FPGA Design Flow

Outline

- Overall Flow
- Logic Synthesis
- Mapping
- Place & Route
- Simulation
- Configuration
- Board-level consideration

FPGA Design Process (1)

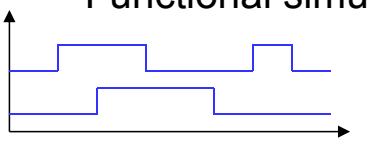
Specification

Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds…..

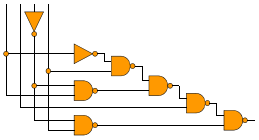
HDL description (Your HDL Source Files)

```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
        port(
                clock, reset, ener_decr: in std_logic;
                data_input: in std_logic_vector(31 downto 0);
                data_output: out std_logic_vector(31 downto 0);
                out_full: in std_logic;
                key_input: in std_logic_vector(31 downto 0);
                key_read: out std_logic;
        );
end AES_core;
```

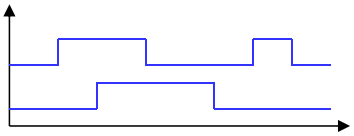
Functional simulation



Synthesis

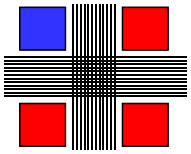


Post-synthesis simulation

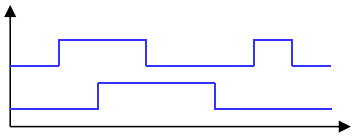


FPGA Design Process (2)

Implementation
(Mapping, Placing & Routing)

