

18LV52 – ADVANCED VLSI DESIGN LABORATORY

Kumar S (20MV03)

John Samuel Ebenezer A(20MV33)

Ragul M(20MV34)

LAB EXPERIMENT IV

Report submitted in partial fulfilment of the requirements for the
degree of

MASTER OF ENGINEERING

**Branch: ELECTRONICS AND COMMUNICATION
ENGINEERING**

Of Anna University



AUGUST 2021

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641004

Contents

Aim:	3
Design Description:	3
Verilog Code	29
PDK Details	100
Logic Synthesis Tool Flow	106
Yosys flow	106
OpenLane automated flow	108
OpenLane Interactive flow	110
Scripts for OpenLane flow	117
Generation of SERV Utilization and Area Report	129
Generation of Power Report	148
Generation of Timing Report	150
Maximum Operating Frequency Calculations	162
Schematic of Synthesized Netlist	163
Synthesized Netlist File	164
Layout Generation by Magic Tool	186
Final GDSII layout	198
Inferences	202

RTL TO GDSII FLOW using OpenLane

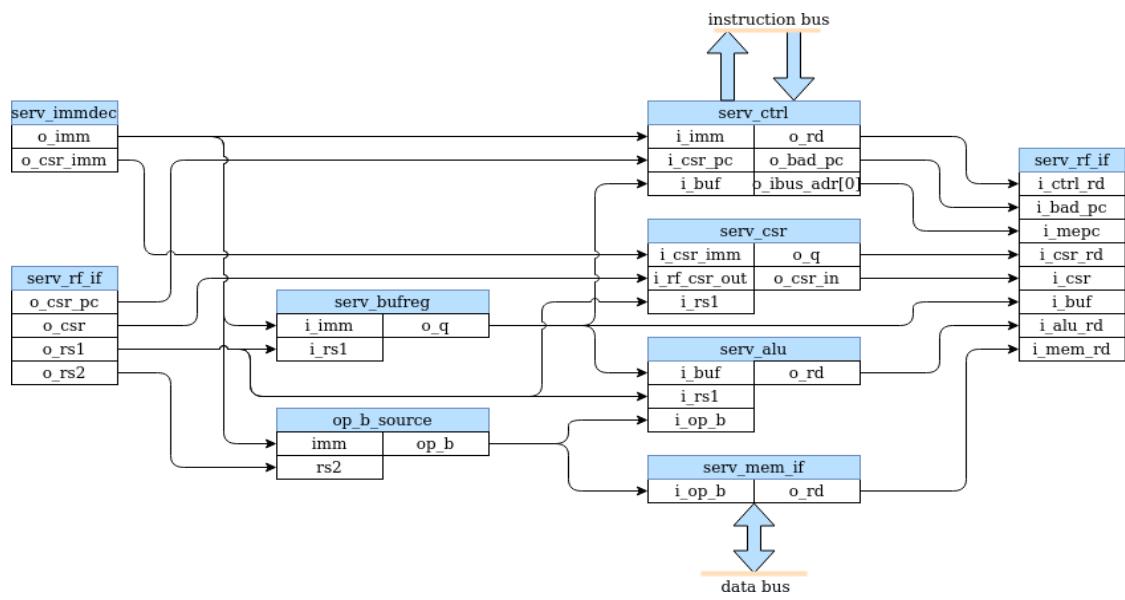
Aim:

To design and synthesize a customized design for

- Logic Synthesis Tool Flow using Yosys
- Obtaining the synthesized netlist along with block diagram after the flow.
- Performing Physical Design from RTL to GDS2 using OpenLane
- Generation of Area, Power and Timing report for the whole flow

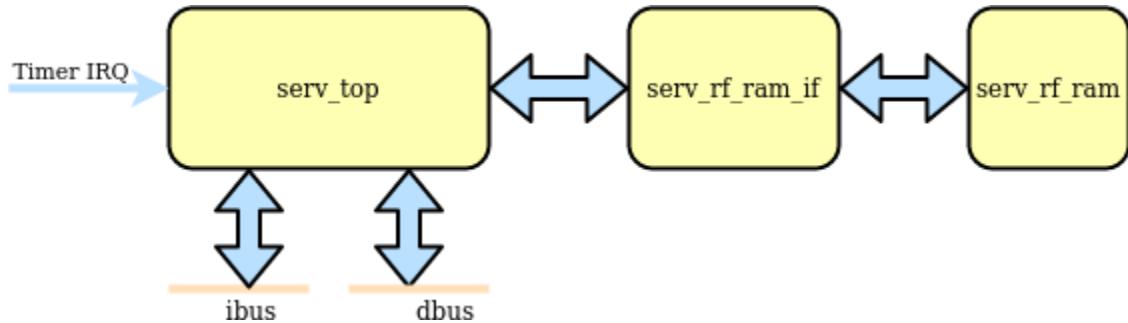
Design Description:

SERV is a bit-serial CPU which means that the internal datapath is one bit wide. SERV internal dataflow show the internal dataflow. For each instruction, data is read from the register file or the immediate fields of the instruction word and the result of the operation is stored back into the register file. Reading and writing memory is handled through the memory interface module.



SERV internal dataflow

serv_rf_top

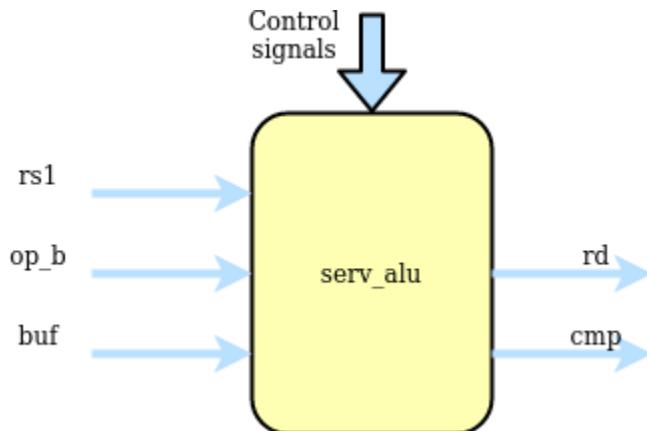


`serv_rf_top` is a top-level convenience wrapper that includes SERV and the default RF implementation and just exposes the timer IRQ and instruction/data wishbone buses.

serv_top

`serv_top` is the top-level of the SERV core without an RF

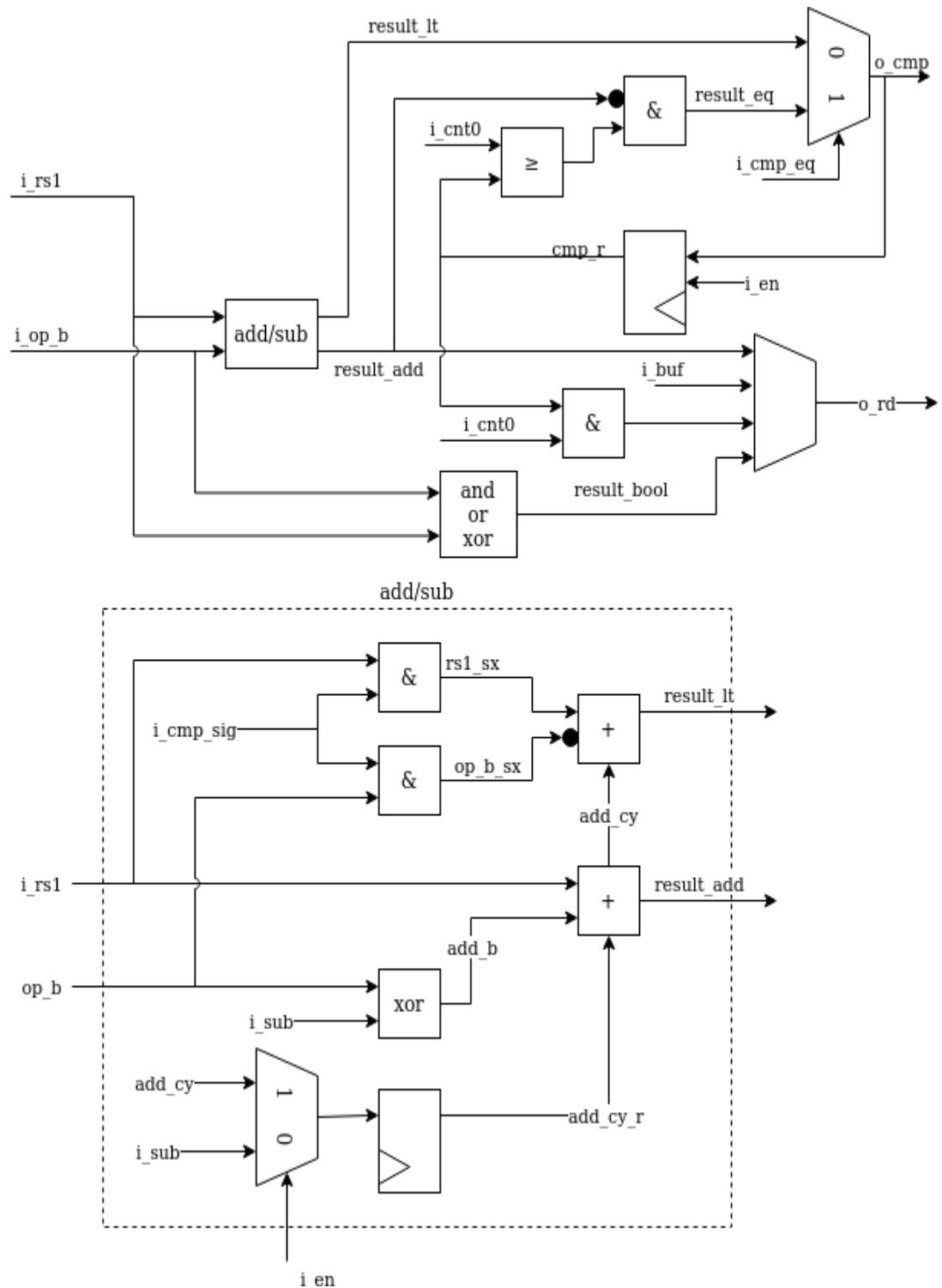
serv_alu



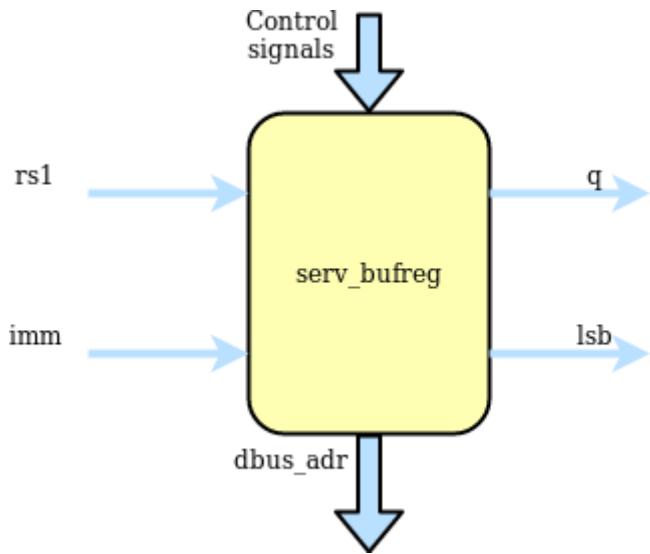
`serv_alu` handles alu operations. The first input operand (A) comes from `i_rs1` and the second operand (B) comes from `i_rs2` or `i_imm` depending on the type of operation. The data passes through the add/sub or bool logic unit and finally ends up in `o_rd` to be written to the destination register. The output

`o_cmp` is used for conditional branches to decide whether or not to take the branch.

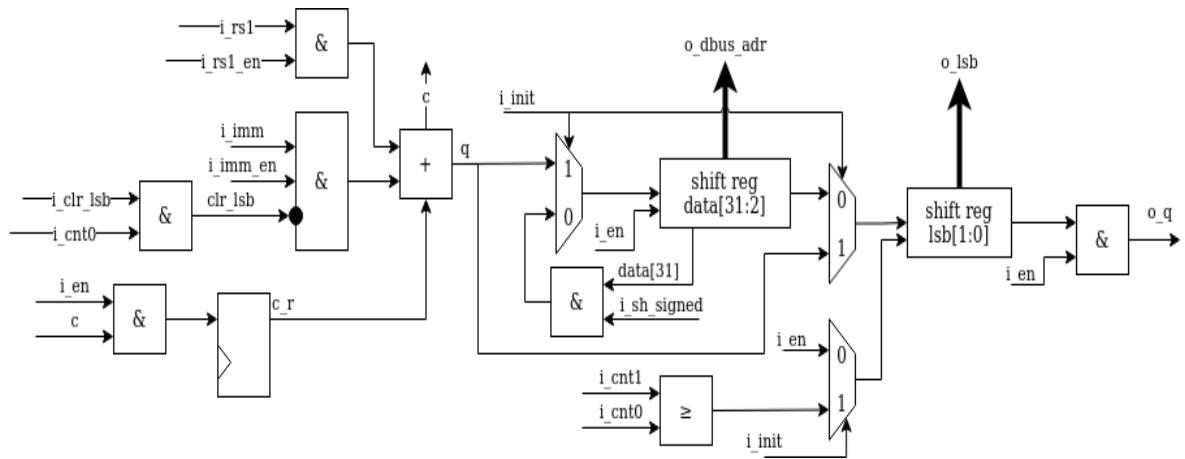
The add/sub unit can do additions $A+B$ or subtractions $A-B$ by converting it to $A+\bar{B}+1$. Subtraction mode (`i_sub = 1`) is also used for the comparisions in the `slt*` and conditional branch instructions. The $+1$ used in subtraction mode is done by preloading the carry input with 1. Less-than comparisons are handled by converting the expression $A < B$ to $A-B < 0$ and checking the MSB, which will be set when the result is less than 0. This however requires sign-extending the operands to 33-bit inputs. For signed operands (when `i_cmp_sig` is set), the extra bit is the same as the MSB. For unsigned, the extra bit is always 0. Because the ALU is only active for 32 cycles, the 33rd bit must be calculated in parallel to the ordinary addition. The result from this operations is available in `result_lt`. For equality checks, `result_eq` checks that all bits are 0 from the subtraction.



