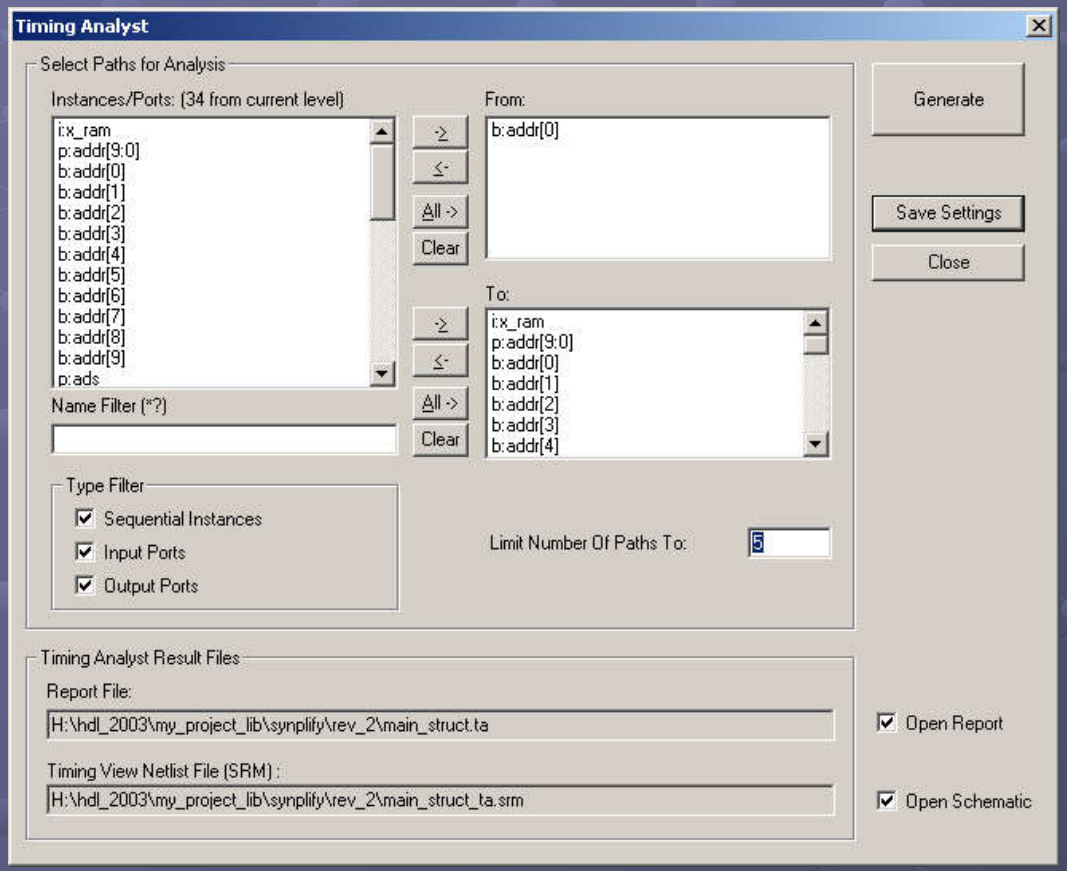
Viewing critical path

- Critical path can be viewed by pressing on 
- Delay values are written near each component of the path



Timing Analyst

- Timing analyst opened by pressing on 
- Timing analyst gives a possibility to analyze different paths in the design
- Timing analyst can be opened only from Technology View

