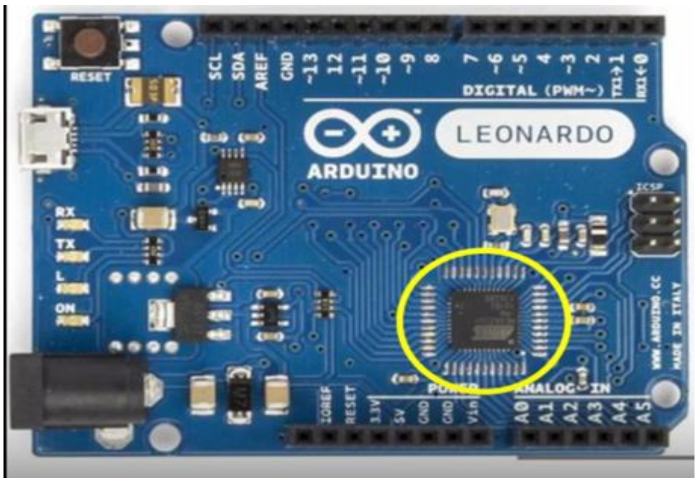
NASSCON – VSD SoC Design Program [24 April – 7 May, 2024] is a two weeks hands on workshop which helps to understand RTL2GDSII flow and SoC design flow.

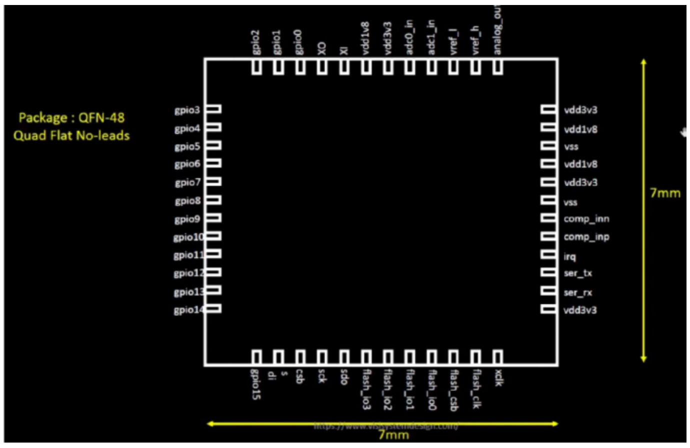
Day 1 - Inception of open-source EDA, OpenLANE and Sky130 PDK

Day 1.1 - SKY130_D1_SK1 - How to talk to computers

In general the entire electronic gadget contains microprocessor or microcontroller as a CPU in the mother board PCB. The following Arduino board contains microcontroller ATmega328P chip as highlighted in the figure. This microcontroller is packaged with QFN-48 with surface mount.



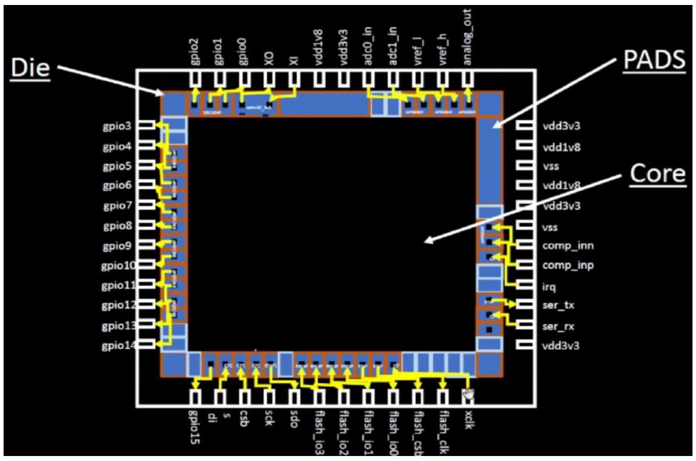
The following diagram shows the dimension of QFN-48 package with 7 mm × 7 mm.



To understand the concept of Die, Pads and Core design is provided in the following diagram.