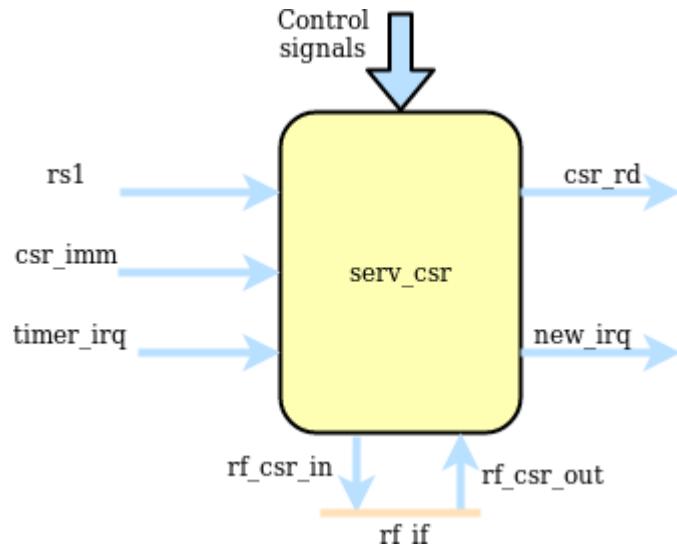
serv_bufreg



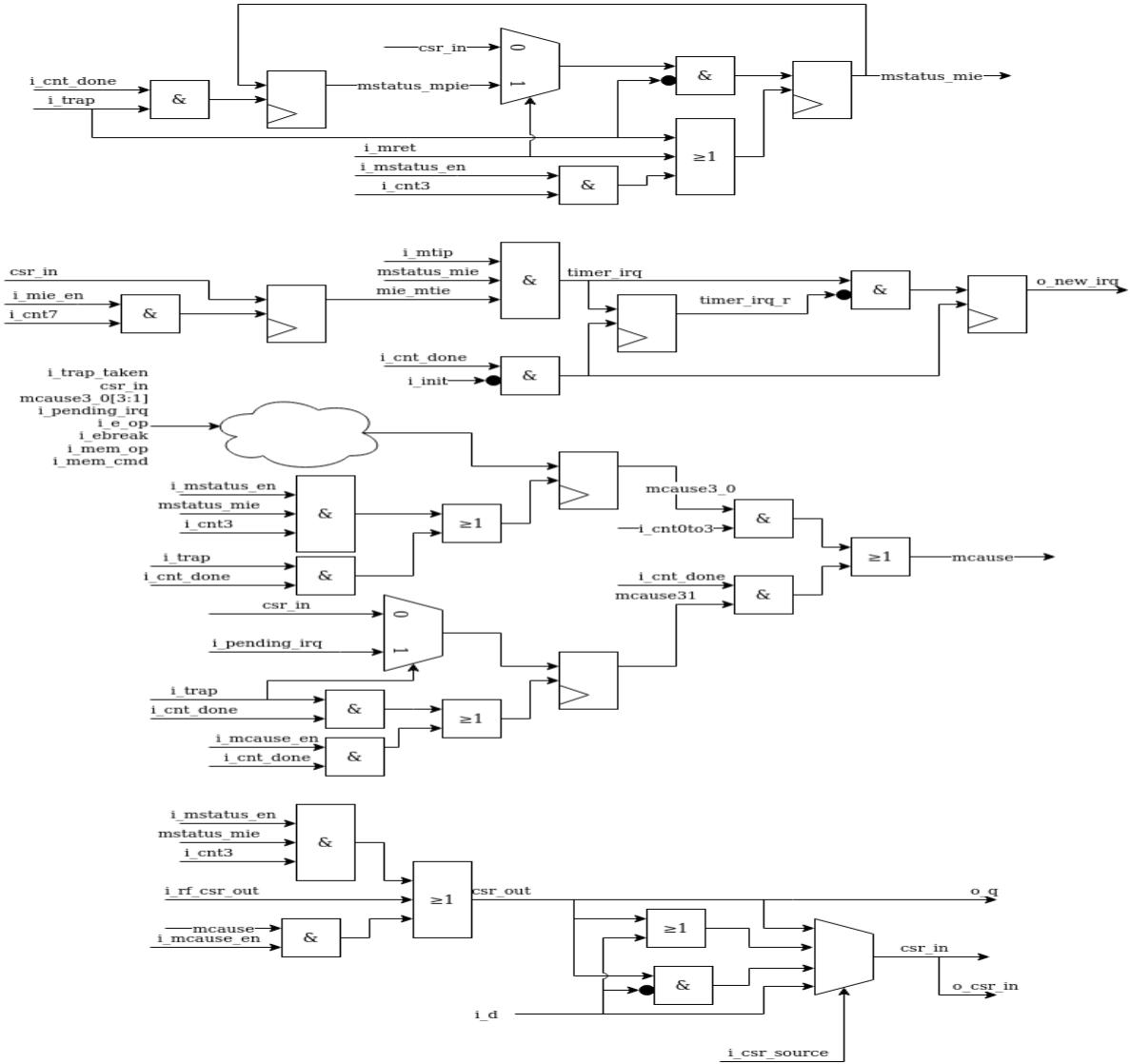
For two-stage operations, `serv_bufreg` holds data between stages. This data can be the effective address for branches or load/stores or data to be shifted for shift ops. It has a serial output for streaming out results during stage two and a parallel output that forms the `dbus` address. `serv_bufreg` also keeps track of the two `lsb` when calculating addresses. This is used to check for alignment errors. In order to support these different modes, the input to the shift register can come from `rs1`, the immediate (`imm`), `rs1+imm` or looped back from the shift register output. The latter is used for shift operations. For some operations, the LSB of the immediate is cleared before written to the shift register. The two LSB of the shift register are special. When the shift register is loaded, these two get written first before the rest of the register is filled up. This allows the memory interface to check data/address alignment early.



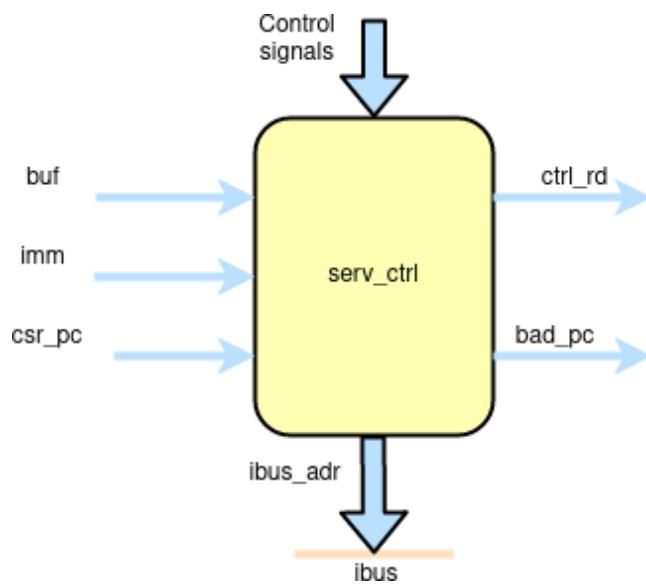
serv_CSR



serv_CSR handles CSR accesses and all status related to (timer) interrupts. Out of the eight CSRs supported by SERV, only four resides in serv_CSR (mstatus, mie, mcause and mip) and for those registers, SERV only implement the bits required for ecall, ebreak, misalignment and timer interrupts. The four remaining CSRs are commonly stored in the RF



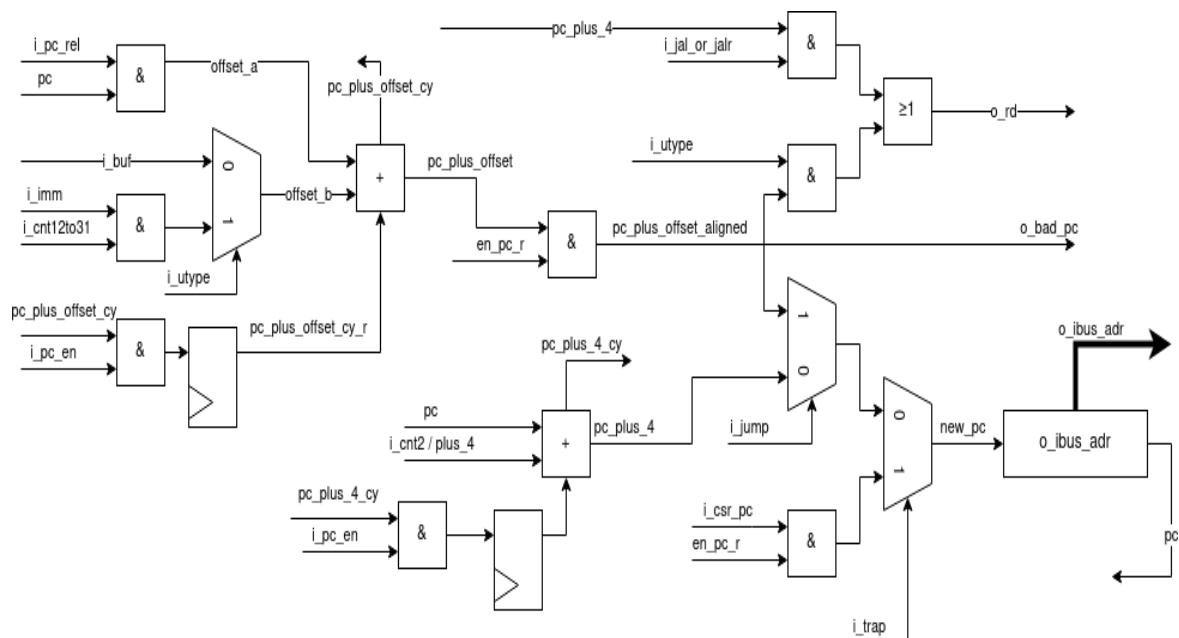
serv_ctrl



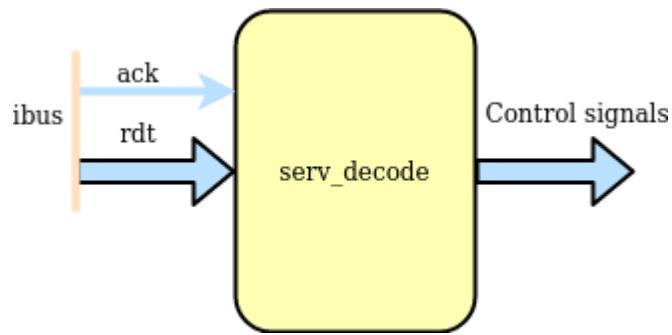
`serv_ctrl` keeps track of the current PC and contains the logic needed to calculate the next PC. The PC is stored in shift register with a parallel output connected to the instruction bus.

The new PC can come from three sources. For normal instructions, it is incremented by four, which is the next 32-bit address. Jumps can be absolute or relative to the current PC. Absolute jumps are precalculated in `serv_bufreg` and written directly to the PC. PC relative jumps have the offset part precalculated in `serv_bufreg` which gets added to the current PC before storing as the new PC. The third source for the new PC comes from the CSR registers when entering or returning traps.

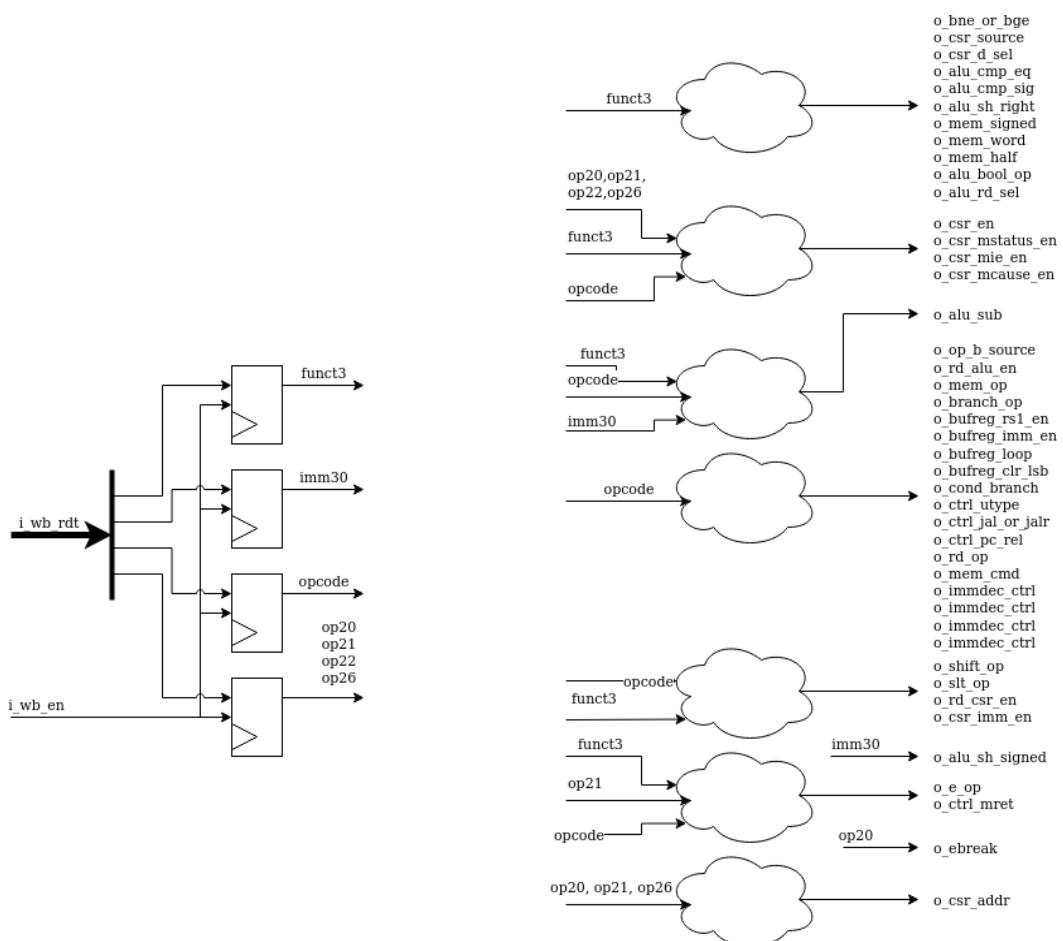
Some operations (LUI, AUIPC, jumps, entering or returning from traps) also update the destination register through `o_rd`. In the case of misaligned instruction traps, the invalid PC is written to `o_bad_pc` to be passed to `mtval`.