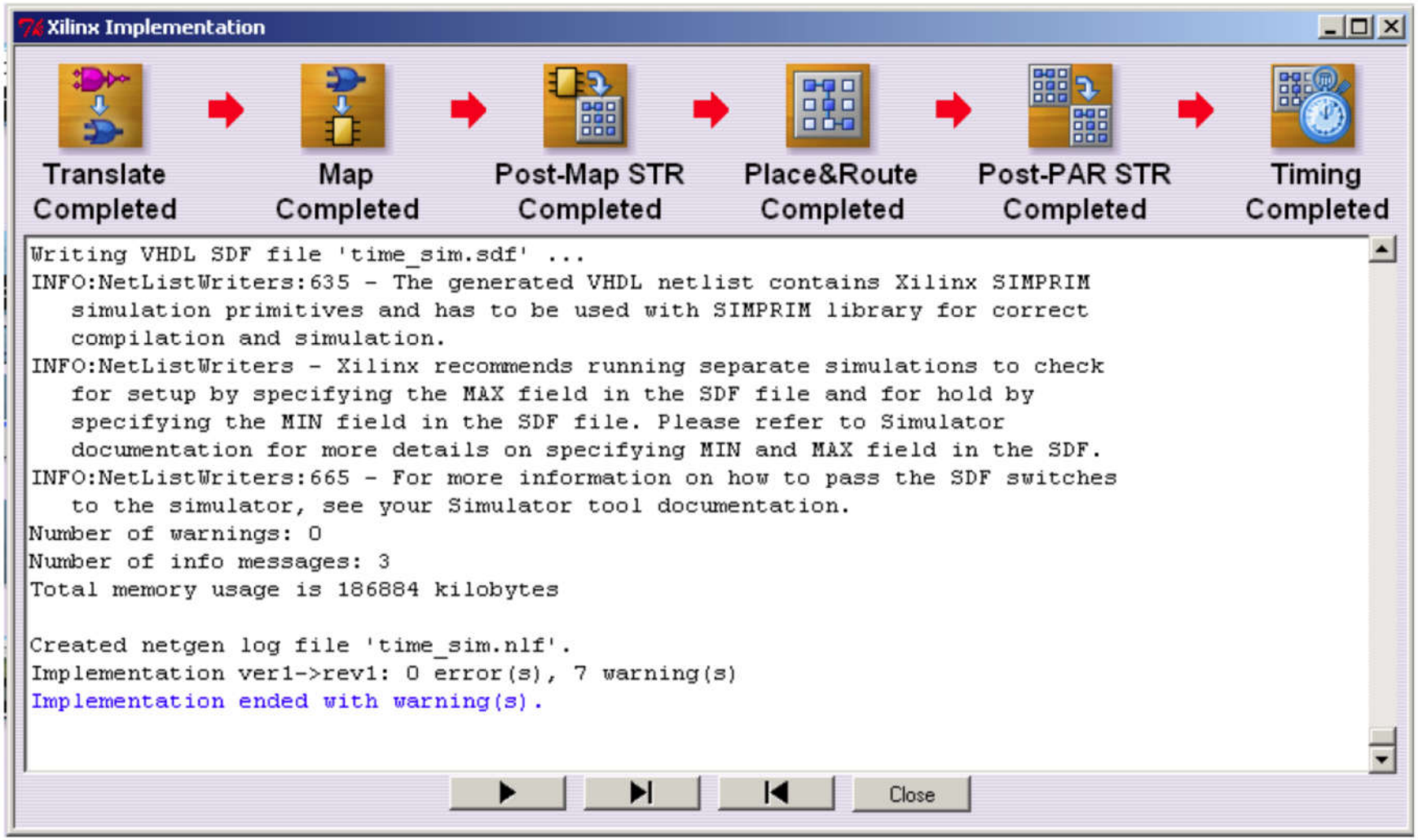


Implementation

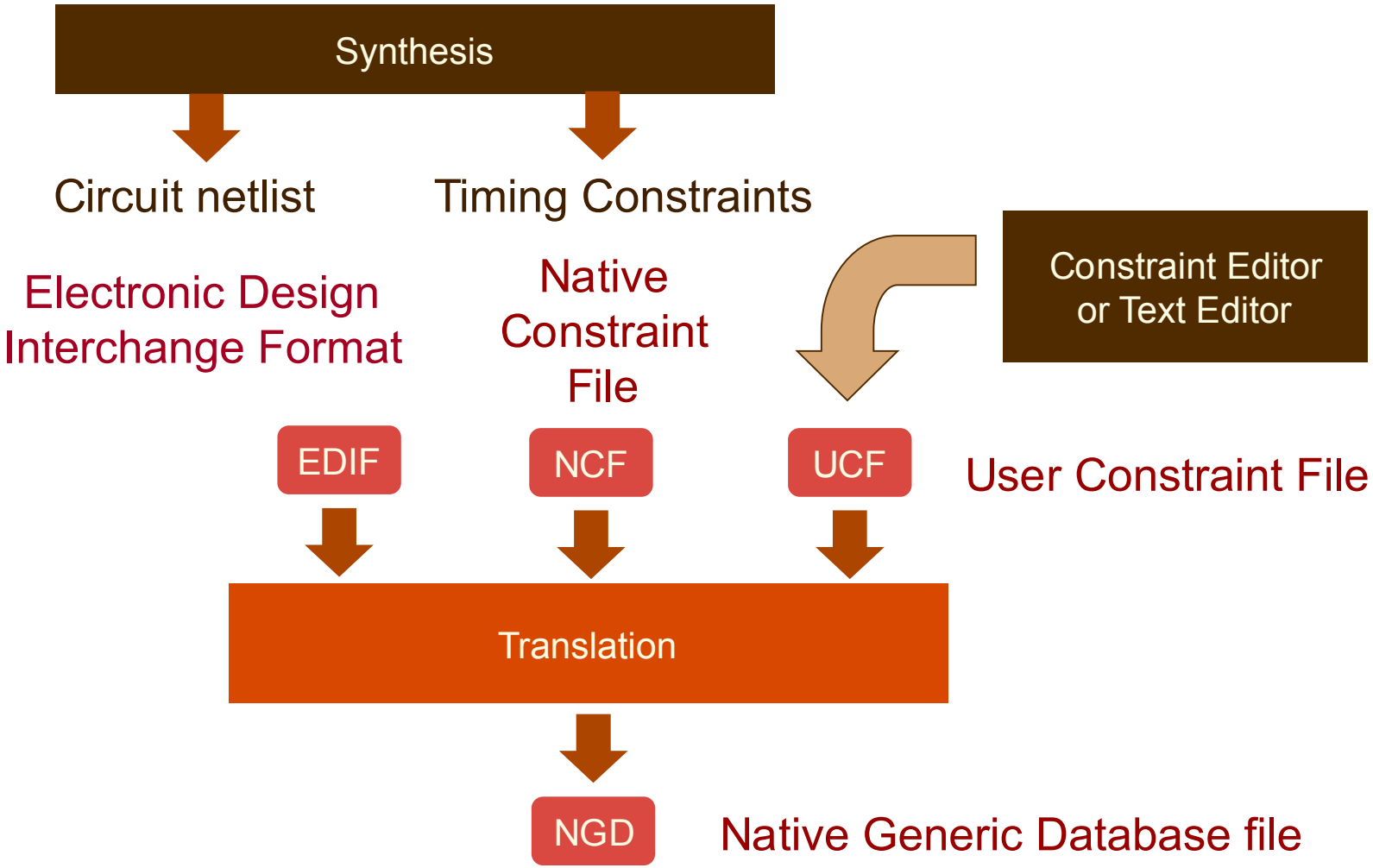
Implementation

- After synthesis the entire implementation process is performed by FPGA vendor tools

