Tips about an FPGA

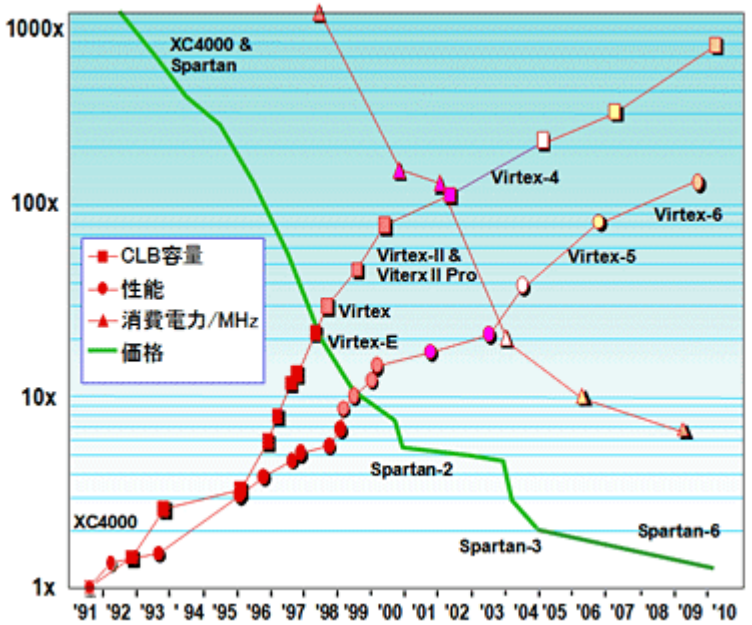
FPGA (field-programmable gate array)

FPGA : An integrated circuit (IC) designed to be configured by customer or designer after manufacturing

-- Developed in ~80s

-- Widely used (customer level) from ~90s

-- Handling digital signal



Spec comparison (number of cells, power consumption, cost)

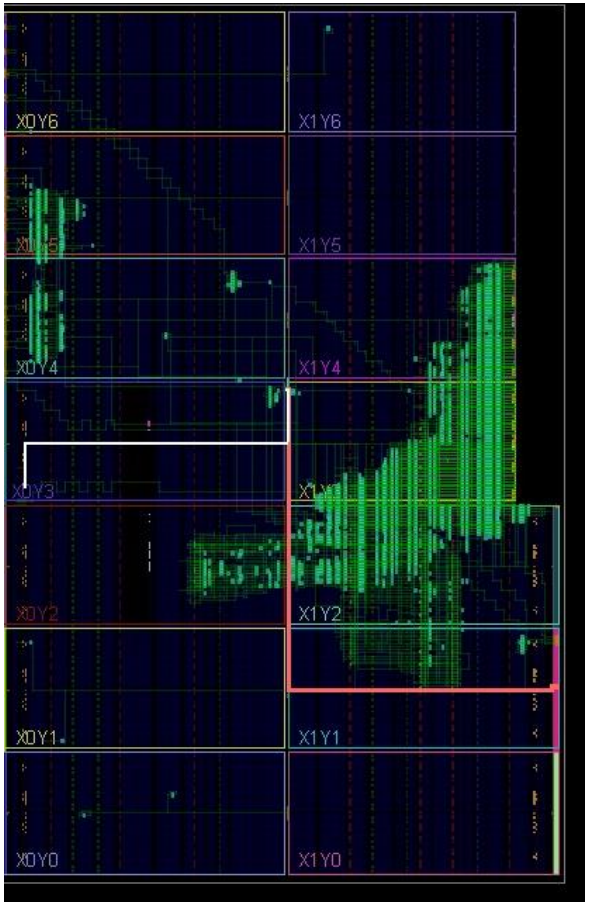
ASIC ?

- Cost per one chip : FPGA > ASIC
  -- but for ASIC, usually, you need to order in unit of lot

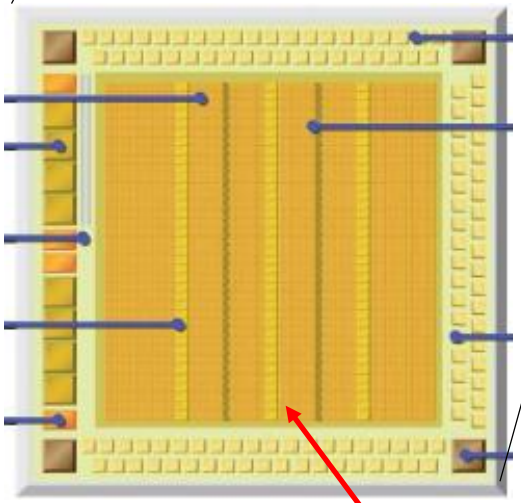
- Initial (development) cost : ASIC >> FPGA
  -- including facilities, capability of reutilization etc.