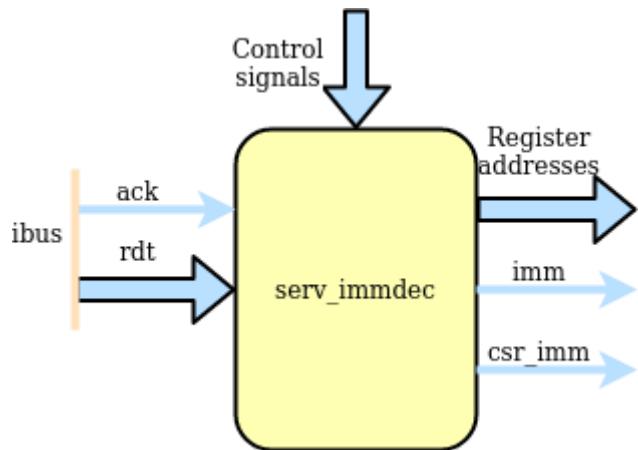
`serv_decode`



`serv_decode` is responsible for decoding the operation word coming from `ibus` into a set of control signals that are used internally in SERV.



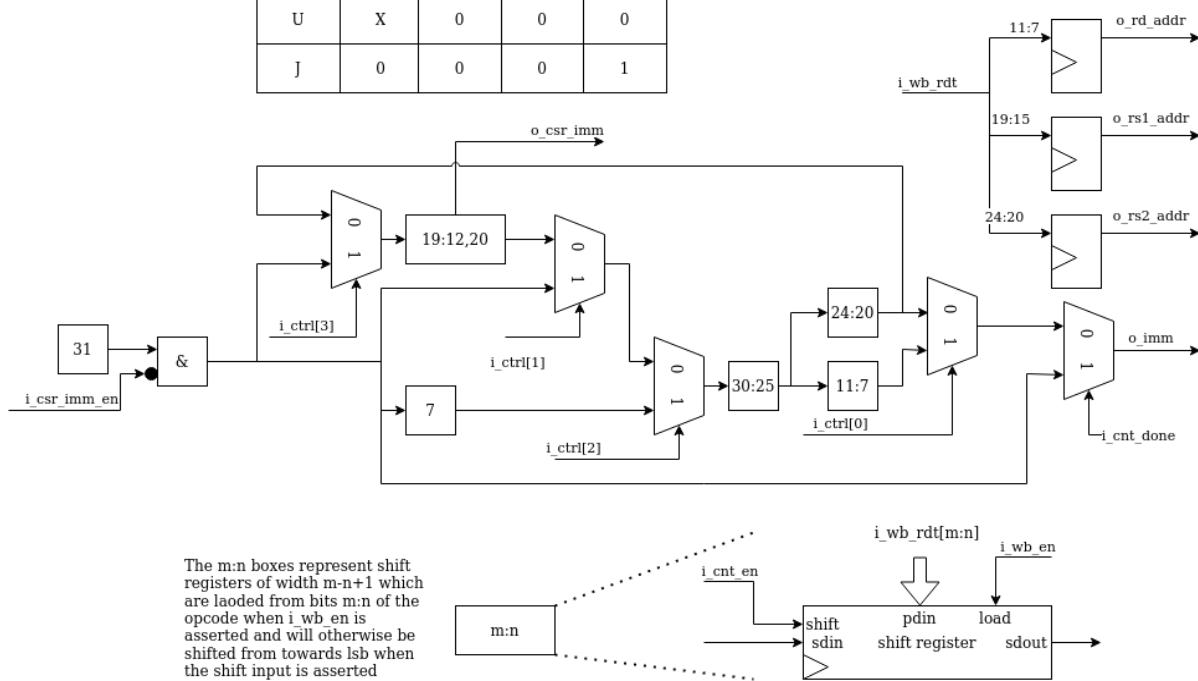
`serv_immddec`



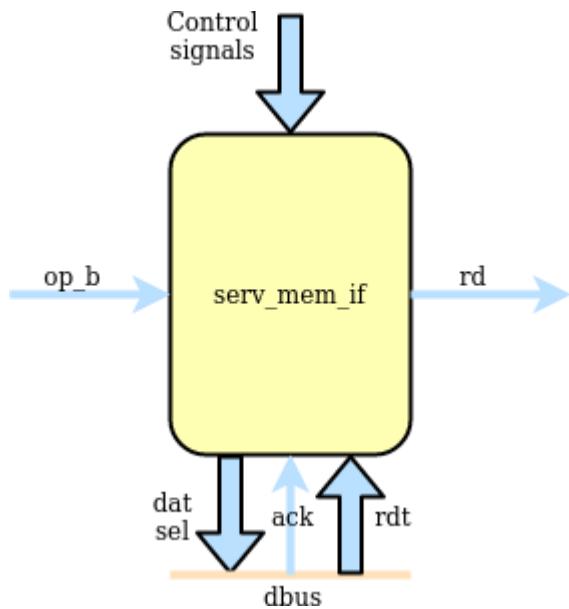
The main responsibility of `serv_immdec` is to stitch together the pieces of immediates from the instruction word and push it out in the correct order. When a new instruction arrives, the relevant parts are placed into a number of shift registers, and the connections between the registers are setup differently depending on the type of operation.

`serv_immdec` also extracts the register addresses from the operation word.

Optype	ctrl[0]	ctrl[1]	ctrl[2]	ctrl[3]
I	0	1	0	X
S	1	1	0	X
B	1	X	1	X
U	X	0	0	0
J	0	0	0	1



serv_mem_if



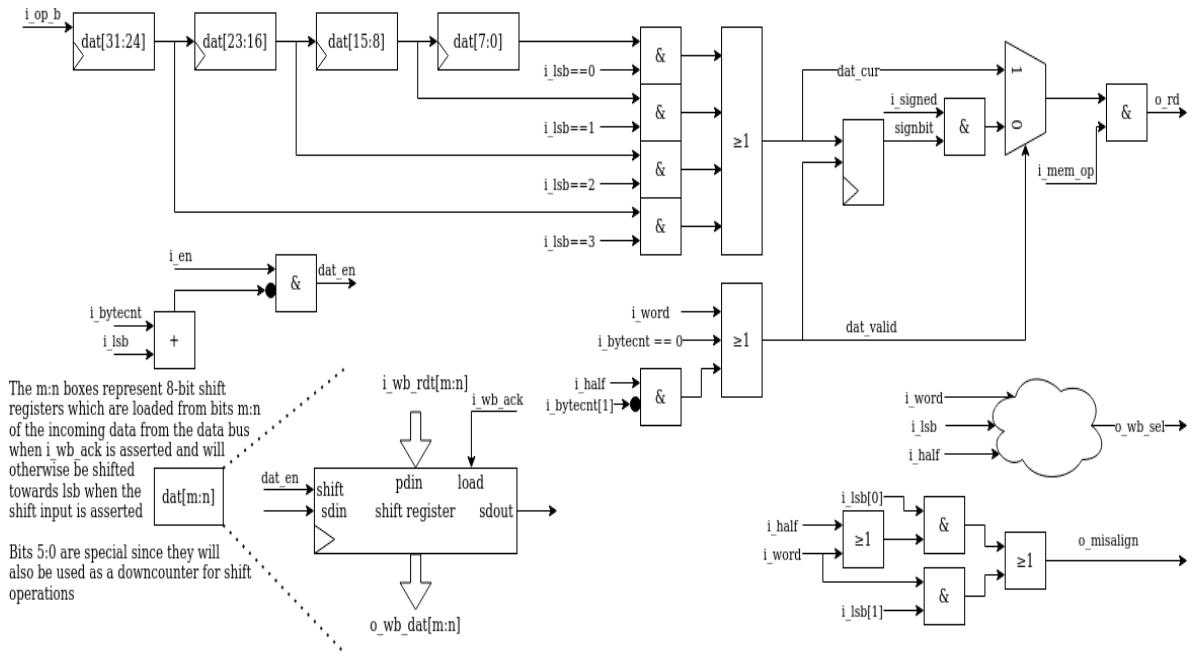
serv_mem_if prepares the data to be sent out on the *dbus* during store operations and serializes the incoming data during loads

The memory interface is centered around four byte-wide shift registers connected in series. During store operations, the *dat_en* signal is asserted long enough to shift in the data from rs2 to the right place in the shift registers and the parallel output of the shift registers is then presented to the data bus as a 32-bit word together with a byte mask. The [Data bus byte mask](#) table summarizes the logic for when the individual byte select signals are asserted depending on the two LSB of the data address together with the size (byte, halfword, word) of the write operation.

During load operations, the data from the bus is latched into the shift registers. *dat_en* is again asserted to shift out data from the registers. *i_lsb* decides from which byte stage of the shift register to tap the data, depending on the alignment of the received data. The *dat_valid* signal makes sure to only present valid data to *o_rd* and otherwise fill in with zeros or sign extension.

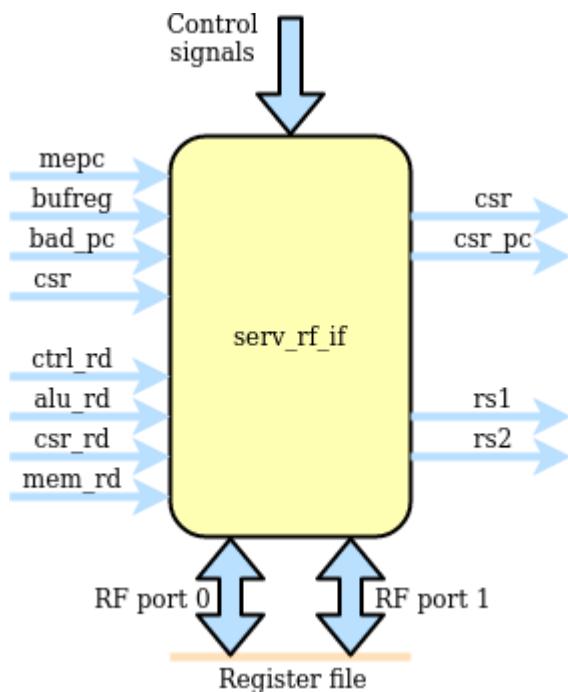
When SERV is built with *WITH_CSR*, there is also logic to detect misaligned accesses which asserts the *o_misalign* flag to the core.

The shift register used for stores and loads are also used to keep track of the number of steps to shift for left/right shift operations. In this mode, the six LSB of the register is loaded with the shift amount during the init stage and the used as a down counter which raises *o_sh_done* and *o_sh_done_r* when the number of shift steps have been completed.

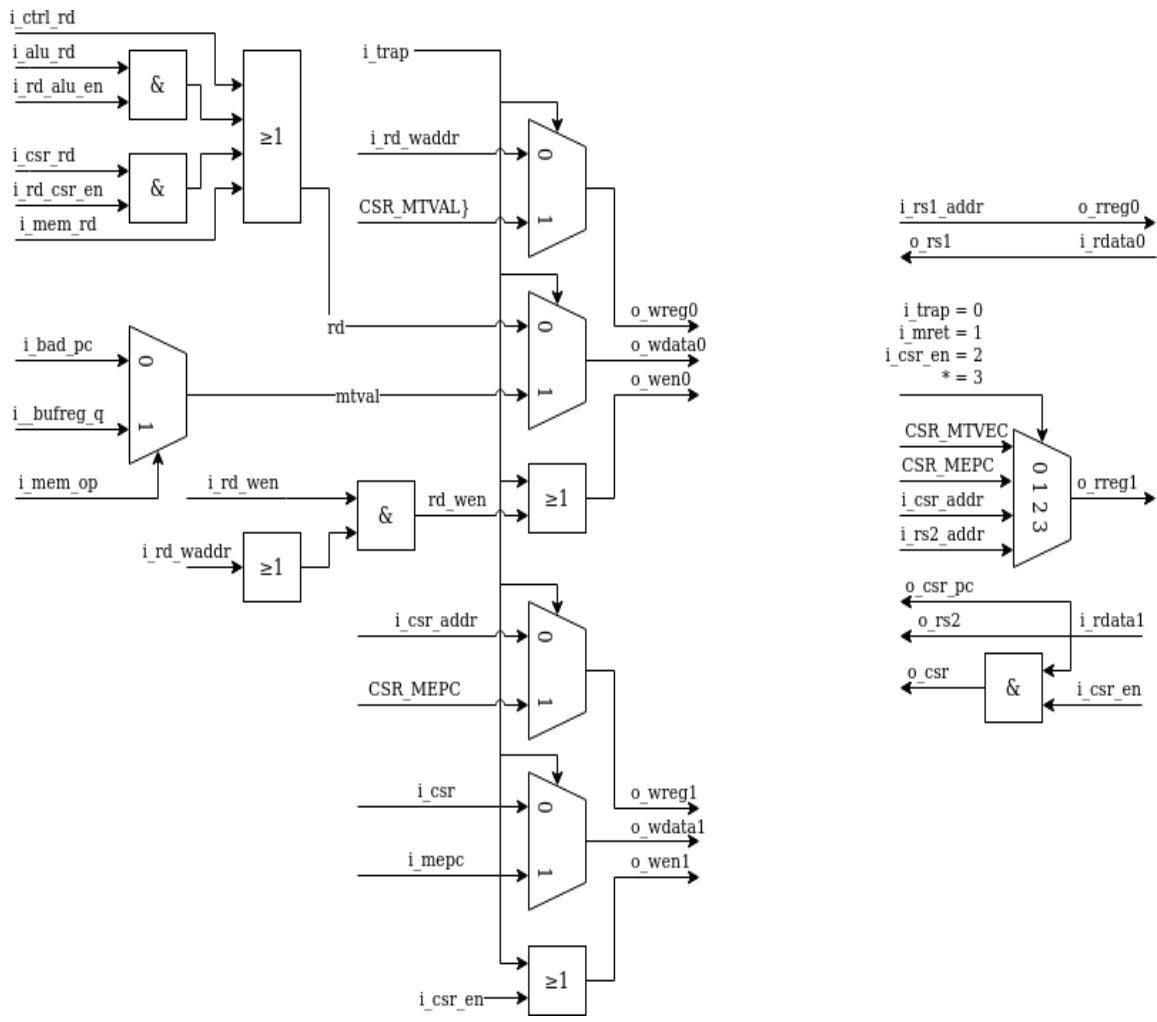


op type	lsb	3	2	1	0
sb	00	0	0	0	1
sb	01	0	0	1	0
sb	10	0	1	0	0
sb	11	1	0	0	0
sh	00	0	0	1	1
sh	10	1	1	0	0
sw	00	1	1	1	1
Logic expression		$(i_lsb == 11) / i_word / (i_half \& i_lsb[1])$	$(i_lsb == 10) / (i_half \& i_word)$	$(i_lsb == 01) / i_word / (i_half \& !i_lsb[1])$	$i_lsb == 0$