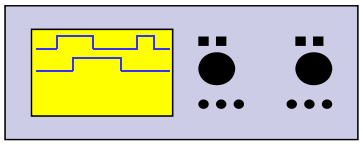


Timing simulation



Configuration

On chip testing

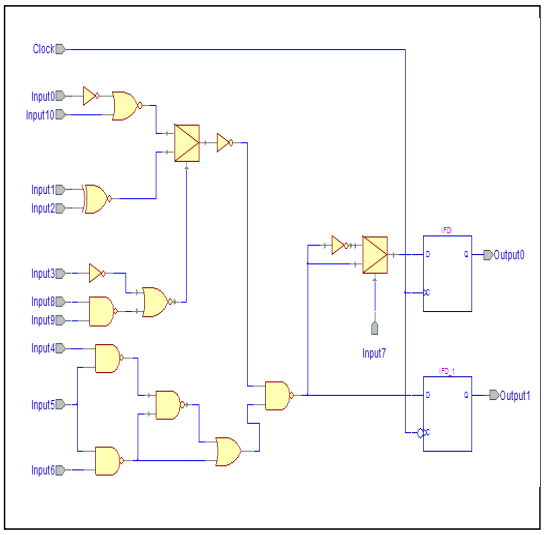


Logic Synthesis

VHDL/Verilog description

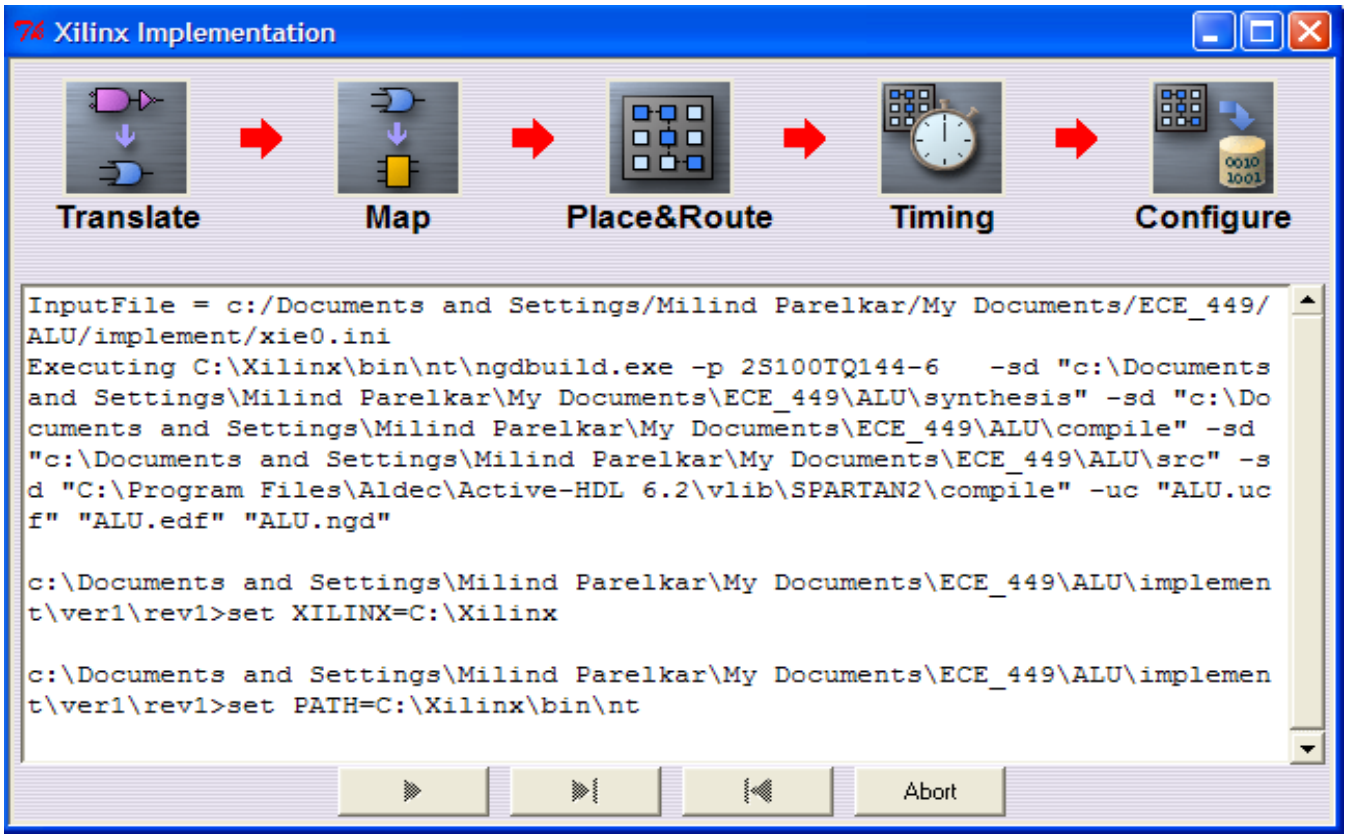
```
architecture MLU_DATAFLOW of MLU is

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

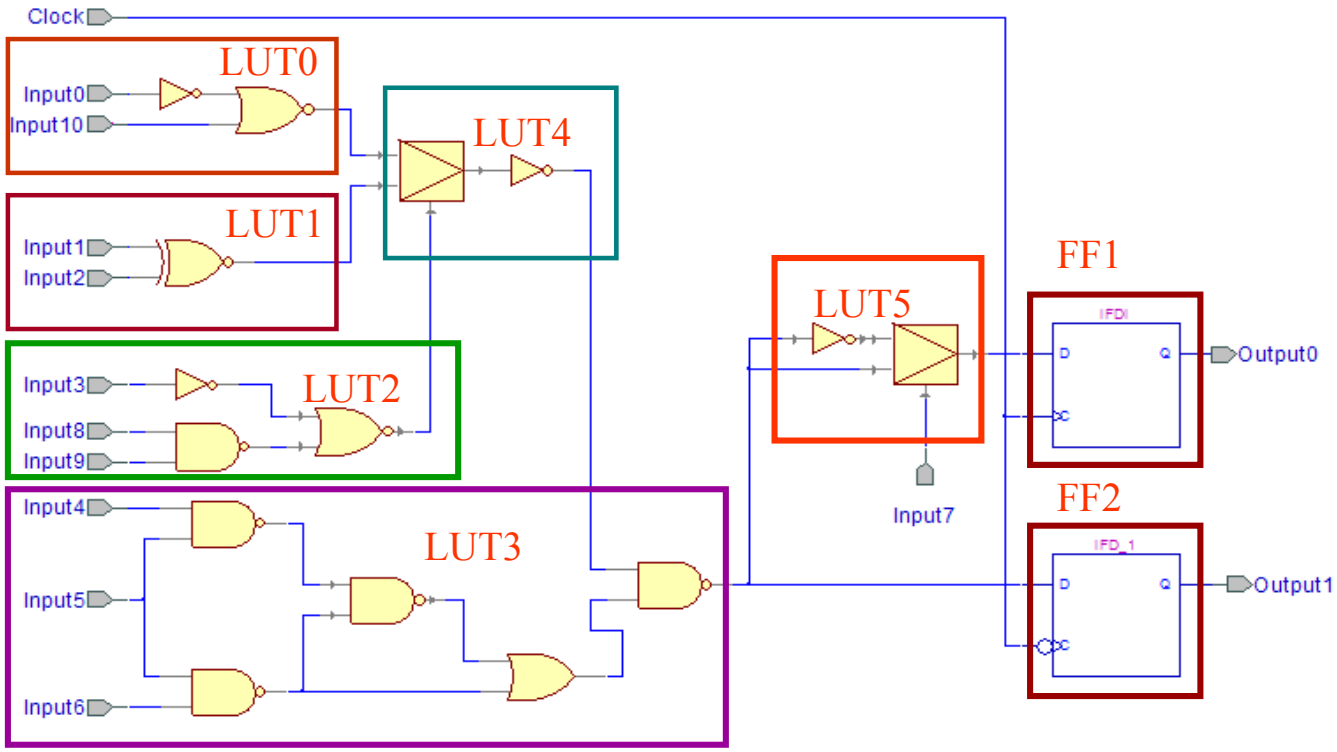


Features of Synthesis Tools

- Interpret RTL code
- Produce synthesized circuit netlist in a standard EDIF format
- Give preliminary performance estimates
- Some can display circuit schematics corresponding to EDIF netlist