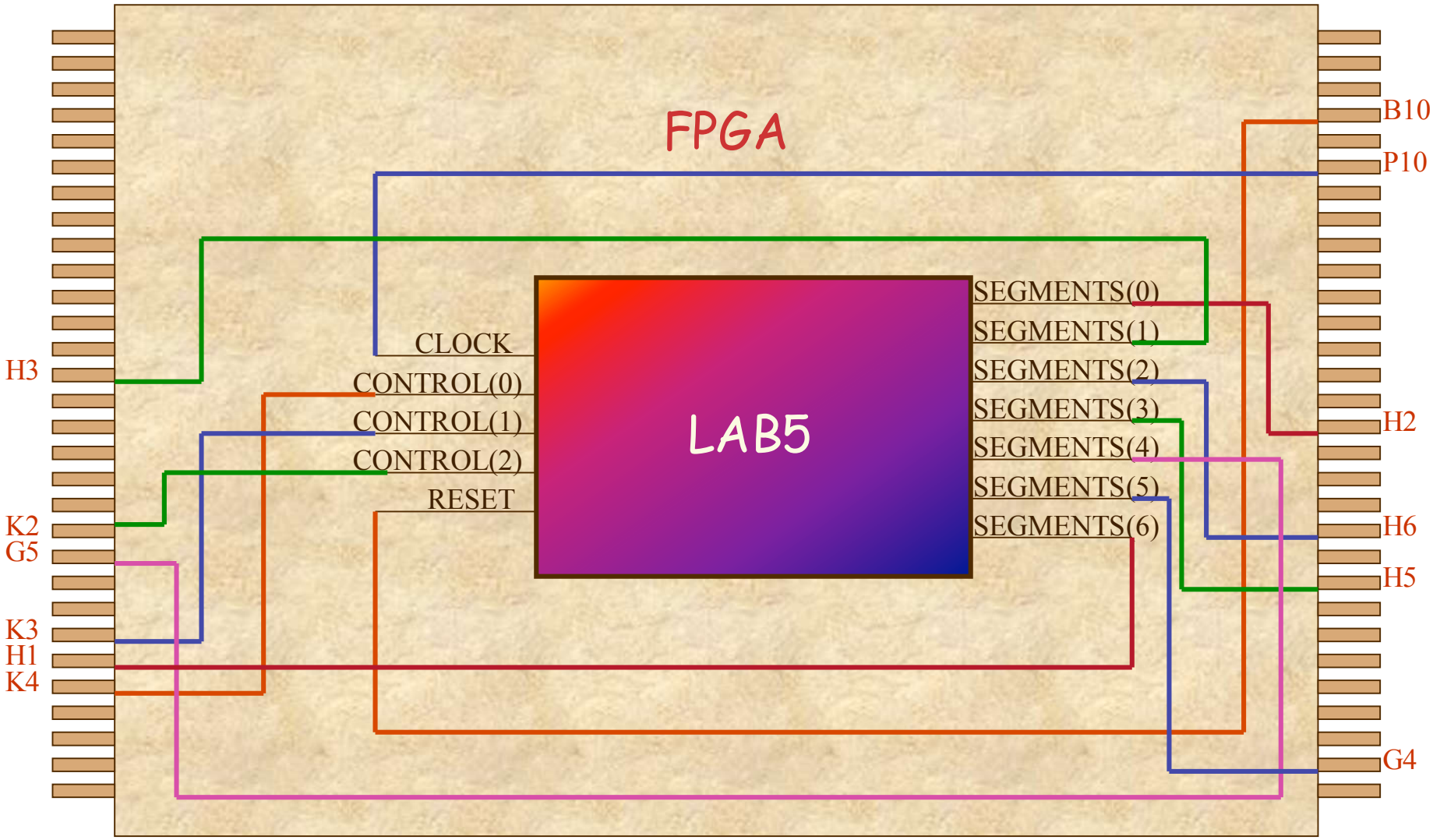
Implementation



Translation

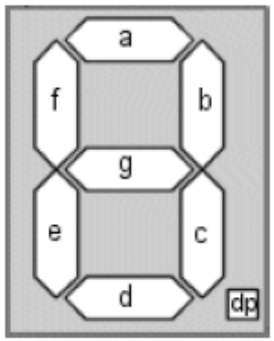


Pin Assignment



Description	*FPGA Pins*
Seven Segment 0 'a'	H2
Seven Segment 0 'b'	H3
Seven Segment 0 'c'	H6
Seven Segment 0 'd'	H5
Seven Segment 0 'e'	G5
Seven Segment 0 'f'	G4
Seven Segment 0 'g'	H1
Seven Segment 0 'dp'	C2