serv_rf_if



serv_rf_if is the gateway between the core and an RF implementation. It transforms all control signals that affect register reads or writes and exposes two read and write ports to the RF. This allows implementors to plug in an RF implementation that is best suited for the technology to be used. The general purpose registers are allocated to address 0-31. In addition, four CSR are defined at addresses 32-35.



serv_rf_ram

serv_rf_ram is the default RF implementation using an SRAM-like interface. Suitable for FPGA implementations

serv_rf_ram_if

serv_rf_ram_if converts between the SERV RF IF and the serv_rf_ram interface

serv_state

serv_state keeps track of the state for the core and contains all dynamic control signals during an operations life time. Also controls the accesses towards the RF, ibus and dbus

New instructions are fetched by asserting o_ibus_cyc until there is a response on i_ibus_ack. Instruction fetches occur when the reset signal is deasserted, which is what gets SERV started, or when the PC has finished updating its value.

shift_reg

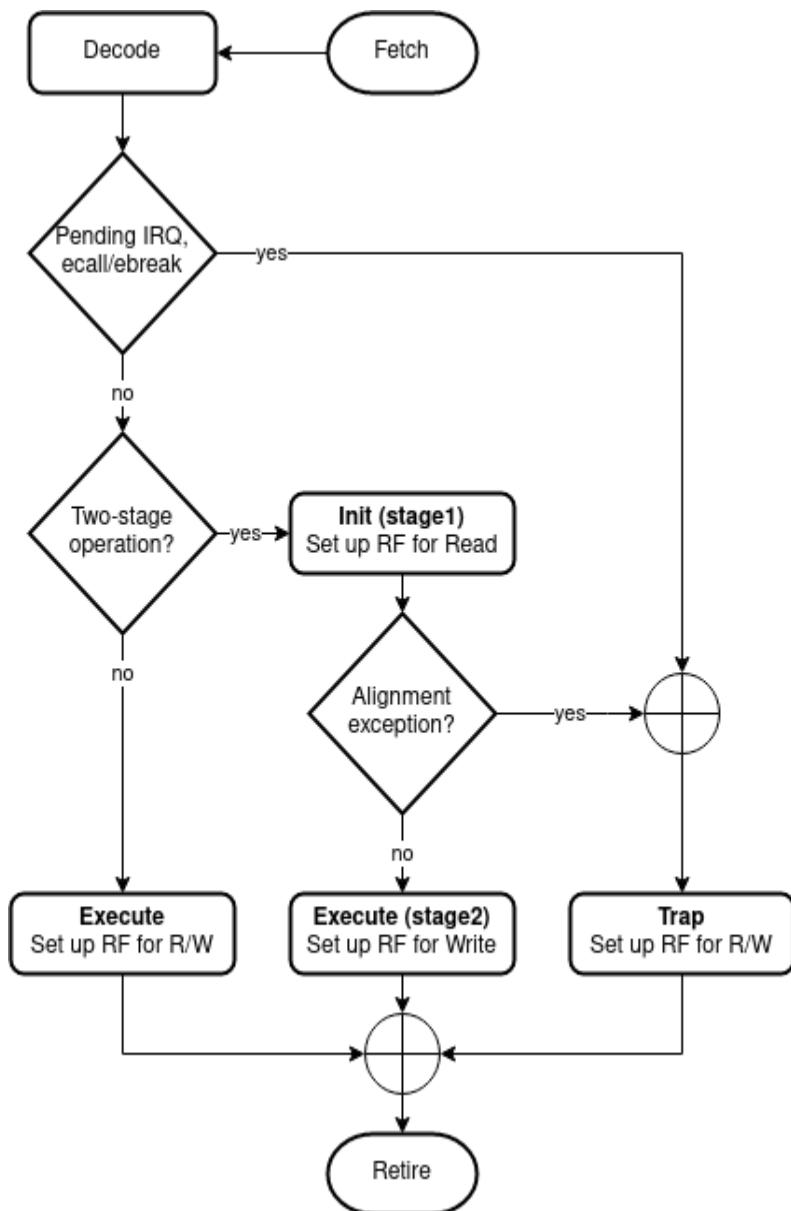
shift_reg is a shift register implementation used in various places in SERV

serv_shift

serv_shift lives inside the ALU and contains the control logic for shift operations

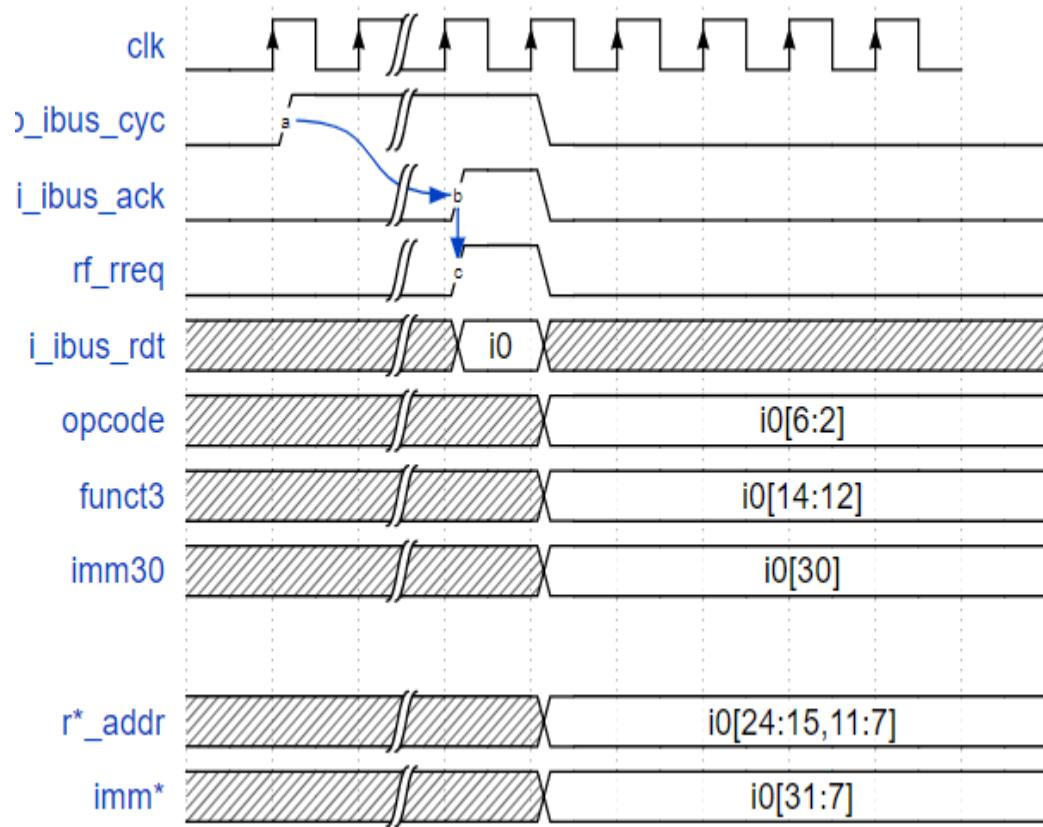
Instruction life cycle

The life cycle of an instruction starts by the core issuing a request for a new instruction on the ibus and ends when the PC has been updated with the address of the next instruction. This section goes through what happens between those points for the various types of instructions. SERV distinguishes between two-stage and one-stage operations with the former category being all jump (branch), shift, slt and load/store instructions and the latter all other operations. In addition to this, exceptions are a special case. Only two-stage operations (jump, load/store) can cause an exception. Regardless of instruction type, they all start out the same way.



Fetch

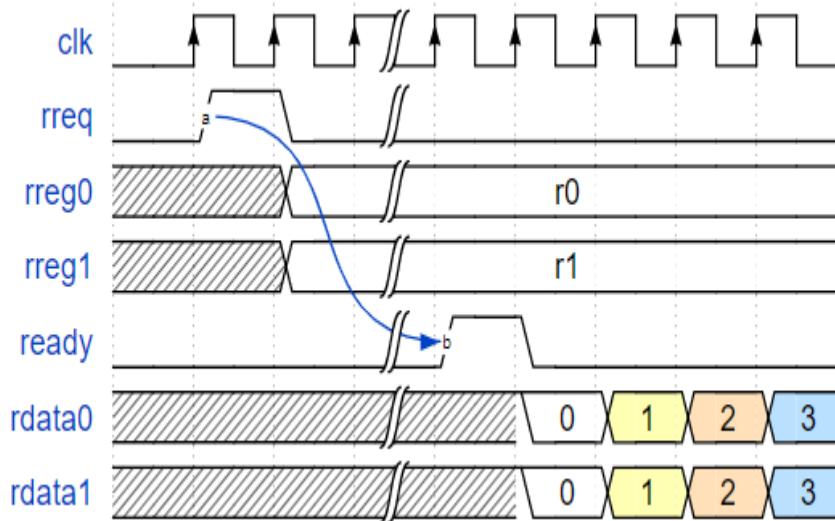
The bus requests begin by SERV raising `o_ibus_cyc` until the memory responds with an `i_ibus_ack` and presents the instruction on `i_ibus_rdt`. Upon seeing the ack, SERV will lower `cyc` to indicate the end of the bus cycle.



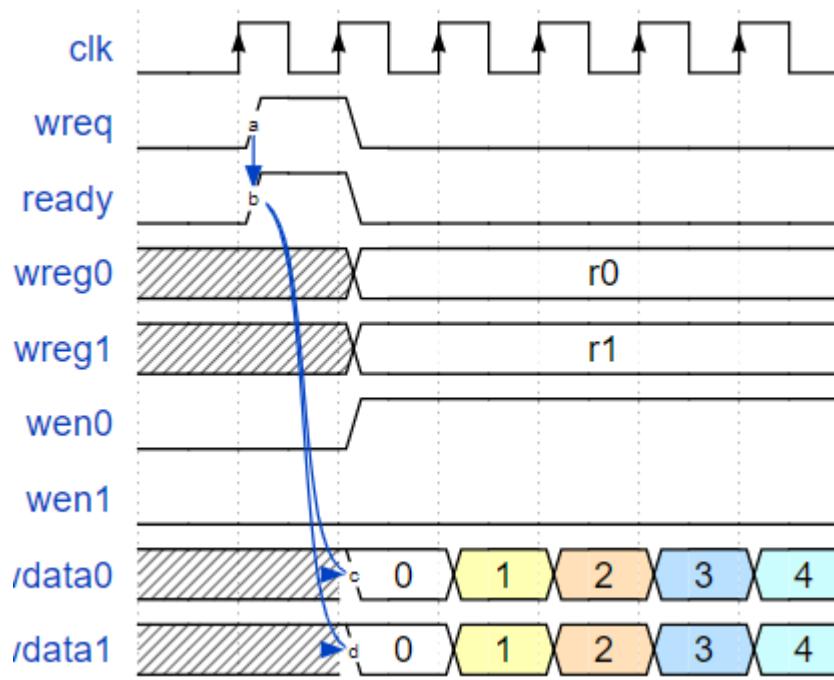
Decode

When the ack appears, two things happen in SERV. The relevant portions of the instruction such as opcode, funct3 and immediate value are saved in serv_decode and serv_immdec. The saved bits of the instruction is then decoded to create the internal control signals that corresponds to the current instruction. The decoded control signals remain static throughout the instruction life cycle.

The other thing to happen is that a request to start accessing the register file is sent by strobing rf_rreq which prepares the register file for both read and write access.



The interface between the core and the register file is described in a protocol where the core strobes `rreq` and present the registers to read on the following cycle. The register file will prepare to stream out data bit from the two requested registers. The cycle before it sends out the first bit (LSB) it will strobe `rf_ready`. Writes work in a similar way in that the registers to write has to be presented the cycle after `rf_wreq` is strobed and that the register file will start accepting data the cycle after it has strobed `rf_ready`. Note that the delay between `rf_wreq` and `rf_ready` does not have to be the same as from `rf_rreq` to `rf_ready`. Also note that register data will only be written to a register if the corresponding write enable signal is asserted. In the diagram below, only register `r0` will be written to.