Xilinx Implementation
Translate
Map
Place&Route
Timing
Configure
InputFile = c:/Documents and Settings/Milind Parelkar/My Documents/ECE_449/ALU/implement/xie0.ini
Executing C:\Xilinx\bin\nt\ngdbuild.exe -p 2S100TQ144-6 -sd "c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\synthesis" -sd "c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\compile" -sd "c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\src" -sd "C:\Program Files\Aldec\Active-HDL 6.2\vlib\SPARTAN2\compile" -uc "ALU.ucf" "ALU.edf" "ALU.ngd"
c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\implement\ver1\rev1>set XILINX=C:\Xilinx
c:\Documents and Settings\Milind Parelkar\My Documents\ECE_449\ALU\implement\ver1\rev1>set PATH=C:\Xilinx\bin\nt
Abort

Mapping

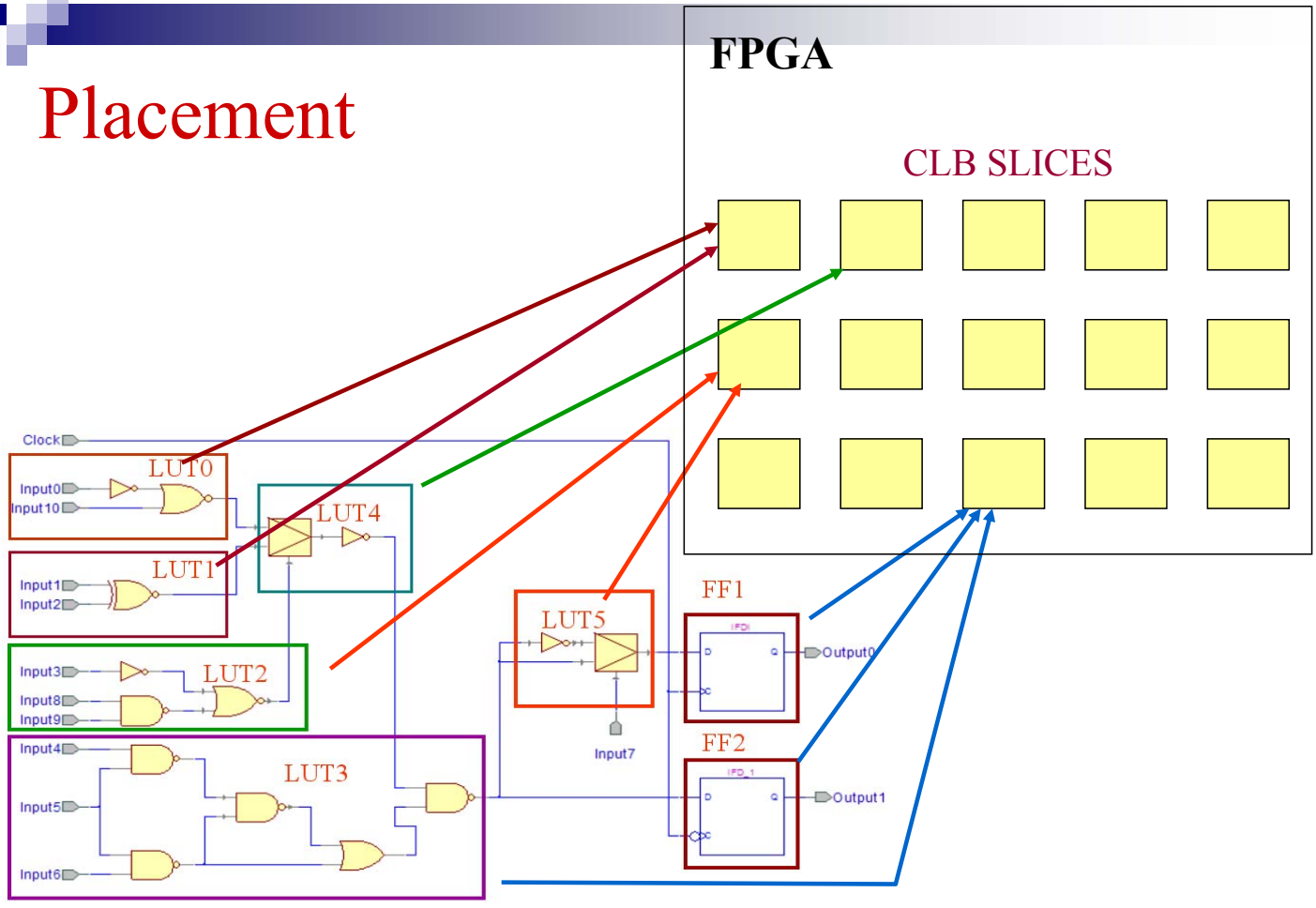

Clock
LUT0
Input0
Input10
LUT4
LUT1
Input1
Input2
LUT5
FF1
IFDI
D
Q
C
Output0
Input3
LUT2
Input8
Input9
Input7
FF2
IFD_1
Input4
LUT3
Input5
Input6
Output1

Sample mapping report

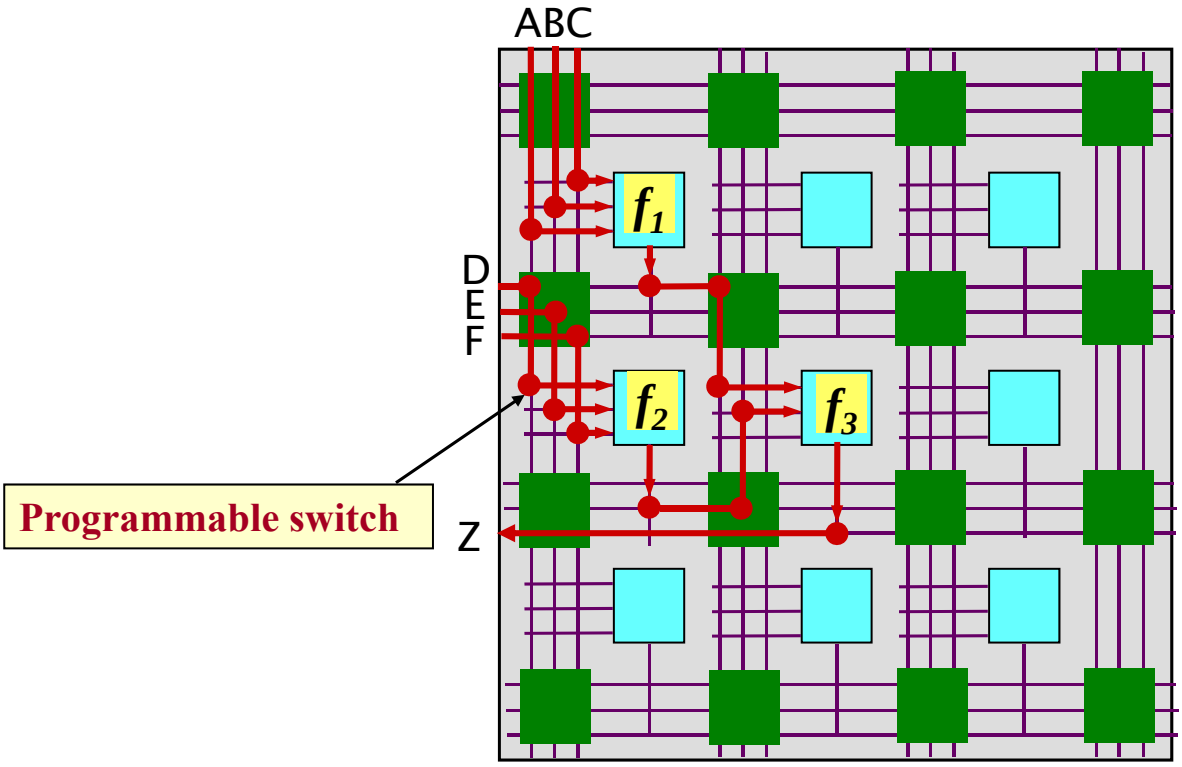
Design Summary

Number of errors: 0
Number of warnings: 0
Logic Utilization:
Number of Slice Flip Flops: 144 out of 4,704 3%
Number of 4 input LUTs: 173 out of 4,704 3%
Logic Distribution:
Number of occupied Slices: 145 out of 2,352 6%
Number of Slices containing only related logic: 145 out of 145 100%
Number of Slices containing unrelated logic: 0 out of 145 0%
*See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs: 210 out of 4,704 4%
Number used as logic: 173
Number used as a route-thru: 5
Number used as 16x1 RAMs: 32
Number of bonded IOBs: 74 out of 176 42%
Number of GCLKs: 1 out of 4 25%
Number of GCLKIOBs: 1 out of 4 25

Placement



Routing

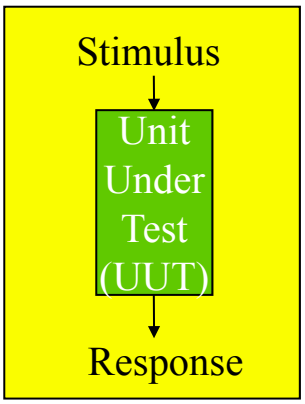


Sample place & route report