Seven Segment 1 'a'	J1
Seven Segment 1 'b'	J2
Seven Segment 1 'c'	K2
Seven Segment 1 'd'	C3
Seven Segment 1 'e'	C1
Seven Segment 1 'f'	H4
Seven Segment 1 'g'	B1
Seven Segment 1 'dp'	J4

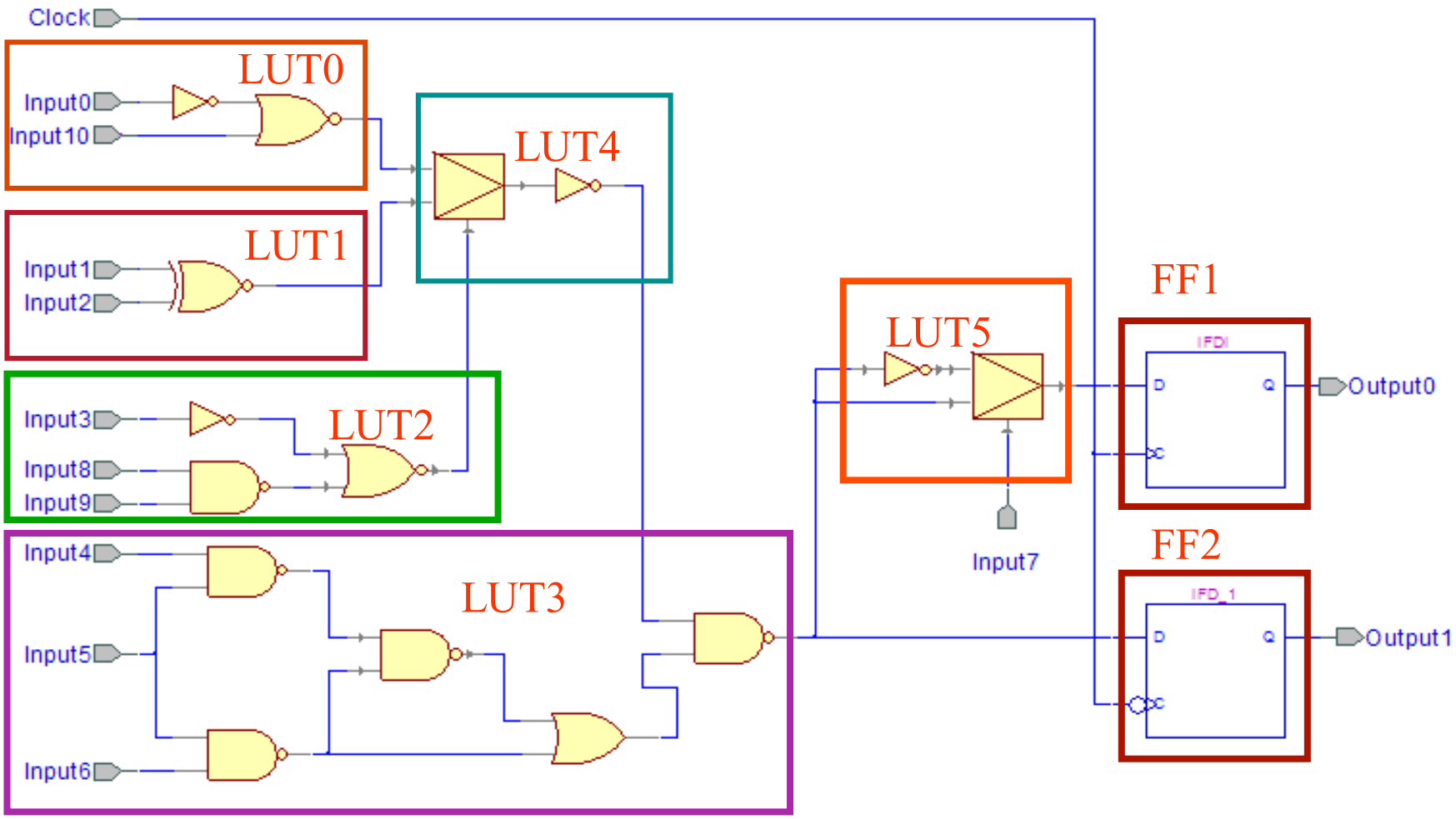
The segments of the display are labelled "a-g" and "dp" in the table above and the figure below.