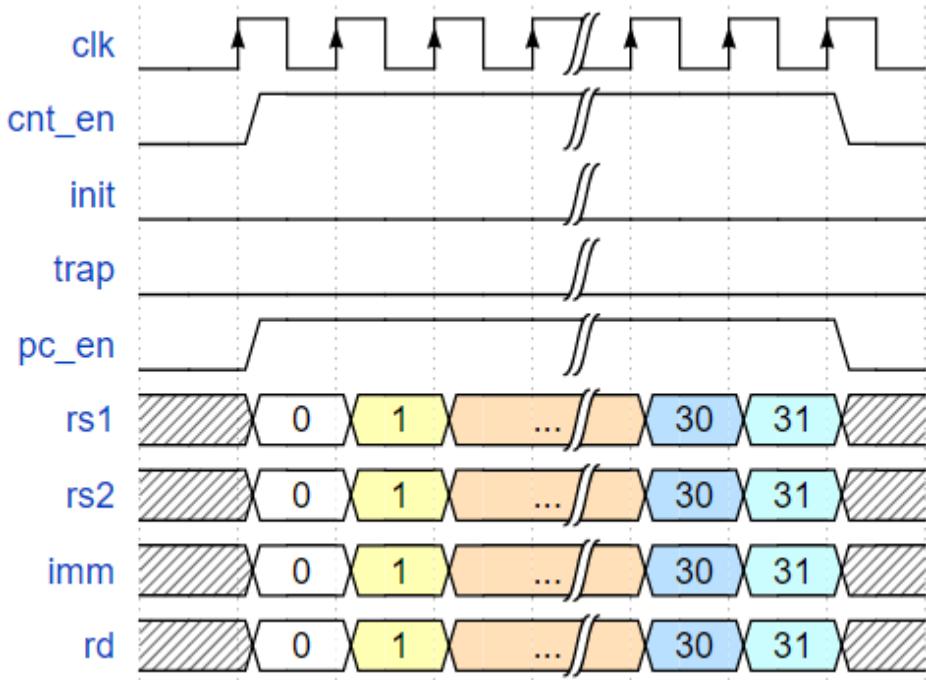


Execute

After the instruction has been decoded and the register file prepared for reads (and possibly writes) the core knows whether it is a one-stage or two-stage instruction. These are handled differently and we will begin by looking at one-stage instructions. A stage in SERV is 32 consecutive cycles during which the core is active and processes inputs and creates results one bit at a time, starting with the LSB.

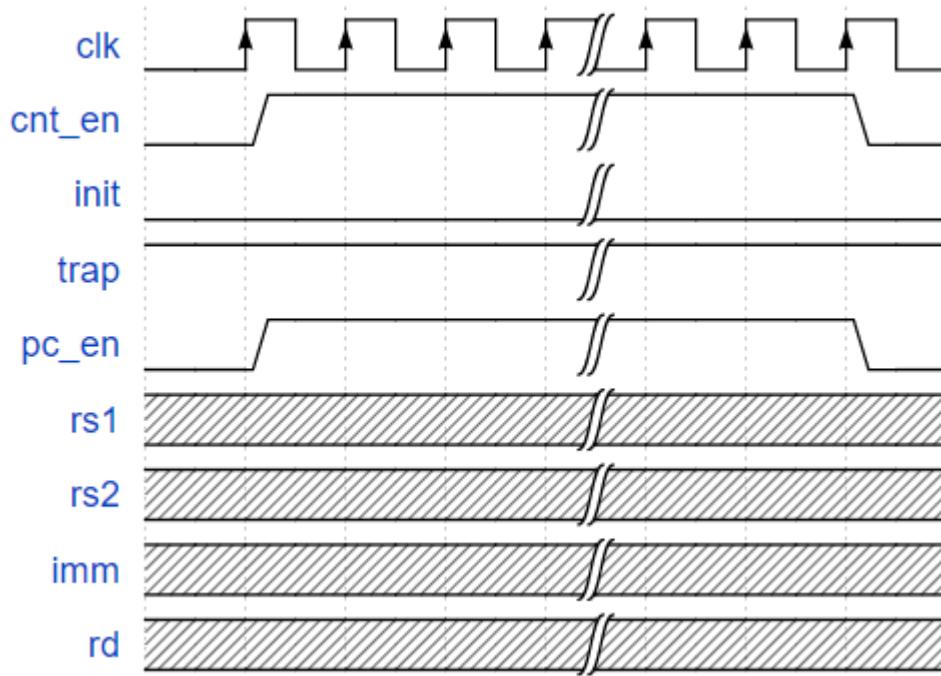
One-stage instructions

Most operations are one-stage operations which finish in 32 cycles + fetch overhead. During a one-stage operation, the RF is read and written simultaneously as well as the PC which is increased by four to point to the next instruction. trap and init signals are low to distinguish from other stages.



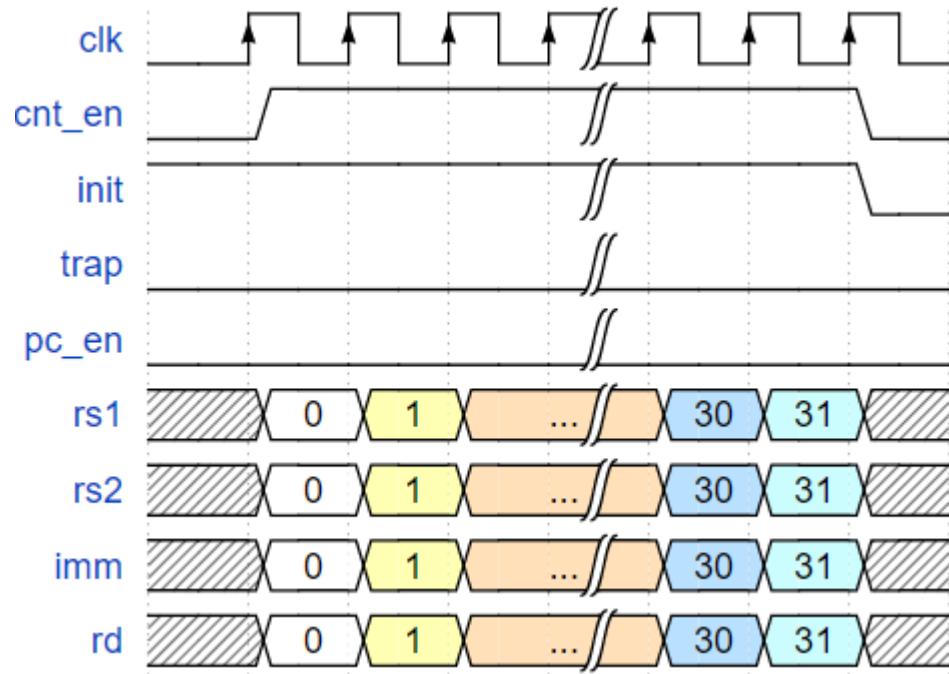
Interrupts and ecall/ebreak

External timer interrupts and ecall/ebreak are also one-stage operations with some notable differences. The new PC is fetched from the MTVEC CSR and instead of writing to rd, the MEPC and MVAL CSR registers are written. All this is handled by serv_state raising the trap signal during the instruction's execution.



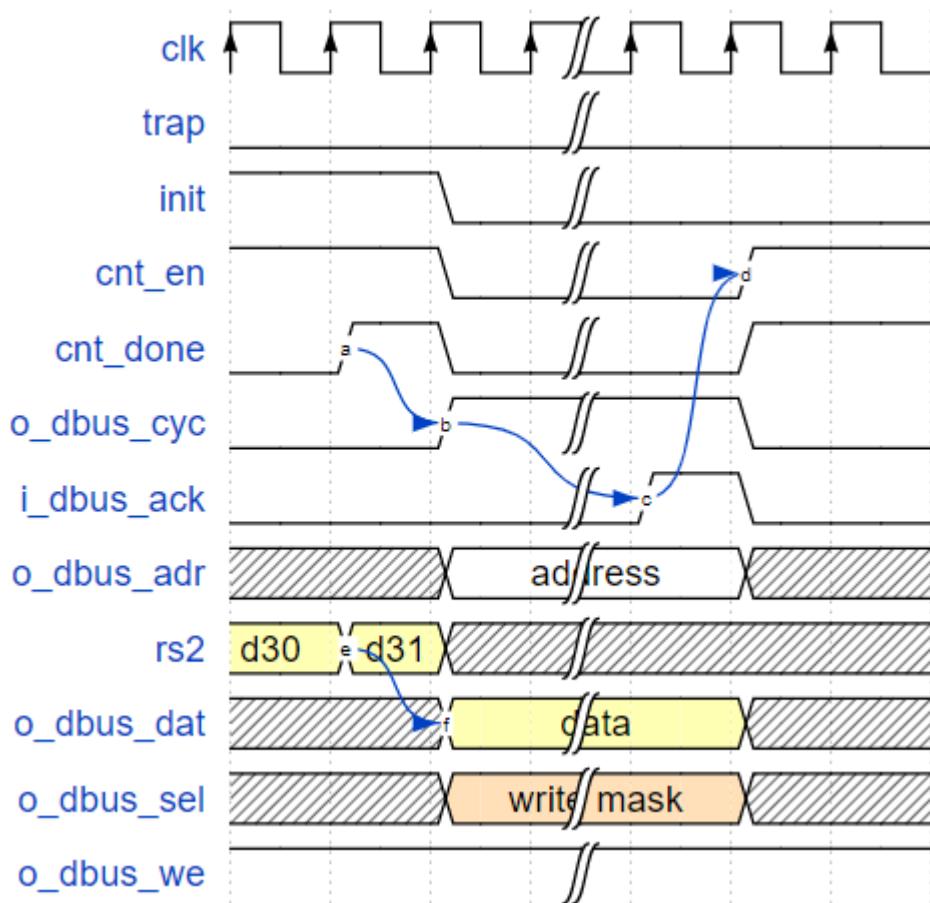
Two-stage operations

Some operations need to be executed in two stages. In the first stage the operands are read out from the immediate and the rs1/rs2 registers and potential results are written to PC and rd in the second stage. Various things happen between the stages depending on the type of operation. SERV has types of four two-stage operations; memory, shift, slt and branch operations. In all cases the first stage is distinguished by having the init signal raised and only performing reads from the RF.

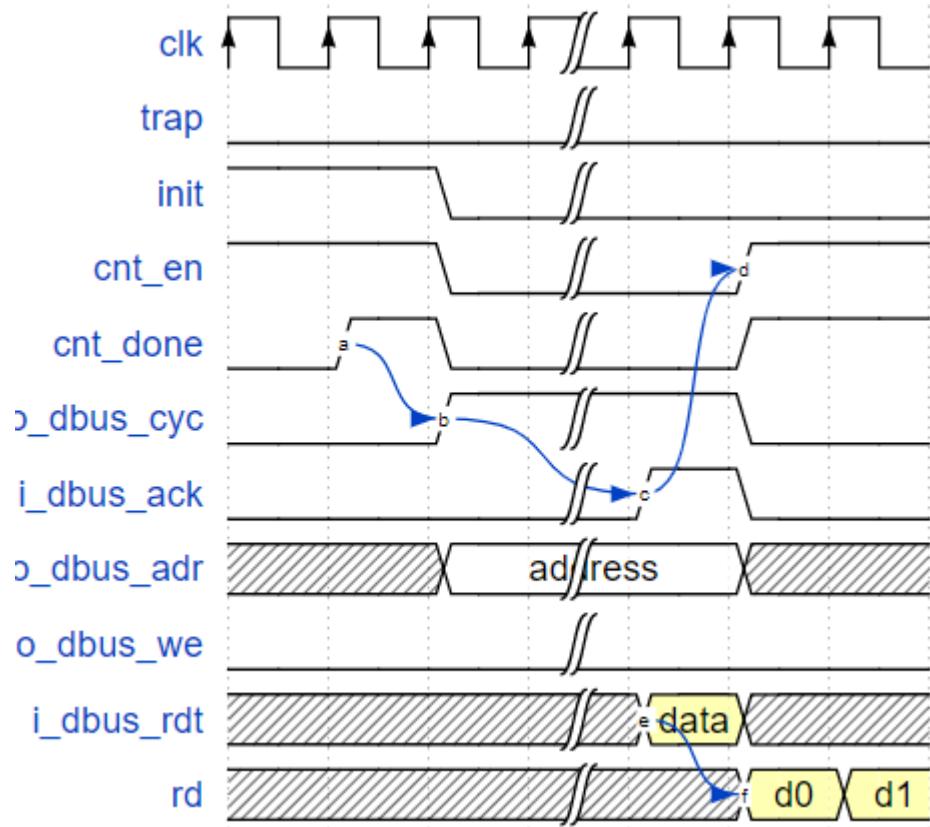


memory

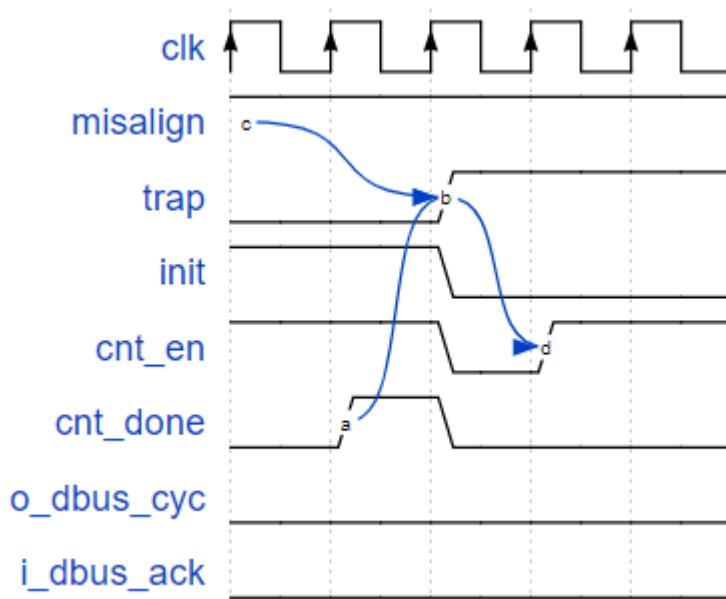
Loads and stores are memory operations. In the init stage, the data address to access is calculated, checked for alignment and stored in `serv_bufreg`. For stores, the data to write is also shifted into the data register in `serv_mem_if`.