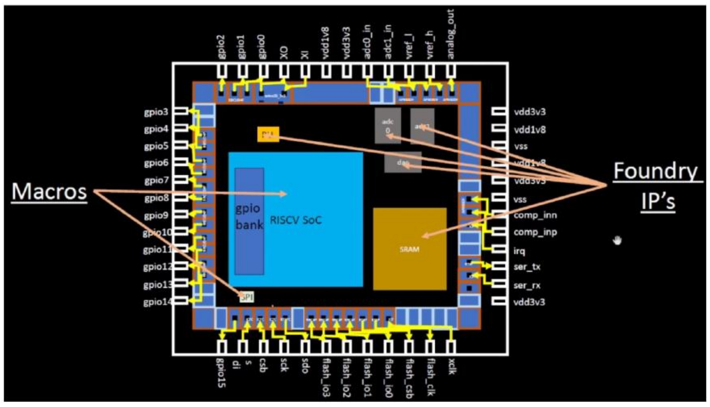
Die – In the Integrated Circuits (IC), the overall functional circuit or core is fabricated in a small dimension.

Pads – The core function need to connect with external world in terms of providing the input and getting the output.

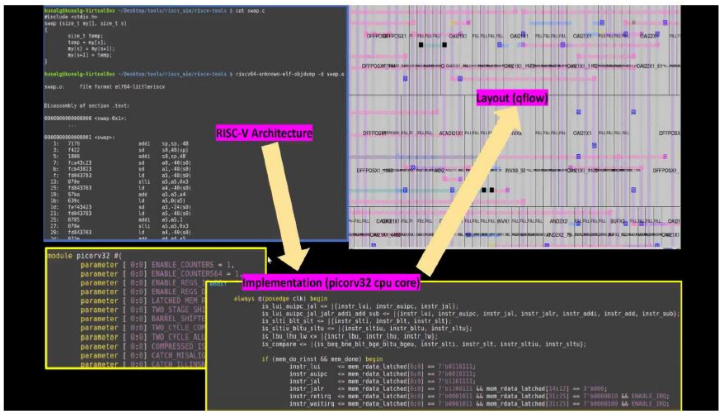


.

Further, the package contains two major modules named as Macros and Foundry IPs. Macros are known as pre-defined circuits with proper verification and in Foundry IPs is design for the required design from the customer for specific applications.



One of the popular Open Standard - Instruction Set Architecture (ISA) is known as RISC-V architecture which comes with free royalty licences. The following figures shows how a C-program (RISC-V architecture) helps to implement Picorv32 CPU core and finally with hardware layout.