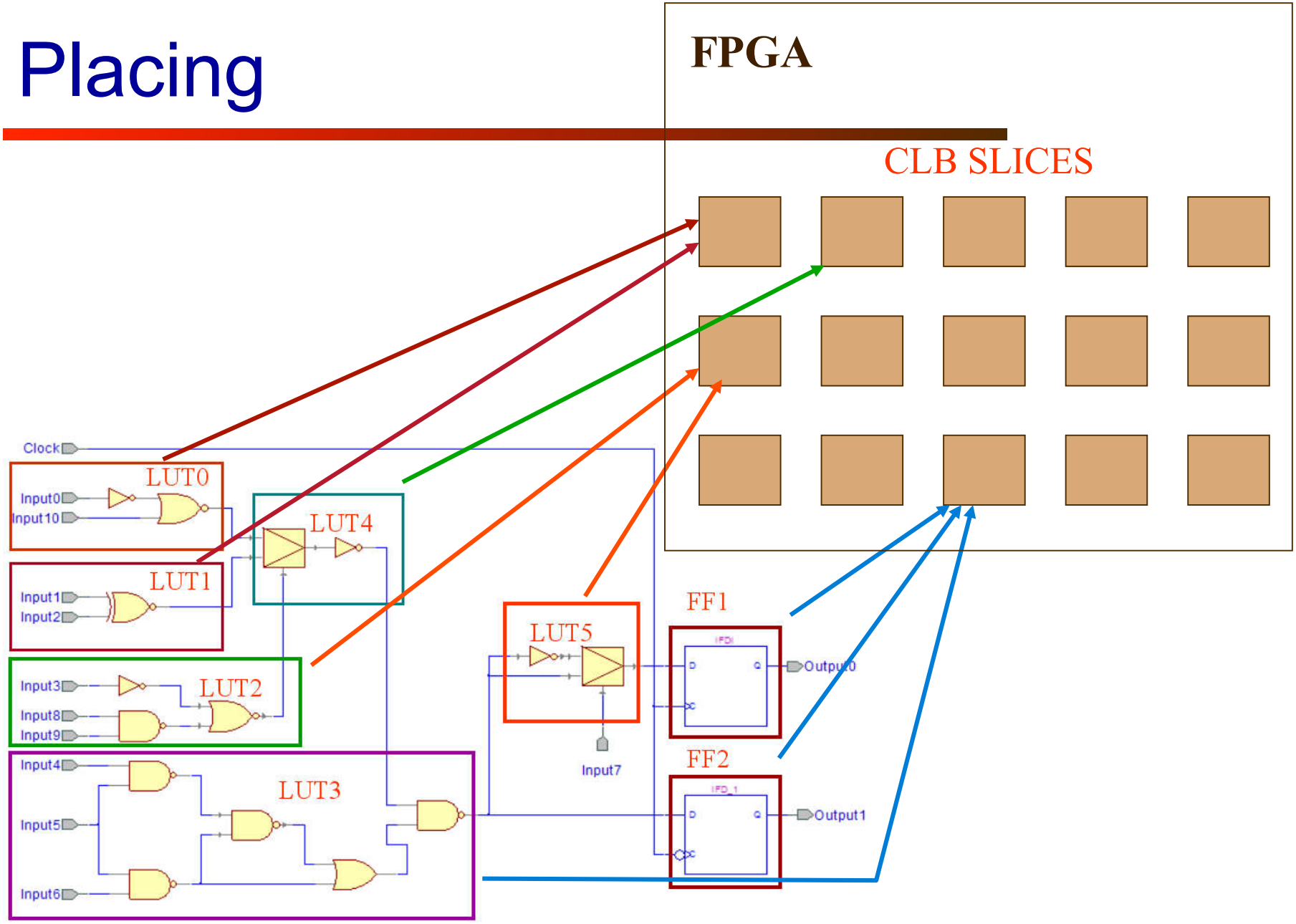
Example of an UCF File

```
NET "CLOCK" LOC = "P10";
NET "reset" LOC = "B10";
NET "S_SEG0<6>" LOC = "H1";
NET "S_SEG0<5>" LOC = "G4";
NET "S_SEG0<4>" LOC = "G5";
NET "S_SEG0<3>" LOC = "H5";
NET "S_SEG0<2>" LOC = "H6";
NET "S_SEG0<1>" LOC = "H3";
NET "S_SEG0<0>" LOC = "H2";
```