If the address has correct alignment, the **o_dbus_cyc** signal is raised to signal an access on the data bus after the **init** stage has finished and waits for an incoming **i_dbus_ack**, and incoming data in case of loads. After an incoming ack, **o_dbus_cyc** is lowered and stage 2 begins. For stores, the only remaining work in stage 2 is to update the PC. For loads, the incoming data is shifted into rd.



If the calculated address in the init stage was misaligned, SERV will raise an exception. Instead of performing an external bus access it will set mcause and raise the trap signal, which causes SERV to store the current PC to mepc, store misaligned address to mtval and set the new PC from mtvec which will enter the exception handler.



Verilog Code

Top module

serv_rf_top

```
`default_nettype none
```

```
module serv_rf_top
#(parameter RESET_PC = 32'd0,
```

```
/* Register signals before or after the decoder
  0 : Register after the decoder. Faster but uses more
resources
  1 : (default) Register before the decoder. Slower but
uses less resources
 */
```

```

parameter PRE_REGISTER = 1,
/* Amount of reset applied to design
"NONE" : No reset at all. Relies on a POR to set
correct initialization
values and that core isn't reset during runtime
"MINI" : Standard setting. Resets the minimal
amount of FFs needed to
restart execution from the instruction at
RESET_PC
*/
parameter RESET_STRATEGY = "MINI",
parameter WITH_CSR = 1,
parameter RF_WIDTH = 2,
parameter RF_L2D =
$clog2((32+(WITH_CSR*4))*32/RF_WIDTH))
(
    input wire      clk,
    input wire      i_rst,
    input wire      i_timer_irq,
`ifdef RISCV_FORMAL
    output wire     rvfi_valid,
    output wire [63:0] rvfi_order,
    output wire [31:0] rvfi_insn,
    output wire     rvfi_trap,
    output wire     rvfi_halt,
    output wire     rvfi_intr,
    output wire [1:0] rvfi_mode,
    output wire [1:0] rvfi_ixl,
    output wire [4:0] rvfi_rs1_addr,
    output wire [4:0] rvfi_rs2_addr,
    output wire [31:0] rvfi_rs1_rdata,
    output wire [31:0] rvfi_rs2_rdata,
    output wire [4:0] rvfi_rd_addr,

```

```

output wire [31:0] rvfi_rd_wdata,
output wire [31:0] rvfi_pc_rdata,
output wire [31:0] rvfi_pc_wdata,
output wire [31:0] rvfi_mem_addr,
output wire [3:0] rvfi_mem_rmask,
output wire [3:0] rvfi_mem_wmask,
output wire [31:0] rvfi_mem_rdata,
output wire [31:0] rvfi_mem_wdata,
`endif
output wire [31:0] o_ibus_adr,
output wire      o_ibus_cyc,
input wire [31:0] i_ibus_rdt,
input wire      i_ibus_ack,
output wire [31:0] o_dbus_adr,
output wire [31:0] o_dbus_dat,
output wire [3:0] o_dbus_sel,
output wire      o_dbus_we ,
output wire      o_dbus_cyc,
input wire [31:0] i_dbus_rdt,
input wire      i_dbus_ack);

```

localparam CSR_REGS = WITH_CSR*4;

```

wire      rf_wreq;
wire      rf_rreq;
wire [4+WITH_CSR:0] wreg0;
wire [4+WITH_CSR:0] wreg1;
wire      wen0;
wire      wen1;
wire      wdata0;
wire      wdata1;
wire [4+WITH_CSR:0] rreg0;
wire [4+WITH_CSR:0] rreg1;

```

```
    wire      rf_ready;
    wire      rdata0;
    wire      rdata1;

    wire [RF_L2D-1:0] waddr;
    wire [RF_WIDTH-1:0] wdata;
    wire      wen;
    wire [RF_L2D-1:0] raddr;
    wire [RF_WIDTH-1:0] rdata;

serv_rf_ram_if
#(.width  (RF_WIDTH),
 .reset_strategy (RESET_STRATEGY),
 .csr_regs (CSR_REGS))
rf_ram_if
(.i_clk  (clk),
 .i_rst  (i_rst),
 .i_wreq  (rf_wreq),
 .i_rreq  (rf_rreq),
 .o_ready (rf_ready),
 .i_wreg0 (wreg0),
 .i_wreg1 (wreg1),
 .i_wen0  (wen0),
 .i_wen1  (wen1),
 .i_wdata0 (wdata0),
 .i_wdata1 (wdata1),
 .i_rreg0 (rreg0),
 .i_rreg1 (rreg1),
 .o_rdata0 (rdata0),
 .o_rdata1 (rdata1),
 .o_waddr (waddr),
 .o_wdata (wdata),
 .o_wen  (wen),
```

```

.o_raddr (raddr),
.i_rdata (rdata));

serv_rf_ram
#(.width (RF_WIDTH),
.csr_regs (CSR_REGS))
rf_ram
(.i_clk (clk),
.i_waddr (waddr),
.i_wdata (wdata),
.i_wen (wen),
.i_raddr (raddr),
.o_rdata (rdata));

serv_top
#(.RESET_PC (RESET_PC),
.PRE_REGISTER (PRE_REGISTER),
.RESET_STRATEGY (RESET_STRATEGY),
.WITH_CSR (WITH_CSR))
cpu
(
.clk (clk),
.i_rst (i_rst),
.i_timer_irq (i_timer_irq),
`ifdef RISCV_FORMAL
.rvfi_valid (rvfi_valid ),
.rvfi_order (rvfi_order ),
.rvfi_insn (rvfi_insn ),
.rvfi_trap (rvfi_trap ),
.rvfi_halt (rvfi_halt ),
.rvfi_intr (rvfi_intr ),
.rvfi_mode (rvfi_mode ),
.rvfi_ixl (rvfi_ixl ),

```

```
.rvfi_rs1_addr (rvfi_rs1_addr ),
.rvfi_rs2_addr (rvfi_rs2_addr ),
.rvfi_rs1_rdata (rvfi_rs1_rdata),
.rvfi_rs2_rdata (rvfi_rs2_rdata),
.rvfi_rd_addr (rvfi_rd_addr ),
.rvfi_rd_wdata (rvfi_rd_wdata ),
.rvfi_pc_rdata (rvfi_pc_rdata ),
.rvfi_pc_wdata (rvfi_pc_wdata ),
.rvfi_mem_addr (rvfi_mem_addr ),
.rvfi_mem_rmask (rvfi_mem_rmask),
.rvfi_mem_wmask (rvfi_mem_wmask),
.rvfi_mem_rdata (rvfi_mem_rdata),
.rvfi_mem_wdata (rvfi_mem_wdata),
`endif
.o_rf_rreq (rf_rreq),
.o_rf_wreq (rf_wreq),
.i_rf_ready (rf_ready),
.o_wreg0 (wreg0),
.o_wreg1 (wreg1),
.o_wen0 (wen0),
.o_wen1 (wen1),
.o_wdata0 (wdata0),
.o_wdata1 (wdata1),
.o_rreg0 (rreg0),
.o_rreg1 (rreg1),
.i_rdata0 (rdata0),
.i_rdata1 (rdata1),

.o_ibus_adr (o_ibus_adr),
.o_ibus_cyc (o_ibus_cyc),
.i_ibus_rdt (i_ibus_rdt),
.i_ibus_ack (i_ibus_ack),
```

```
.o_dbus_adr (o_dbus_adr),  
.o_dbus_dat (o_dbus_dat),  
.o_dbus_sel (o_dbus_sel),  
.o_dbus_we (o_dbus_we),  
.o_dbus_cyc (o_dbus_cyc),  
.i_dbus_rdt (i_dbus_rdt),  
.i_dbus_ack (i_dbus_ack));
```

```
endmodule  
`default_nettype wire
```

Sub modules

Serv_alu

```
`default_nettype none  
module serv_alu  
(  
    input wire      clk,  
    //State  
    input wire      i_en,  
    input wire      i_cnt0,  
    output wire     o_cmp,  
    //Control  
    input wire      i_sub,  
    input wire [1:0] i_bool_op,  
    input wire      i_cmp_eq,  
    input wire      i_cmp_sig,  
    input wire [2:0] i_rd_sel,  
    //Data  
    input wire      i_rs1,  
    input wire      i_op_b,
```

```

input wire      i_buf,
output wire    o_rd);

wire      result_add;

reg       cmp_r;

wire      add_cy;
reg       add_cy_r;

//Sign-extended operands
wire rs1_sx = i_rs1 & i_cmp_sig;
wire op_b_sx = i_op_b & i_cmp_sig;

wire add_b = i_op_b^i_sub;

assign {add_cy,result_add} = i_rs1+add_b+add_cy_r;

wire result_lt = rs1_sx + ~op_b_sx + add_cy;

wire result_eq = !result_add & (cmp_r | i_cnt0);

assign o_cmp = i_cmp_eq ? result_eq : result_lt;

/*

```

The result_bool expression implements the following operations between

i_rs1 and i_op_b depending on the value of i_bool_op

00 xor
 01 0
 10 or
 11 and

i_bool_op will be 01 during shift operations, so by outputting zero under this condition we can safely or result_bool with i_buf

```

/*
wire result_bool = ((i_rs1 ^ i_op_b) & ~ i_bool_op[0]) |
(i_bool_op[1] & i_op_b & i_rs1);

assign o_rd = i_buf |
(i_rd_sel[0] & result_add) |
(i_rd_sel[1] & cmp_r & i_cnt0) |
(i_rd_sel[2] & result_bool);

always @(posedge clk) begin
add_cy_r <= i_en ? add_cy : i_sub;

if (i_en)
  cmp_r <= o_cmp;
end

endmodule

```

Serv_bufreg

```

module serv_bufreg
(
  input wire      i_clk,
  //State
  input wire      i_cnt0,
  input wire      i_cnt1,
  input wire      i_en,
  input wire      i_init,

```

```

output reg [1:0] o_lsb,
//Control
input wire      i_rs1_en,
input wire      i_imm_en,
input wire      i_clr_lsb,
input wire      i_sh_signed,
//Data
input wire      i_rs1,
input wire      i_imm,
output wire     o_q,
//External
output wire [31:0] o_dbus_adr);

wire          c, q;
reg           c_r;
reg [31:2]    data;

wire          clr_lsb = i_cnt0 & i_clr_lsb;

assign {c,q} = {1'b0,(i_rs1 & i_rs1_en)} +
{1'b0,(i_imm & i_imm_en & !clr_lsb)} + c_r;

always @(posedge i_clk) begin
//Make sure carry is cleared before loading new data
  c_r <= c & i_en;

  if (i_en)
    data <= {i_init ? q : (data[31] & i_sh_signed),
data[31:3]};

  if (i_init ? (i_cnt0 | i_cnt1) : i_en)
    o_lsb <= {i_init ? q : data[2],o_lsb[1]};

```

```
end

assign o_q = o_lsb[0] & i_en;
assign o_dbus_adr = { data, 2'b00};
```

```
endmodule
```

serv_csr

```
`default_nettype none
module serv_csr
(
    input wire      i_clk,
    //State
    input wire      i_init,
    input wire      i_en,
    input wire      i_cnt0to3,
    input wire      i_cnt3,
    input wire      i_cnt7,
    input wire      i_cnt_done,
    input wire      i_mem_op,
    input wire      i_mtip,
    input wire      i_trap,
    output reg     o_new_irq,
    //Control
    input wire      i_e_op,
    input wire      i_ebreak,
    input wire      i_mem_cmd,
    input wire      i_mstatus_en,
    input wire      i_mie_en,
    input wire      i_mcause_en,
    input wire [1:0] i_csr_source,
    input wire      i_mret,
```

```

input wire      i_csr_d_sel,
//Data
input wire      i_rf_csr_out,
output wire     o_csr_in,
input wire      i_csr_imm,
input wire      i_rs1,
output wire     o_q);

localparam [1:0]
  CSR_SOURCE_CSR = 2'b00,
  CSR_SOURCE_EXT = 2'b01,
  CSR_SOURCE_SET = 2'b10,
  CSR_SOURCE_CLR = 2'b11;

reg            mstatus_mie;
reg            mstatus_mpie;
reg            mie_mtie;

reg            mcause31;
reg [3:0]      mcause3_0;
wire           mcause;

wire           csr_in;
wire           csr_out;

reg            timer_irq_r;

wire           d = i_csr_d_sel ? i_csr_imm : i_rs1;

assign csr_in = (i_csr_source == CSR_SOURCE_EXT)
? d :
  (i_csr_source == CSR_SOURCE_SET) ?
csr_out | d :

```

```

        (i_csr_source == CSR_SOURCE_CLR) ?
csr_out & ~d :
        (i_csr_source == CSR_SOURCE_CSR) ?
csr_out :
        1'bx;

    assign csr_out = (i_mstatus_en & mstatus_mie &
i_cnt3) |
        i_rf_csr_out |
        (i_mcause_en & i_en & mcause);

assign o_q = csr_out;

wire timer_irq = i_mtip & mstatus_mie & mie_mtie;

assign mcause = i_cnt0to3 ? mcause3_0[0] : // [3:0]
        i_cnt_done ? mcause31 // [31]
        : 1'b0;

assign o_csr_in = csr_in;

always @(posedge i_clk) begin
    if (!i_init & i_cnt_done) begin
        timer_irq_r <= timer_irq;
        o_new_irq <= timer_irq & !timer_irq_r;
    end

    if (i_mie_en & i_cnt7)
        mie_mtie <= csr_in;

/*
    The mie bit in mstatus gets updated under three
    conditions