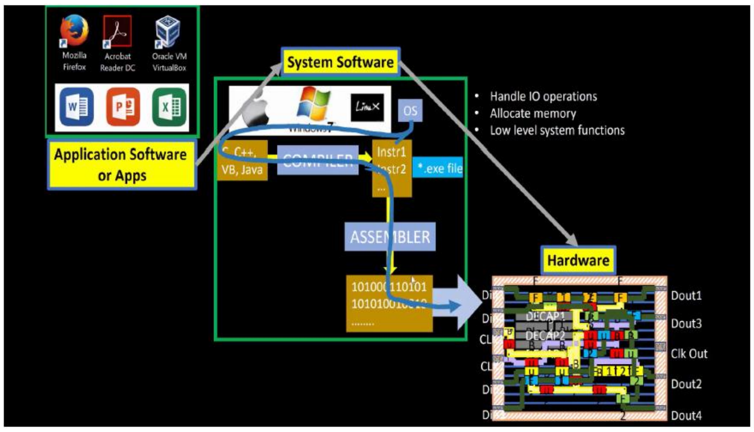


```
kunalg@kunalg-VirtualBox ~/Desktop/tools/riscv_sim/riscv-tools $ cat swap.c
#include <stdio.h>
swap (size_t my[], size_t s)
{
        size_t temp;
        temp = my[s];
        my[s] = my[s+1];
        my[s+1] = temp;
}
kunalg@kunalg-VirtualBox ~/Desktop/tools/riscv_sim/riscv-tools $ riscv64-unknown-elf-objdump -d swap.o

swap.o:     file format elf64-littleriscv


Disassembly of section .text:

0000000000000000 <swap-0x1>:
        ...

0000000000000001 <swap>:
   1:   7179                    addi    sp,sp,-48
   3:   f422                    sd      s0,40(sp)
   5:   1800                    addi    s0,sp,48
   7:   fca43c23                sd      a0,-40(s0)
   b:   fcb43823                sd      a1,-48(s0)
   f:   fd043783                ld      a5,-48(s0)
  13:   078e                    slli    a5,a5,0x3
  15:   fd843703                ld      a4,-40(s0)
  19:   97ba                    add     a5,a5,a4
  1b:   639c                    ld      a5,0(a5)
  1d:   fef43423                sd      a5,-24(s0)
  21:   fd043783                ld      a5,-48(s0)
  25:   0785                    addi    a5,a5,1
  27:   078e                    slli    a5,a5,0x3
  29:   fd843703                ld      a4,-40(s0)
```

```
module picorv32 #(
        parameter [ 0:0] ENABLE_COUNTERS = 1,
        parameter [ 0:0] ENABLE_COUNTERS64 = 1,
        parameter [ 0:0] ENABLE_REGS_1
        parameter [ 0:0] ENABLE_REGS_D
        parameter [ 0:0] LATCHED_MEM_R
        parameter [ 0:0] TWO_STAGE_SHI
        parameter [ 0:0] BARREL_SHIFTE
        parameter [ 0:0] TWO_CYCLE_COM
        parameter [ 0:0] TWO_CYCLE_ALU
        parameter [ 0:0] COMPRESSED_IS
        parameter [ 0:0] CATCH_MISALIG
        parameter [ 0:0] CATCH_ILLINSN
```

```
        always @(posedge clk) begin
                is_lui_auipc_jal <= |{instr_lui, instr_auipc, instr_jal};
                is_lui_auipc_jal_jalr_addi_add_sub <= |{instr_lui, instr_auipc, instr_jal, instr_jalr, instr_addi, instr_add, instr_sub};
                is_slti_blt_slt <= |{instr_slti, instr_blt, instr_slt};
                is_sltiu_bltu_sltu <= |{instr_sltiu, instr_bltu, instr_sltu};
                is_lbu_lhu_lw <= |{instr_lbu, instr_lhu, instr_lw};
                is_compare <= |{is_beq_bne_blt_bge_bltu_bgeu, instr_slti, instr_slt, instr_sltiu, instr_sltu};

                if (mem_do_rinst && mem_done) begin
                        instr_lui     <= mem_rdata_latched[6:0] == 7'b0110111;
                        instr_auipc   <= mem_rdata_latched[6:0] == 7'b0010111;
                        instr_jal     <= mem_rdata_latched[6:0] == 7'b1101111;
                        instr_jalr    <= mem_rdata_latched[6:0] == 7'b1100111 && mem_rdata_latched[14:12] == 3'b000;
                        instr_retirq  <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000010 && ENABLE_IRQ;
                        instr_waitirq <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000100 && ENABLE_IRQ;
```

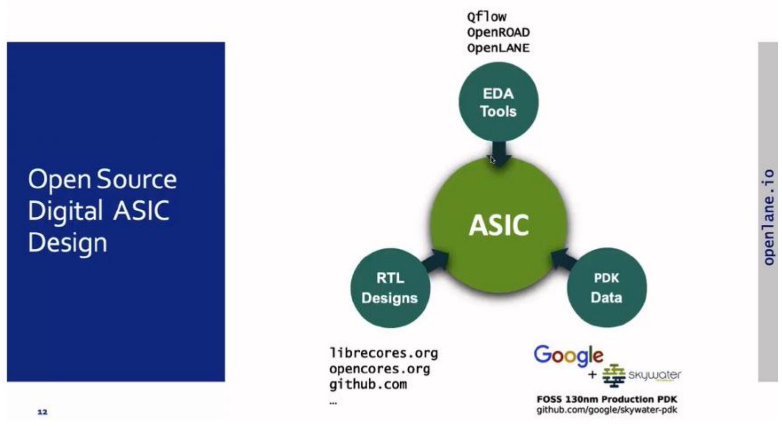
The elaborated flow was given below by highlighting from initial application software stage to hardware layout with various conversion stages. In the system software, the C / C++ code was compiled and generated assembly code, then with help of assembler, the final binary code was generated and given as a input to hardware design layout.
In summary, initial process starts with ISA, then RTL Synthesis (CPU- Picorv32) and finally Physical design (layout).