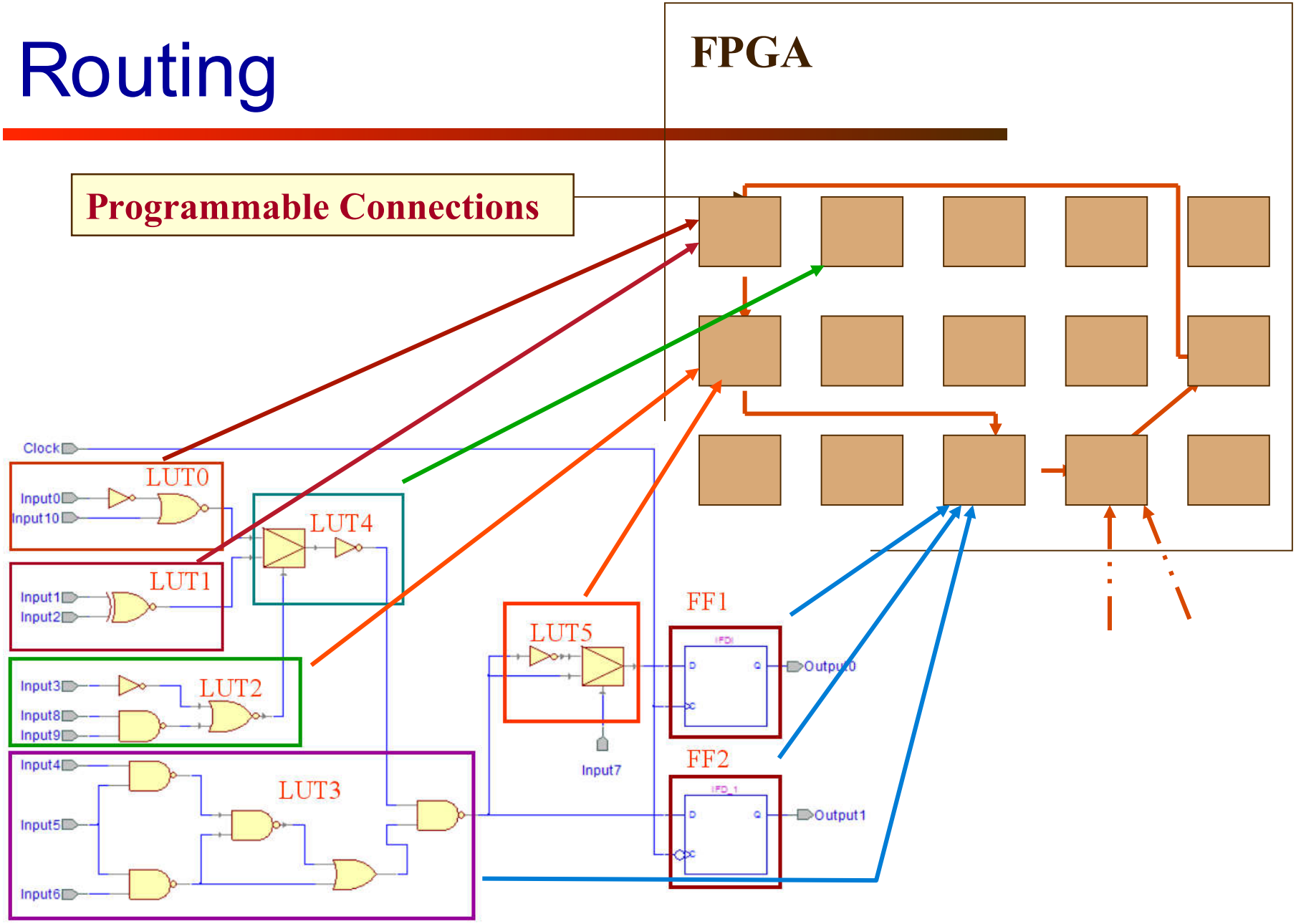
Mapping


Clock
LUT0
Input0
Input10
LUT4
LUT1
Input1
Input2
LUT5
FF1
IFDI
D
Q
C
Output0
Input3
LUT2
Input8
Input9
Input7
FF2
IFD_1
Input4
LUT3
Input5
Input6
Output1

Placing


FPGA
CLB SLICES
Clock
LUT0
Input0
Input10
LUT1
Input1
Input2
LUT4
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT5
Input7
FF1
FF2
Output0
Output1