```

When a trap is taken, the bit is cleared
During an mret instruction, the bit is restored from
mpie
During a mstatus CSR access instruction it's assigned
when
bit 3 gets updated

These conditions are all mutually exclusive
*/
if ((i_trap & i_cnt_done) | i_mstatus_en & i_cnt3 |
i_mret)
 mstatus_mie <= !i_trap & (i_mret ? mstatus_mpie :
csr_in);

/*
Note: To save resources mstatus_mpie (mstatus bit 7)
is not
readable or writable from sw
*/
if (i_trap & i_cnt_done)
 mstatus_mpie <= mstatus_mie;

/*
The four lowest bits in mcause hold the exception
code

These bits get updated under three conditions
During an mcause CSR access function, they are
assigned when
bits 0 to 3 gets updated

During an external interrupt the exception code is set to

7, since SERV only support timer interrupts

During an exception, the exception code is assigned to indicate

if it was caused by an ebreak instruction (3),
ecall instruction (11), misaligned load (4), misaligned store (6)
or misaligned jump (0)

The expressions below are derived from the following truth table

```
irq => 0111 (timer=7)
e_op => x011 (ebreak=3, ecall=11)
mem => 01x0 (store=6, load=4)
ctrl => 0000 (jump=0)
*/
if (i_mcause_en & i_en & i_cnt0to3 | (i_trap &
i_cnt_done)) begin
    mcause3_0[3] <= (i_e_op & !i_ebreak) | (!i_trap &
    csr_in);
    mcause3_0[2] <= o_new_irq | i_mem_op | (!i_trap
    & mcause3_0[3]);
    mcause3_0[1] <= o_new_irq | i_e_op | (i_mem_op
    & i_mem_cmd) | (!i_trap & mcause3_0[2]);
    mcause3_0[0] <= o_new_irq | i_e_op | (!i_trap &
    mcause3_0[1]);
end
if (i_mcause_en & i_cnt_done | i_trap)
    mcause31 <= i_trap ? o_new_irq : csr_in;
end
```

```
endmodule
```

```
serv_ctrl
```

```
`default_nettype none
module serv_ctrl
#(parameter RESET_STRATEGY = "MINI",
  parameter RESET_PC = 32'd0,
  parameter WITH_CSR = 1)
(
  input wire      clk,
  input wire      i_rst,
  //State
  input wire      i_pc_en,
  input wire      i_cnt12to31,
  input wire      i_cnt0,
  input wire      i_cnt2,
  //Control
  input wire      i_jump,
  input wire      i_jal_or_jalr,
  input wire      i_utype,
  input wire      i_pc_rel,
  input wire      i_trap,
  //Data
  input wire      i_imm,
  input wire      i_buf,
  input wire      i_csr_pc,
  output wire     o_rd,
  output wire     o_bad_pc,
  //External
  output reg [31:0] o_ibus_adr);
```

```

wire      pc_plus_4;
wire      pc_plus_4_cy;
reg       pc_plus_4_cy_r;
wire      pc_plus_offset;
wire      pc_plus_offset_cy;
reg       pc_plus_offset_cy_r;
wire      pc_plus_offset_aligned;
wire      plus_4;

wire pc = o_ibus_adr[0];

wire      new_pc;

wire      offset_a;
wire      offset_b;

assign plus_4 = i_cnt2;

assign o_bad_pc = pc_plus_offset_aligned;

assign {pc_plus_4_cy,pc_plus_4} =
pc+plus_4+pc_plus_4_cy_r;

generate
  if (WITH_CSR)
    assign new_pc = i_trap ? (i_csr_pc & !i_cnt0) :
i_jump ? pc_plus_offset_aligned : pc_plus_4;
    else
      assign new_pc = i_jump ? pc_plus_offset_aligned :
pc_plus_4;
  endgenerate
  assign o_rd = (i_utype & pc_plus_offset_aligned) |
(pc_plus_4 & i_jal_or_jalr);

```

```

    assign offset_a = i_pc_rel & pc;
    assign offset_b = i_utype ? (i_imm & i_cnt12to31):
i_buf;
    assign {pc_plus_offset_cy,pc_plus_offset} =
offset_a+offset_b+pc_plus_offset_cy_r;

    assign pc_plus_offset_aligned = pc_plus_offset &
!i_cnt0;

initial if (RESET_STRATEGY == "NONE")
o_ibus_adr = RESET_PC;

always @(posedge clk) begin
    pc_plus_4_cy_r <= i_pc_en & pc_plus_4_cy;
    pc_plus_offset_cy_r <= i_pc_en & pc_plus_offset_cy;

if (RESET_STRATEGY == "NONE") begin
    if (i_pc_en)
        o_ibus_adr <= {new_pc, o_ibus_adr[31:1]};
    end else begin
        if (i_pc_en | i_RST)
            o_ibus_adr <= i_RST ? RESET_PC : {new_pc,
o_ibus_adr[31:1]};
        end
    end
endmodule

```

serv_decode

```
`default_nettype none
```

```
module serv_decode #(
    parameter [0:0] PRE_REGISTER = 1
) (
    input wire      clk,
    //Input
    input wire [31:2] i_wb_rdt,
    input wire      i_wb_en,
    //To state
    output reg     o_sh_right,
    output reg     o_bne_or_bge,
    output reg     o_cond_branch,
    output reg     o_e_op,
    output reg     o_ebreak,
    output reg     o_branch_op,
    output reg     o_mem_op,
    output reg     o_shift_op,
    output reg     o_slt_op,
    output reg     o_rd_op,
    //To bufreg
    output reg     o_bufreg_rs1_en,
    output reg     o_bufreg_imm_en,
    output reg     o_bufreg_clr_lsb,
    output reg     o_bufreg_sh_signed,
    //To ctrl
    output reg     o_ctrl_jal_or_jalr,
    output reg     o_ctrl_utype,
    output reg     o_ctrl_pc_rel,
    output reg     o_ctrl_mret,
    //To alu
    output reg     o_alu_sub,
    output reg [1:0] o_alu_bool_op,
    output reg     o_alu_cmp_eq,
    output reg     o_alu_cmp_sig,
```

```
output reg [2:0] o_alu_rd_sel,  
//To mem IF  
output reg      o_mem_signed,  
output reg      o_mem_word,  
output reg      o_mem_half,  
output reg      o_mem_cmd,  
//To CSR  
output reg      o_csr_en,  
output reg [1:0] o_csr_addr,  
output reg      o_csr_mstatus_en,  
output reg      o_csr_mie_en,  
output reg      o_csr_mcause_en,  
output reg [1:0] o_csr_source,  
output reg      o_csr_d_sel,  
output reg      o_csr_imm_en,  
//To top  
output reg [3:0] o_immdec_ctrl,  
output reg [3:0] o_immdec_en,  
output reg      o_op_b_source,  
output reg      o_rd_csr_en,  
output reg      o_rd_alu_en);  
  
reg [4:0] opcode;  
reg [2:0] funct3;  
reg      op20;  
reg      op21;  
reg      op22;  
reg      op26;  
  
reg      imm30;  
  
//opcode
```

```

    wire op_or_opimm = (!opcode[4] & opcode[2] &
!opcode[0]);

    wire co_mem_op  = !opcode[4] & !opcode[2] &
!opcode[0];
    wire co_branch_op = opcode[4] & !opcode[2];

    //jal,branch = imm
    //jalr      = rs1+imm
    //mem       = rs1+imm
    //shift     = rs1
    wire co_bufreg_rs1_en = !opcode[4] | (!opcode[1] &
opcode[0]);
    wire co_bufreg_imm_en = !opcode[2];

    //Clear LSB of immediate for BRANCH and JAL ops
    //True for BRANCH and JAL
    //False for JALR/LOAD/STORE/OP/OPIMM?
    wire co_bufreg_clr_lsb = opcode[4] & ((opcode[1:0] ==
2'b00) | (opcode[1:0] == 2'b11));

    //Conditional branch
    //True for BRANCH
    //False for JAL/JALR
    wire co_cond_branch = !opcode[0];

    wire co_ctrl_utype    = !opcode[4] & opcode[2] &
opcode[0];
    wire co_ctrl_jal_or_jalr = opcode[4] & opcode[0];

    //PC-relative operations
    //True for jal, b* auipc
    //False for jalr, lui

```

```

wire co_ctrl_pc_rel = (opcode[2:0] == 3'b000) |
    (opcode[1:0] == 2'b11) |
    (opcode[4:3] == 2'b00);

//Write to RD
//True for OP-IMM, AUIPC, OP, LUI, SYSTEM,
JALR, JAL, LOAD
//False for STORE, BRANCH, MISC-MEM
wire co_rd_op = (opcode[2] |
    (!opcode[2] & opcode[4] & opcode[0]) |
    (!opcode[2] & !opcode[3] & !opcode[0]));

//
//funct3
//

wire co_sh_right  = funct3[2];
wire co_bne_or_bge = funct3[0];

//
// opcode & funct3
//

wire co_shift_op = op_or_opimm & (funct3[1:0] ==
2'b01);
wire co_slt_op  = op_or_opimm & (funct3[2:1] ==
2'b01);

//Matches system ops except eceall/ebreak/mret
wire csr_op = opcode[4] & opcode[2] & (~funct3);

//
//op20
wire co_ebreak = op20;

```

```

//opcode & funct3 & op21

    wire co_ctrl_mret = opcode[4] & opcode[2] & op21 &
!(funct3);
    //Matches system opcodes except CSR accesses (funct3
== 0)
    //and mret (!op21)
    wire co_e_op = opcode[4] & opcode[2] & !op21 &
!(funct3);

//opcode & funct3 & imm30

    wire co_bufreg_sh_signed = imm30;

/*
True for sub, b*, slt*
False for add*
op  opcode f3 i30
b*  11000 xxx x t
addi 00100 000 x f
slt* 0x100 01x x t
add  01100 000 0 f
sub  01100 000 1 t
*/
    wire co_alu_sub = funct3[1] | funct3[0] | (opcode[3] &
imm30) | opcode[4];

/*
Bits 26, 22, 21 and 20 are enough to uniquely identify
the eight supported CSR regs

```

mtvec, mscratch, mepc and mtval are stored externally (normally in the RF) and are

treated differently from mstatus, mie and mcause which are stored in serv_csr.

The former get a 2-bit address as seen below while the latter get a one-hot enable signal each.

Hex		2 222		Reg		csr
adr		6 210		name		addr
300		0_000		mstatus		xx
304		0_100		mie		xx
305		0_101		mtvec		01
340		1_000		mscratch		00
341		1_001		mepc		10
342		1_010		mcause		xx
343		1_011		mtval		11

*/

//true for mtvec,mscratch,mepc and mtval

//false for mstatus, mie, mcause

wire csr_valid = op20 | (op26 & !op21);

wire co_rd_csr_en = csr_op;

wire co_csr_en = csr_op & csr_valid;

wire co_csr_mstatus_en = csr_op & !op26 & !op22;

wire co_csr_mie_en = csr_op & !op26 & op22 &
!op20;

```

    wire co_csr_mcause_en = csr_op      & op21 &
!op20;

    wire [1:0] co_csr_source = funct3[1:0];
    wire co_csr_d_sel = funct3[2];
    wire co_csr_imm_en = opcode[4] & opcode[2] &
funct3[2];
    wire [1:0] co_csr_addr = {op26 & op20, !op26 | op21};

    wire co_alu_cmp_eq = funct3[2:1] == 2'b00;

    wire co_alu_cmp_sig = ~((funct3[0] & funct3[1]) |
(funct3[1] & funct3[2]));

    wire co_mem_cmd = opcode[3];
    wire co_mem_signed = ~funct3[2];
    wire co_mem_word  = funct3[1];
    wire co_mem_half  = funct3[0];

    wire [1:0] co_alu_bool_op = funct3[1:0];

    wire [3:0] co_immdec_ctrl;
    //True for S (STORE) or B (BRANCH) type
instructions
    //False for J type instructions
    assign co_immdec_ctrl[0] = opcode[3:0] == 4'b1000;
    //True for OP-IMM, LOAD, STORE, JALR (I S)
    //False for LUI, AUIPC, JAL      (U J)
    assign co_immdec_ctrl[1] = (opcode[1:0] == 2'b00) |
(opcode[2:1] == 2'b00);
    assign co_immdec_ctrl[2] = opcode[4] & !opcode[0];
    assign co_immdec_ctrl[3] = opcode[4];

```

```

    wire [3:0] co_immdec_en;
    assign co_immdec_en[3] = opcode[4] | opcode[3] |
opcode[2] | !opcode[0];           //B I J S U
    assign co_immdec_en[2] = (opcode[4] & opcode[2]) |
!opcode[3] | opcode[0];         // I J U
    assign co_immdec_en[1] = (opcode[2:1] == 2'b01) |
(opcode[2] & opcode[0]) | co_csr_imm_en;// J U
    assign co_immdec_en[0] = ~co_rd_op;
//B   S

    wire [2:0] co_alu_rd_sel;
    assign co_alu_rd_sel[0] = (funct3 == 3'b000); //
Add/sub
    assign co_alu_rd_sel[1] = (funct3[2:1] == 2'b01);
//SLT*
    assign co_alu_rd_sel[2] = funct3[2]; //Bool

//0 (OP_B_SOURCE_IMM) when OPIMM
//1 (OP_B_SOURCE_RS2) when BRANCH or OP
    wire co_op_b_source = opcode[3];

    wire co_rd_alu_en = !opcode[0] & opcode[2] &
!opcode[4];

generate
    if (PRE_REGISTER) begin

        always @(posedge clk) begin
            if (i_wb_en) begin
                funct3 <= i_wb_rdt[14:12];
                imm30 <= i_wb_rdt[30];
                opcode <= i_wb_rdt[6:2];
                op20 <= i_wb_rdt[20];