DAY 1.1 - SKY130_D1_SK2 - SoC design and OpenLANE

SKY_L1 - Introduction to all components of open-source digital ASIC design

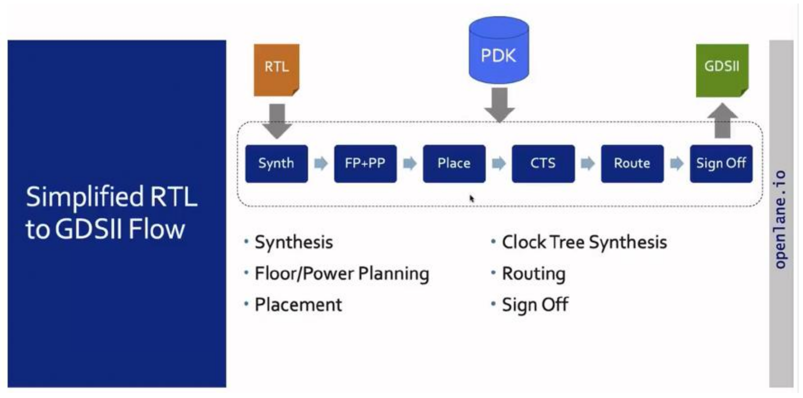
OpenlLANE is a popular environment, which integrates various EDA tools to support from RTL to GDSII form. This arrangement integrates three major domains known as EDA Tools (Qflow, OpenROAD), PDK Data (Skywater130nm) and RTL Design (Github.com) to achieve efficient ASIC flow.



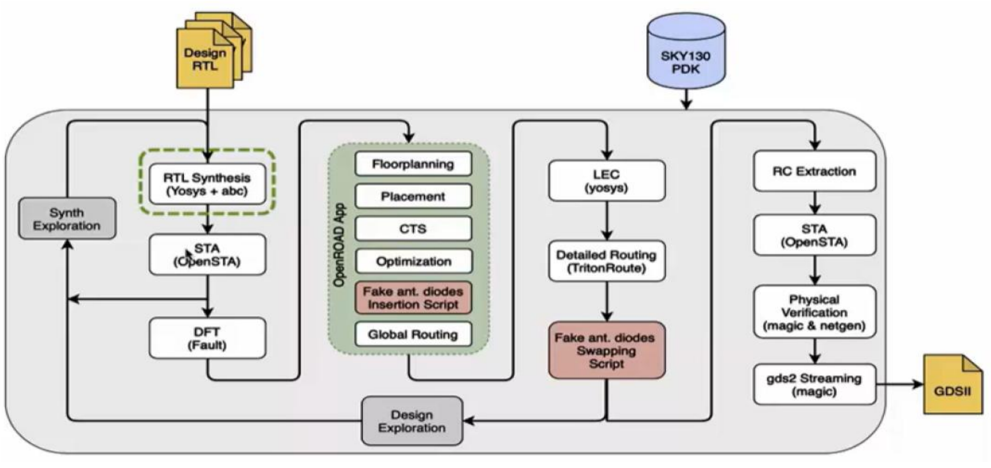
SKY_L2 - Simplified RTL2GDS flow

RTL2GDSII flow has lot of technical stages known as Synthesis, Floor Planning, Placement, Clock Tree Synthesis, Routing and Final Sign off.

Synthesis – It is the process of converting RTL code (Verilog / VHDL) into required Standard Cell Libraries (SCL) and it is known as Netlist of the design.
Floor Planning – Next stage of synthesis is known as floor planning, which allocates different design block and connect with proper pads. Macro floor planning and power planning are considered as very important processing of floor plane.
Placement – This process helps for better placement of the cells on the proper floor plan and aligned with the availability design space. This process is considered for global and detailed placement.
Clock Tree Synthesis – It helps to deliver the master clock to all the required sequential blocks like flip flops in terms of Tree structure.
Routing – This process is used to interconnect the design with various metal layers as per the requirement and availability.
Sign Off – This stage contains Physical verification (DRC and LVS) and Timing verification (STA)