- Analogue/Degital : FPGA can not deal with analog signal
  (there are already attempts to process analogue signal in FPGA ...)

What they are doing .... ?



After compilation the firmware



In the IC, there are many (huge amount of) cells, and what we have to do is to connect the cells like making a road or route.

HDL (Hardware Description Language)

- Computer language used to described the behavior of IC.

-- The definition of “firmware” is a software which define & control hardwares, therefore, we can often call it “firmware” as well.

- There are many kinds of HDL , like the other programming language. Major ones are “Verilog” and “VHDL” .

So far, I have used “VHDL”, and now I am still learning “Verilog” for KC705.

Just examples from my code

```
55
56      // Memory Address
57      // Address  0 - 7  : Run Control commands ( reset , start, ,,, )
58      // Address  8 - 15 : Data Mode settings I. ( chip address, row/column, ,,, )
59      // Address 16 - 23 : Data Mode settings II. ( suppression mode, timing delay, nframe, ,,, )
60      // Address 24 - 31 : Reserved.
61
62      reg       mem_access_rden  = 1'h0;
63      reg [4:0] mem_address_rden = 5'h0;
64
65      always @( posedge CLK )
66      begin
67          case ( ad_cnt )
68              6'd0  :  mem_address_rden <= 5'd0;    // Memory Address 0  :  Slot for communication test
69              6'd1  :  mem_address_rden <= 5'd1;    // Memory Address 1  : "Reset" command
70              6'd2  :  mem_address_rden <= 5'd2;    // Memory Address 2  : "Initilization" command
71              6'd3  :  mem_address_rden <= 5'd3;    // Memory Address 3  : "Start" command
72              6'd4  :  mem_address_rden <= 5'd4;    // Memory Address 4  : "Stop"  command
73              6'd5  :  mem_address_rden <= 5'd5;    // Memory Address 5  :     reserved
74              6'd6  :  mem_address_rden <= 5'd6;    // Memory Address 6  :     reserved
75              6'd7  :  mem_address_rden <= 5'd7;    // Memory Address 7  :     reserved
76
77              6'd8  :  mem_address_rden <= 5'd8;    // Memory Address 8  : "Chip sector Address"
78              6'd9  :  mem_address_rden <= 5'd9;    // Memory Address 9  :     reserved
79              6'd10 :  mem_address_rden <= 5'd10;   // Memory Address 10 : "row start"
80              6'd11 :  mem_address_rden <= 5'd11;   // Memory Address 11 : "row end"
81              6'd12 :  mem_address_rden <= 5'd12;   // Memory Address 12 : "column start"
82              6'd13 :  mem_address_rden <= 5'd13;   // Memory Address 13 : "column end"
83              6'd14 :  mem_address_rden <= 5'd14;   // Memory Address 14 :     reserved
84              6'd15 :  mem_address_rden <= 5'd15;   // Memory Address 15 :     reserved
```

Verilog

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

entity FIR_CORRECTION_MOD is
        port ( ADC_IN       : in std_logic_vector(9 downto 0);

               FIR_Flag     : in std_logic;

               FIR_PARAM00     : in std_logic_vector(15 downto 0);
               FIR_PARAM01     : in std_logic_vector(15 downto 0);
               FIR_PARAM02     : in std_logic_vector(15 downto 0);
               FIR_PARAM03     : in std_logic_vector(15 downto 0);
               FIR_PARAM04     : in std_logic_vector(15 downto 0);
               FIR_PARAM05     : in std_logic_vector(15 downto 0);
               FIR_PARAM06     : in std_logic_vector(15 downto 0);
               FIR_PARAM07     : in std_logic_vector(15 downto 0);
               FIR_PARAM08     : in std_logic_vector(15 downto 0);
               FIR_PARAM09     : in std_logic_vector(15 downto 0);
               FIR_PARAM10     : in std_logic_vector(15 downto 0);
               FIR_PARAM11     : in std_logic_vector(15 downto 0);
               FIR_PARAM12     : in std_logic_vector(15 downto 0);
               FIR_PARAM13     : in std_logic_vector(15 downto 0);
               FIR_PARAM14     : in std_logic_vector(15 downto 0);
               FIR_PARAM15     : in std_logic_vector(15 downto 0);

               CLK          : in std_logic;
               RESET        : in std_logic;
               ADC_OUT      : out std_logic_vector(9 downto 0) );
end FIR_CORRECTION_MOD;
```

VHDL

New features

- Embedded CPU

  -- Hard CPU : CPU is really embedded inside of FPGA, and can be connected)

  -- Soft CPU : CPU is realized by firmware and act as if it is a CPU. Need resources, but reduce developing cost, if resources are OK.

- Usage of programming language like C++

-- "and" / "or" / "not" and their combination (at the very beggining)

-- HDL (now) ---> behavior description is available.

-- Programming language, such as C++

For me, this is under developing stage.