```
Timing Score: 0

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

--------------------------------------------------------------------------------
  Constraint                                | Requested  | Actual     | Logic
                                            |            |            | Levels
--------------------------------------------------------------------------------
  TS_clk = PERIOD TIMEGRP "clk" 11.765 ns   | 11.765ns   | 11.622ns   | 13
  HIGH 50%                                  |            |            |
--------------------------------------------------------------------------------
  OFFSET = OUT 11.765 ns AFTER COMP "clk"   | 11.765ns   | 11.491ns   | 1
--------------------------------------------------------------------------------
  OFFSET = IN 11.765 ns BEFORE COMP "clk"   | 11.765ns   | 11.442ns   | 2
--------------------------------------------------------------------------------
```

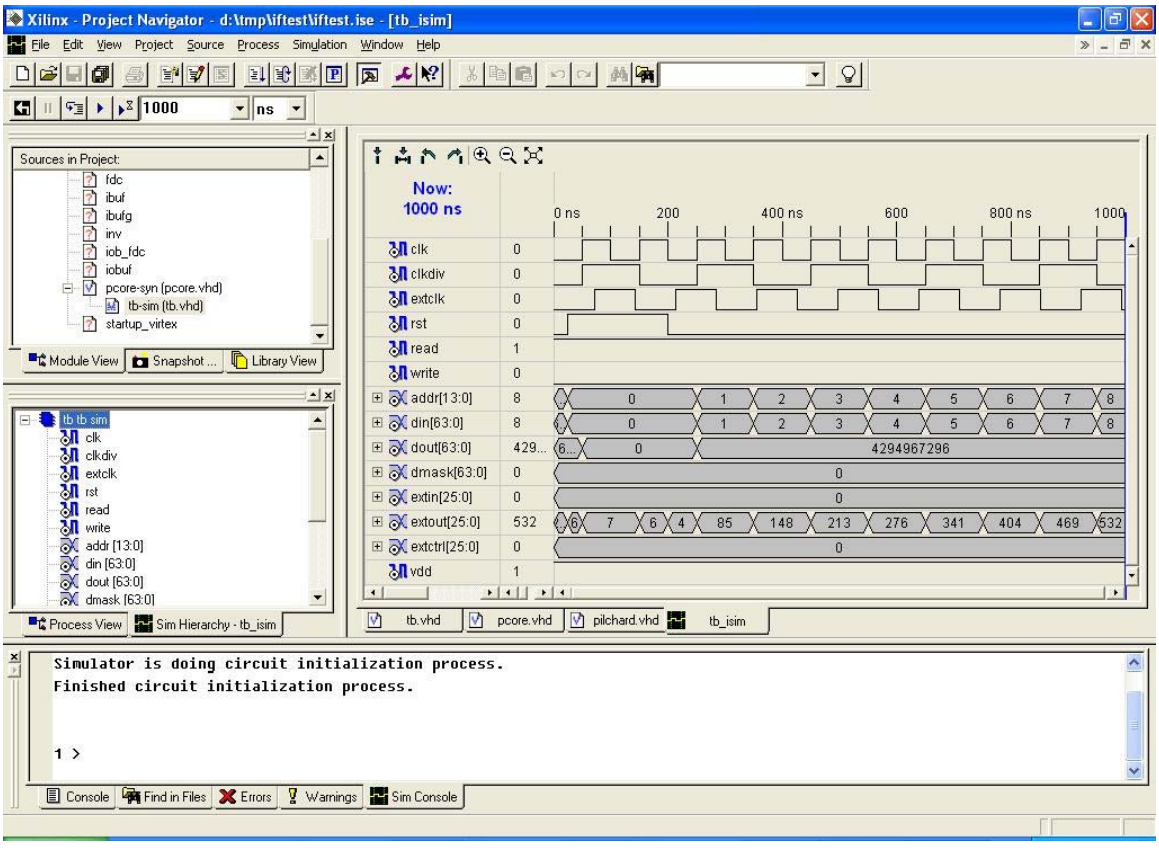
Functional Simulation

- To verify the functionality of a design.
- The user specifies valuations of the circuit's inputs and examines the output of simulation to verify that the circuit operates as expected.
- Functional simulator ignores the logic and interconnect delay.



testbench

Sample functional simulation result



Post-Layout Timing Simulation

- After the physical design tasks are completed, timing simulation is performed to verify the circuit meets the required performance.
- Information from placement & routing can be *back annotated* to the schematic with information on loading and wire delay.