Routing

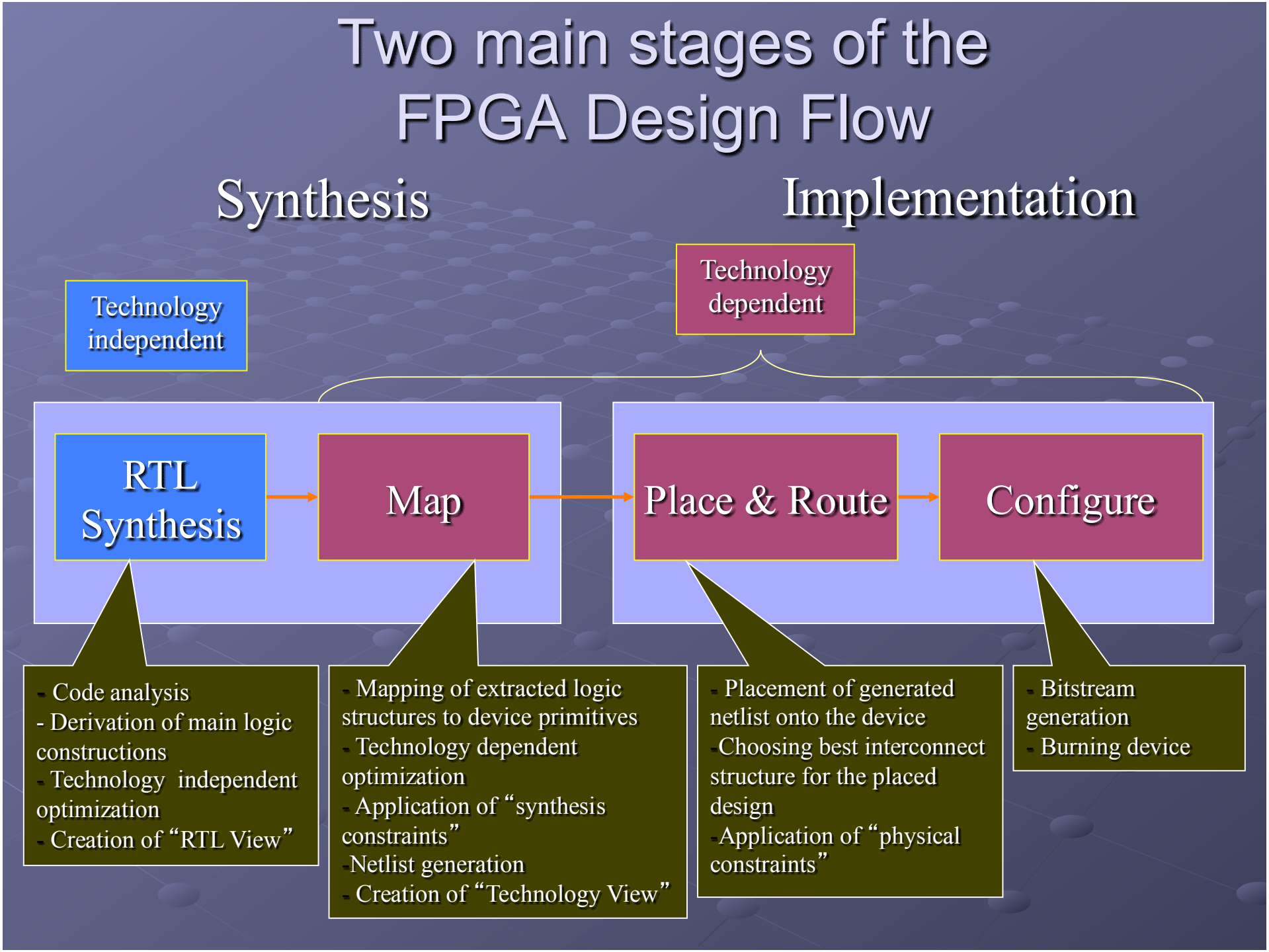

FPGA
Programmable Connections
Clock
LUT0
Input0
Input10
LUT1
Input1
Input2
LUT4
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT5
Input7
FF1
Output0
FF2
Output1

Configuration

- Once a design is implemented, you must create a file that the FPGA can understand
  - This file is called a bit stream: a BIT file (.bit extension)

- The BIT file can be downloaded directly to the FPGA, or can be converted into a PROM file which stores the programming information

Two main stages of the FPGA Design Flow



Report files

Map report header

Xilinx Mapping Report File for Design 'Lab3Demo'

Design Information

Command Line : c:\Xilinx\bin\nt\map.exe -p 3S1500FG320-4 -o map.ncd -pr b -k 4 -cm area -c 100 Lab3Demo.ngd Lab3Demo.pcf
Target Device : **xc3s1500**
Target Package : **fg320**
Target Speed : **-4**
Mapper Version : spartan3 -- $Revision: 1.34 $

Map report

Design Summary

Number of errors: 0
Number of warnings: 0
Logic Utilization:
Number of Slice Flip Flops: 30 out of 26,624 1%
Number of 4 input LUTs: 38 out of 26,624 1%
Logic Distribution:
Number of occupied Slices: 33 out of 13,312 1%
Number of Slices containing only related logic: 33 out of 33 100%
Number of Slices containing unrelated logic: 0 out of 33 0%
*See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs: 62 out of 26,624 1%
Number used as logic: 38
Number used as a route-thru: 24
Number of bonded IOBs: 10 out of 221 4%
IOB Flip Flops: 7
Number of GCLKs: 1 out of 8 12%

Related and Unrelated Logic

Related logic is defined as being logic that shares connectivity – e.g. two LUTs are "related" if they share common inputs.
When assembling slices, Map gives priority to combine logic that is related. Doing so results in the best timing performance.

Unrelated logic shares no connectivity. Map will only begin packing unrelated logic into a slice once 99% of the slices are occupied through related logic packing.

Note that once logic distribution reaches the 99% level through related logic packing, this does not mean the device is completely utilized. Unrelated logic packing will then begin, continuing until all usable LUTs and FFs are occupied.
Depending on your timing budget, increased levels of unrelated logic packing may adversely affect the overall timing performance of your design.

Place & route report

```
Asterisk (*) preceding a constraint indicates it was not met.
   This may be due to a setup or hold violation.

--------------------------------------------------------------------------------------------
  Constraint                                | Requested  | Actual     | Logic  | Absolute   |Number of
                                            |            |            | Levels | Slack      |errors
--------------------------------------------------------------------------------------------
* TS_CLOCK = PERIOD TIMEGRP "CLOCK" 5 ns    | 5.000ns    | 5.140ns    | 4      | -0.140ns   | 5
  HIGH 50%                                  |            |            |        |            |
--------------------------------------------------------------------------------------------
  TS_gen1Hz_Clock1Hz = PERIOD TIMEGRP "gen1 | 5.000ns    | 4.137ns    | 2      | 0.863ns    | 0
  "gen1Hz_Clock1Hz" 5 ns HIGH 50%           |            |            |        |            |
--------------------------------------------------------------------------------------------
```

Post layout timing report

```
Clock to Setup on destination clock CLOCK
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
CLOCK          |    5.140|         |         |         |
---------------+---------+---------+---------+---------+


Timing summary:
---------------

Timing errors: 9  Score: 543

Constraints cover 574 paths, 0 nets, and 187 connections

Design statistics:
   Minimum period:   5.140ns (Maximum frequency: 194.553MHz)
```