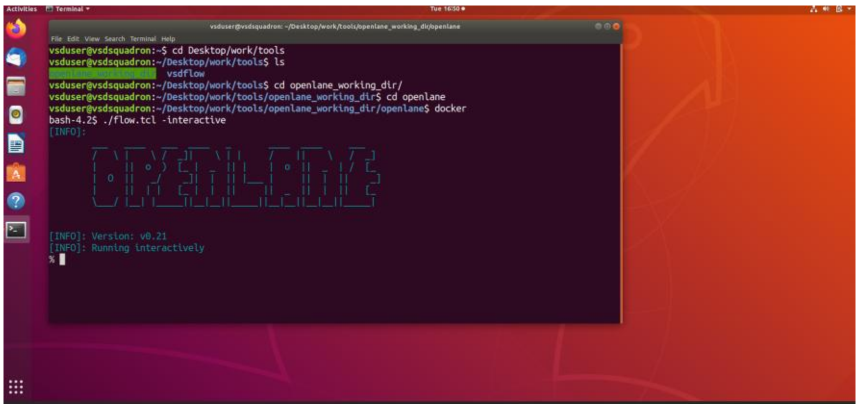
OpenLANE ASIC flow is contains various stages of design process. They are Synthesis Exploration, Regression Testing, Design for Test (DFT), Physical implementation, Logic Equivalence Check (LEC), Dealing with antenna rules violations, Static timing analysis (STA), and Physical verification DRC & LVS,



Day 1.1 - SKY130_D1_SK3 - Get familiar to open-source EDA tools

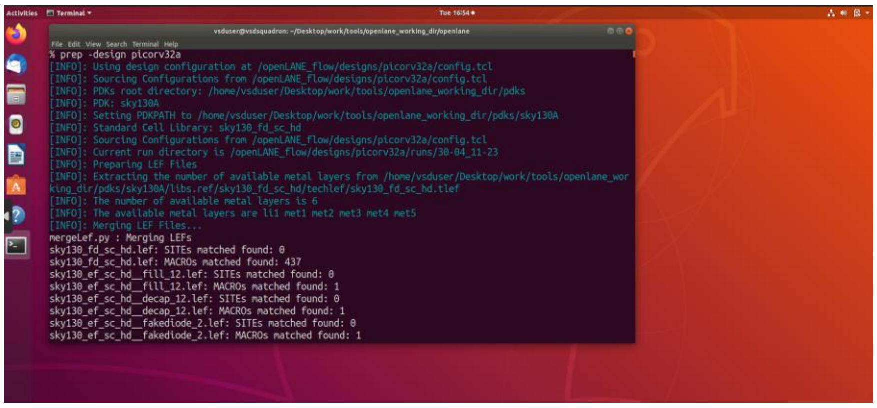
The following commends are used to access OpenLANE environment.

- ➢ cd Desktop/work/tools/openlane_working_dir/openlane
- ➢ docker
- ➢ bash-4.2$ ./flow.tcl –interactive



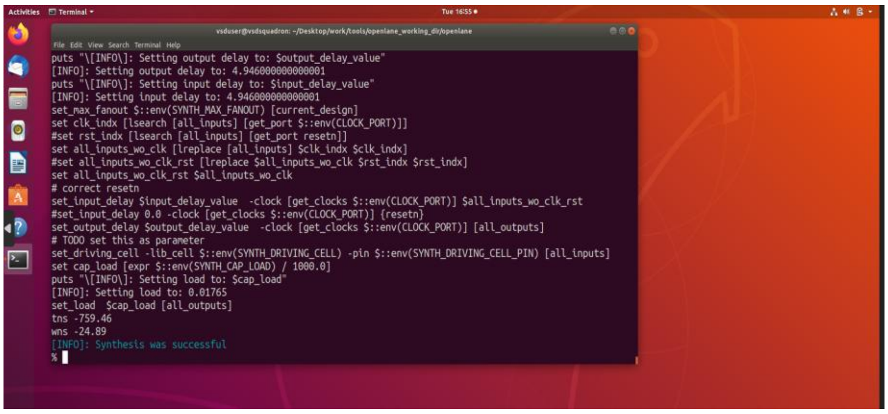
```
vsduser@vsdsquadron:~$ cd Desktop/work/tools
vsduser@vsdsquadron:~/Desktop/work/tools$ ls
openlane_working_dir  vsdflow
vsduser@vsdsquadron:~/Desktop/work/tools$ cd openlane_working_dir/
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir$ cd openlane
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane$ docker
bash-4.2$ ./flow.tcl -interactive
[INFO]:

[INFO]: Version: v0.21
[INFO]: Running interactively
%
```