- Timing simulation simulates the actual propagation delays.

Sample post-layout timing report

Timing summary:

Timing errors: 0 Score: 0

Constraints cover 42912 paths, 0 nets, and 1038 connections

Design statistics:

Minimum period: 11.622ns (Maximum frequency: 86.044MHz)

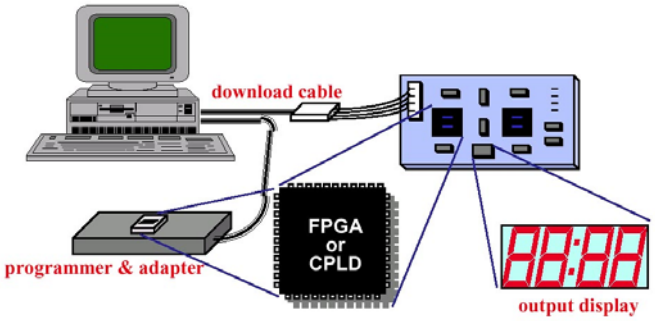
Minimum input required time before clock: 11.442ns

Minimum output required time after clock: 11.491ns

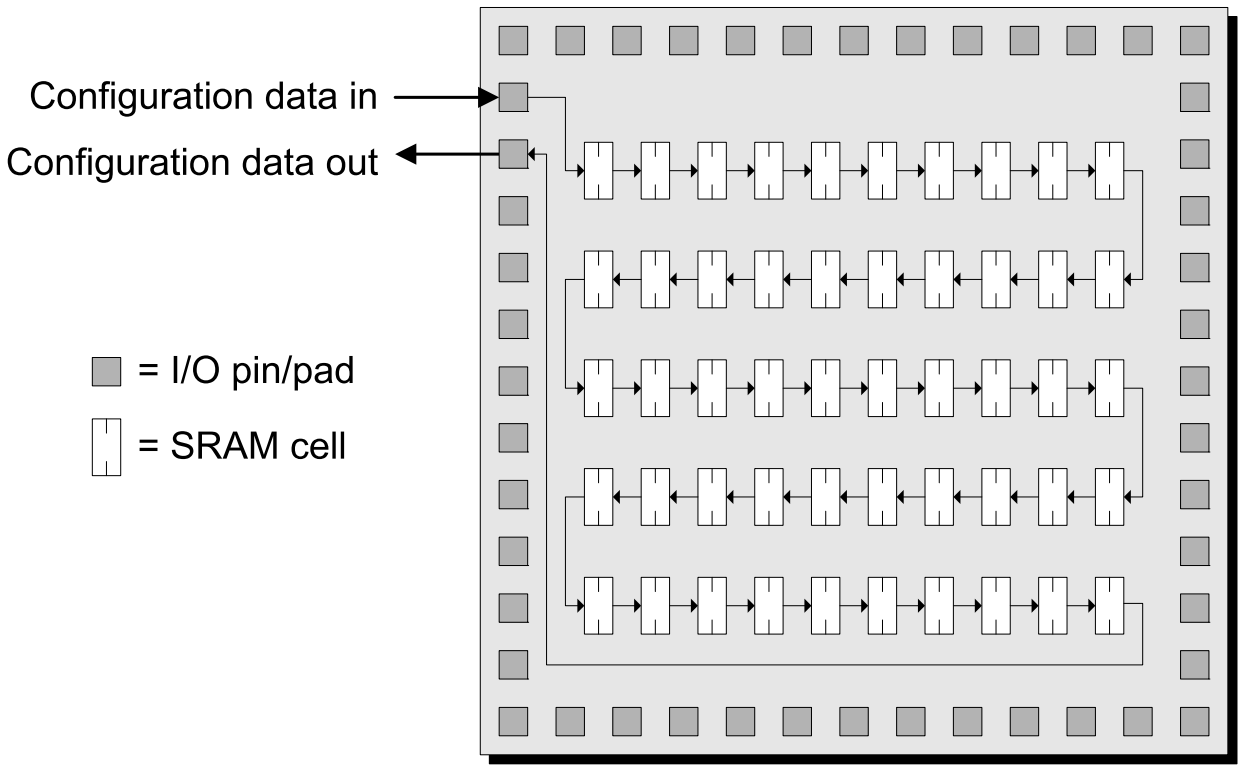
Configuration

- 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

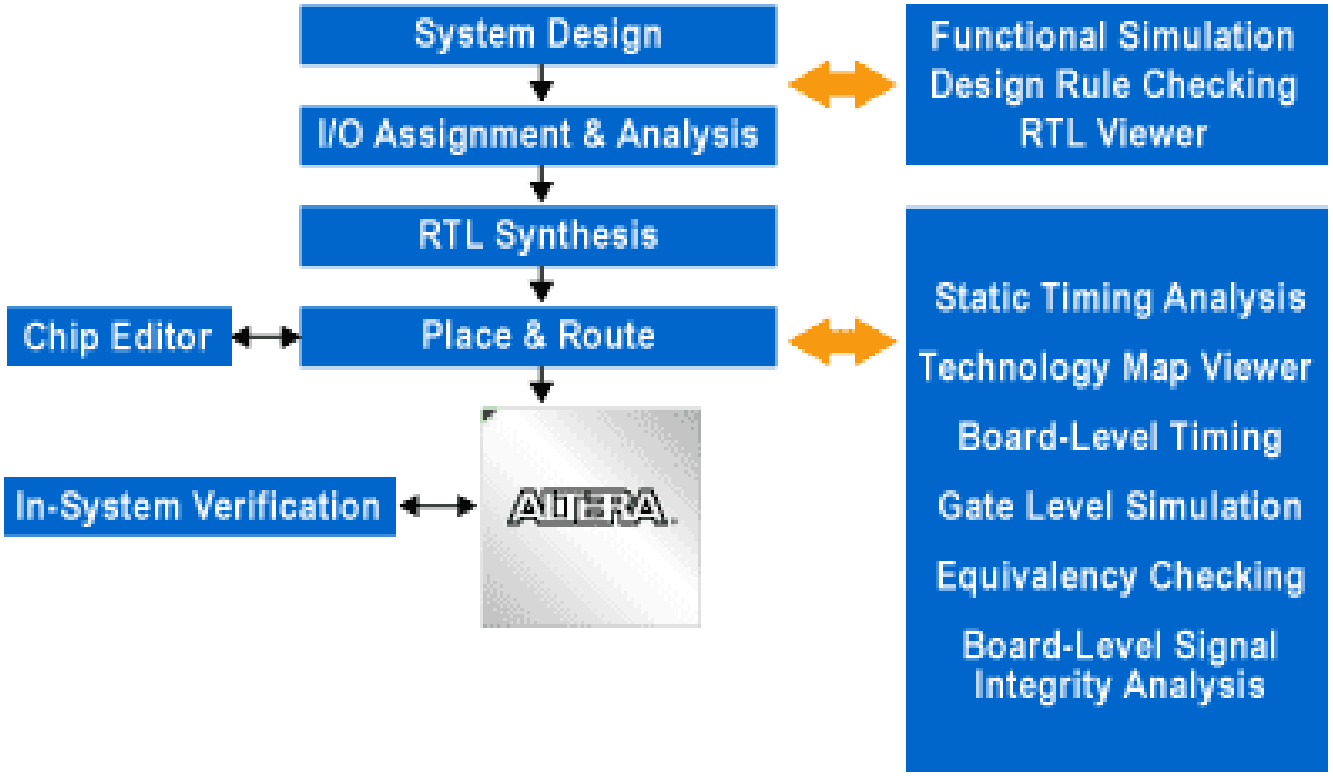


Configuration of SRAM based FPGAs



The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043

FPGA/PCB Co-design Process



Source: Altera