- ➢ package require openlane
- ➢ 0.9
- ➢ prep –design picorv32a



```
% prep -design picorv32a
[INFO]: Using design configuration at /openLANE_flow/designs/picorv32a/config.tcl
[INFO]: Sourcing Configurations from /openLANE_flow/designs/picorv32a/config.tcl
[INFO]: PDKs root directory: /home/vsduser/Desktop/work/tools/openlane_working_dir/pdks
[INFO]: PDK: sky130A
[INFO]: Setting PDKPATH to /home/vsduser/Desktop/work/tools/openlane_working_dir/pdks/sky130A
[INFO]: Standard Cell Library: sky130_fd_sc_hd
[INFO]: Sourcing Configurations from /openLANE_flow/designs/picorv32a/config.tcl
[INFO]: Current run directory is /openLANE_flow/designs/picorv32a/runs/30-04_11-23
[INFO]: Preparing LEF Files
[INFO]: Extracting the number of available metal layers from /home/vsduser/Desktop/work/tools/openlane_wor
king_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd/techlef/sky130_fd_sc_hd.tlef
[INFO]: The number of available metal layers is 6
[INFO]: The available metal layers are li1 met1 met2 met3 met4 met5
[INFO]: Merging LEF Files...
mergeLef.py : Merging LEFs
sky130_fd_sc_hd.lef: SITEs matched found: 0
sky130_fd_sc_hd.lef: MACROs matched found: 437
sky130_ef_sc_hd__fill_12.lef: SITEs matched found: 0
sky130_ef_sc_hd__fill_12.lef: MACROs matched found: 1
sky130_ef_sc_hd__decap_12.lef: SITEs matched found: 0
sky130_ef_sc_hd__decap_12.lef: MACROs matched found: 1
sky130_ef_sc_hd__fakediode_2.lef: SITEs matched found: 0
sky130_ef_sc_hd__fakediode_2.lef: MACROs matched found: 1
```

- ➢ run_synthesis



Activities
Terminal
Tue 16:55
vsduser@vsdsquadron: ~/Desktop/work/tools/openlane_working_dir/openlane
File Edit View Search Terminal Help

```
puts "\[INFO\]: Setting output delay to: $output_delay_value"
[INFO]: Setting output delay to: 4.946000000000001
puts "\[INFO\]: Setting input delay to: $input_delay_value"
[INFO]: Setting input delay to: 4.946000000000001
set_max_fanout $::env(SYNTH_MAX_FANOUT) [current_design]
set clk_indx [lsearch [all_inputs] [get_port $::env(CLOCK_PORT)]]
#set rst_indx [lsearch [all_inputs] [get_port resetn]]
set all_inputs_wo_clk [lreplace [all_inputs] $clk_indx $clk_indx]
#set all_inputs_wo_clk_rst [lreplace $all_inputs_wo_clk $rst_indx $rst_indx]
set all_inputs_wo_clk_rst $all_inputs_wo_clk
# correct resetn
set_input_delay $input_delay_value  -clock [get_clocks $::env(CLOCK_PORT)] $all_inputs_wo_clk_rst
#set_input_delay 0.0 -clock [get_clocks $::env(CLOCK_PORT)] {resetn}
set_output_delay $output_delay_value  -clock [get_clocks $::env(CLOCK_PORT)] [all_outputs]
# TODO set this as parameter
set_driving_cell -lib_cell $::env(SYNTH_DRIVING_CELL) -pin $::env(SYNTH_DRIVING_CELL_PIN) [all_inputs]
set cap_load [expr $::env(SYNTH_CAP_LOAD) / 1000.0]
puts "\[INFO\]: Setting load to: $cap_load"
[INFO]: Setting load to: 0.01765
set_load  $cap_load [all_outputs]
tns -759.46
wns -24.89
[INFO]: Synthesis was successful
% 
```

- From the synthesis report, observe the number of cells and DFF counts and calculate flop rate and percentage of FF's

$$Flop\ Ratio = \frac{1613}{14876} = 0.108429685$$

$$Percentage\ of\ DFF's = 0.108429685 * 100 = 10.84296854\ \%$$