```

```

op21 <= i_wb_rdt[21];
op22 <= i_wb_rdt[22];
op26 <= i_wb_rdt[26];
end
end

always @(*) begin
    o_sh_right      = co_sh_right;
    o_bne_or_bge   = co_bne_or_bge;
    o_cond_branch  = co_cond_branch;
    o_e_op          = co_e_op;
    o_ebreak        = co_ebreak;
    o_branch_op    = co_branch_op;
    o_mem_op        = co_mem_op;
    o_shift_op     = co_shift_op;
    o_slt_op        = co_slt_op;
    o_rd_op         = co_rd_op;
    o_bufreg_rs1_en = co_bufreg_rs1_en;
    o_bufreg_imm_en = co_bufreg_imm_en;
    o_bufreg_clr_lsb = co_bufreg_clr_lsb;
    o_bufreg_sh_signed = co_bufreg_sh_signed;
    o_ctrl_jal_or_jalr = co_ctrl_jal_or_jalr;
    o_ctrl_utype   = co_ctrl_utype;
    o_ctrl_pc_rel  = co_ctrl_pc_rel;
    o_ctrl_mret    = co_ctrl_mret;
    o_alu_sub      = co_alu_sub;
    o_alu_bool_op  = co_alu_bool_op;
    o_alu_cmp_eq   = co_alu_cmp_eq;
    o_alu_cmp_sig  = co_alu_cmp_sig;
    o_alu_rd_sel   = co_alu_rd_sel;
    o_mem_signed   = co_mem_signed;
    o_mem_word     = co_mem_word;
    o_mem_half     = co_mem_half;

```

```
    o_mem_cmd      = co_mem_cmd;
    o_csr_en       = co_csr_en;
    o_csr_addr     = co_csr_addr;
    o_csr_mstatus_en = co_csr_mstatus_en;
    o_csr_mie_en   = co_csr_mie_en;
    o_csr_mcause_en = co_csr_mcause_en;
    o_csr_source   = co_csr_source;
    o_csr_d_sel    = co_csr_d_sel;
    o_csr_imm_en   = co_csr_imm_en;
    o_immdec_ctrl  = co_immdec_ctrl;
    o_immdec_en    = co_immdec_en;
    o_op_b_source  = co_op_b_source;
    o_rd_csr_en    = co_rd_csr_en;
    o_rd_alu_en    = co_rd_alu_en;
end
```

```
end else begin
```

```
    always @(*) begin
        funct3 = i_wb_rdt[14:12];
        imm30  = i_wb_rdt[30];
        opcode = i_wb_rdt[6:2];
        op20   = i_wb_rdt[20];
        op21   = i_wb_rdt[21];
        op22   = i_wb_rdt[22];
        op26   = i_wb_rdt[26];
    end
```

```
    always @(posedge clk) begin
        if (i_wb_en) begin
            o_sh_right    <= co_sh_right;
            o_bne_or_bge  <= co_bne_or_bge;
            o_cond_branch <= co_cond_branch;
```

```
o_e_op      <= co_e_op;
o_ebreak    <= co_ebreak;
o_branch_op <= co_branch_op;
o_mem_op    <= co_mem_op;
o_shift_op  <= co_shift_op;
o_slt_op    <= co_slt_op;
o_rd_op     <= co_rd_op;
o_bufreg_rs1_en <= co_bufreg_rs1_en;
o_bufreg_imm_en <= co_bufreg_imm_en;
o_bufreg_clr_lsb <= co_bufreg_clr_lsb;
o_bufreg_sh_signed <= co_bufreg_sh_signed;
o_ctrl_jal_or_jalr <= co_ctrl_jal_or_jalr;
o_ctrl_utype <= co_ctrl_utype;
o_ctrl_pc_rel <= co_ctrl_pc_rel;
o_ctrl_mret  <= co_ctrl_mret;
o_alu_sub    <= co_alu_sub;
o_alu_bool_op <= co_alu_bool_op;
o_alu_cmp_eq <= co_alu_cmp_eq;
o_alu_cmp_sig <= co_alu_cmp_sig;
o_alu_rd_sel <= co_alu_rd_sel;
o_mem_signed <= co_mem_signed;
o_mem_word   <= co_mem_word;
o_mem_half   <= co_mem_half;
o_mem_cmd    <= co_mem_cmd;
o_csr_en     <= co_csr_en;
o_csr_addr   <= co_csr_addr;
o_csr_mstatus_en <= co_csr_mstatus_en;
o_csr_mie_en <= co_csr_mie_en;
o_csr_mcause_en <= co_csr_mcause_en;
o_csr_source <= co_csr_source;
o_csr_d_sel  <= co_csr_d_sel;
o_csr_imm_en <= co_csr_imm_en;
o_immdec_ctrl <= co_immdec_ctrl;
```

```

        o_immdec_en    <= co_immdec_en;
        o_op_b_source  <= co_op_b_source;
        o_rd_csr_en    <= co_rd_csr_en;
        o_rd_alu_en    <= co_rd_alu_en;
    end
end

end
endgenerate

endmodule

```

serv_immdec

```

`default_nettype none
module serv_immdec
#(parameter SHARED_RFADDR_IMM_REGS = 1)
(
    input wire      i_clk,
    //State
    input wire      i_cnt_en,
    input wire      i_cnt_done,
    //Control
    input wire [3:0] i_immdec_en,
    input wire i_csr_imm_en, input
    wire [3:0] i_ctrl,
    output wire [4:0] o_rd_addr,
    output wire [4:0] o_rs1_addr,
    output wire [4:0] o_rs2_addr,
    //Data

```

```

output wire      o_csr_imm,
output wire      o_imm,
//External
input wire       i_wb_en,
input wire [31:7] i_wb_rdt);

reg              signbit;

reg [8:0] imm19_12_20;
reg          imm7;
reg [5:0] imm30_25;
reg [4:0] imm24_20;
reg [4:0] imm11_7;

assign o_imm = i_cnt_done ? signbit : i_ctrl[0] ?
imm11_7[0] : imm24_20[0];
assign o_csr_imm = imm19_12_20[4];

generate
  if (SHARED_RFADDR_IMM_REGS) begin
    assign o_rs1_addr = imm19_12_20[8:4];
    assign o_rs2_addr = imm24_20;
    assign o_rd_addr = imm11_7;

    always @ (posedge i_clk) begin
      if (i_wb_en) begin
        /* CSR immediates are always zero-extended,
        hence clear the signbit */
        signbit   <= i_wb_rdt[31] & !i_csr_imm_en;
      end
      if (i_wb_en | (i_cnt_en & i_immdec_en[1]))

```

```

    imm19_12_20 <= i_wb_en ?
{ i_wb_rdt[19:12], i_wb_rdt[20] } : { i_ctrl[3] ? signbit :
imm24_20[0], imm19_12_20[8:1] };
    if (i_wb_en | (i_cnt_en))
        imm7      <= i_wb_en ? i_wb_rdt[7]
: signbit;

    if (i_wb_en | (i_cnt_en & i_immdec_en[3]))
        imm30_25  <= i_wb_en ? i_wb_rdt[30:25] :
{ i_ctrl[2] ? imm7 : i_ctrl[1] ? signbit : imm19_12_20[0],
imm30_25[5:1] };

    if (i_wb_en | (i_cnt_en & i_immdec_en[2]))
        imm24_20  <= i_wb_en ? i_wb_rdt[24:20] :
{ imm30_25[0], imm24_20[4:1] };

    if (i_wb_en | (i_cnt_en & i_immdec_en[0]))
        imm11_7   <= i_wb_en ? i_wb_rdt[11:7] :
{ imm30_25[0], imm11_7[4:1] };
    end
end else begin
    reg [4:0] rd_addr;
    reg [4:0] rs1_addr;
    reg [4:0] rs2_addr;

    assign o_rd_addr = rd_addr;
    assign o_rs1_addr = rs1_addr;
    assign o_rs2_addr = rs2_addr;
    always @(posedge i_clk) begin
        if (i_wb_en) begin
            /* CSR immediates are always zero-extended,
hence clear the signbit */
            signbit   <= i_wb_rdt[31] & !i_csr_imm_en;

```

```

        imm19_12_20 <=
{ i_wb_rdt[19:12], i_wb_rdt[20] };
        imm7 <= i_wb_rdt[7];
        imm30_25 <= i_wb_rdt[30:25];
        imm24_20 <= i_wb_rdt[24:20];
        imm11_7 <= i_wb_rdt[11:7];

        rd_addr <= i_wb_rdt[11:7];
        rs1_addr <= i_wb_rdt[19:15];
        rs2_addr <= i_wb_rdt[24:20];
end
if (i_cnt_en) begin
    imm19_12_20 <= { i_ctrl[3] ? signbit :
imm24_20[0], imm19_12_20[8:1] };
    imm7 <= signbit;
    imm30_25 <= { i_ctrl[2] ? imm7 : i_ctrl[1] ?
signbit : imm19_12_20[0], imm30_25[5:1] };
    imm24_20 <= { imm30_25[0],
imm24_20[4:1] };
    imm11_7 <= { imm30_25[0], imm11_7[4:1] };
end
end
endgenerate

endmodule

```

serv_mem_if

```

`default_nettype none
module serv_mem_if

```

```

 #(parameter WITH_CSR = 1)
(
    input wire      i_clk,
    //State
    input wire      i_en,
    input wire      i_init,
    input wire      i_cnt_done,
    input wire [1:0] i_bytect,
    input wire [1:0] i_lsb,
    output wire     o_misalign,
    output wire     o_sh_done,
    output wire     o_sh_done_r,
    //Control
    input wire      i_mem_op,
    input wire      i_shift_op,
    input wire      i_signed,
    input wire      i_word,
    input wire      i_half,
    //Data
    input wire      i_op_b,
    output wire     o_rd,
    //External interface
    output wire [31:0] o_wb_dat,
    output wire [3:0]  o_wb_sel,
    input wire [31:0] i_wb_rdt,
    input wire      i_wb_ack);

    reg      signbit;
    reg [31:0]  dat;

/*

```

Before a store operation, the data to be written needs to
be shifted into

place. Depending on the address alignment, we need to shift different

amounts. One formula for calculating this is to say that we shift when

$i_{lsb} + i_{bytecnt} < 4$. Unfortunately, the synthesis tools don't seem to be

clever enough so the hideous expression below is used to achieve the same

thing in a more optimal way.

*/

```
wire byte_valid =
  (!i_lsb[0] & !i_lsb[1])      |
  (!i_bytecnt[0] & !i_bytecnt[1]) |
  (!i_bytecnt[1] & !i_lsb[1])   |
  (!i_bytecnt[1] & !i_lsb[0])   |
  (!i_bytecnt[0] & !i_lsb[1]);
```

```
wire dat_en = i_shift_op | (i_en & byte_valid);
```

```
wire dat_cur =
  ((i_lsb == 2'd3) & dat[24]) |
  ((i_lsb == 2'd2) & dat[16]) |
  ((i_lsb == 2'd1) & dat[8]) |
  ((i_lsb == 2'd0) & dat[0]);
```

```
wire dat_valid =
  i_word |
  (i_bytecnt == 2'b00) |
  (i_half & !i_bytecnt[1]);
```

```
assign o_rd = i_mem_op & (dat_valid ? dat_cur :
  signbit & i_signed);
```

```

    assign o_wb_sel[3] = (i_lsb == 2'b11) | i_word | (i_half
& i_lsb[1]);
    assign o_wb_sel[2] = (i_lsb == 2'b10) | i_word;
    assign o_wb_sel[1] = (i_lsb == 2'b01) | i_word | (i_half
& !i_lsb[1]);
    assign o_wb_sel[0] = (i_lsb == 2'b00);

    assign o_wb_dat = dat;

/* The dat register has three different use cases for store,
load and
shift operations.
store : Data to be written is shifted to the correct
position in dat during
    init by dat_en and is presented on the data bus as
o_wb_dat
load : Data from the bus gets latched into dat during
i_wb_ack and is then
    shifted out at the appropriate time to end up in the
correct
    position in rd
shift : Data is shifted in during init. After that, the six
LSB are used as
    a downcounter (with bit 5 initially set to 0) that
triggers
    o_sh_done and o_sh_done_r when they wrap
around to indicate that
    the requested number of shifts have been
performed
*/
wire [5:0] dat_shamt = (i_shift_op & !i_init) ?
//Down counter mode
dat[5:0]-1 :

```

```

//Shift reg mode with optional clearing of bit 5
{dat[6] & !(i_shift_op & i_cnt_done),dat[5:1]};

assign o_sh_done = dat_shamt[5];
assign o_sh_done_r = dat[5];

always @(posedge i_clk) begin
    if (dat_en | i_wb_ack)
        dat <= i_wb_ack ? i_wb_rdt : {i_op_b, dat[31:7],
dat_shamt};

    if (dat_valid)
        signbit <= dat_cur;
end

/*
    mem_misalign is checked after the init stage to decide
whether to do a data
    bus transaction or go to the trap state. It is only
guaranteed to be correct
    at this time
*/
assign o_misalign = WITH_CSR & ((i_lsb[0] &
(i_word | i_half)) | (i_lsb[1] & i_word));

endmodule

```

serv_rf_if

```

`default_nettype none
module serv_rf_if
#(parameter WITH_CSR = 1)
//RF Interface

```

```

input wire           i_cnt_en,
output wire [4+WITH_CSR:0] o_wreg0,
output wire [4+WITH_CSR:0] o_wreg1,
output wire           o_wen0,
output wire           o_wen1,
output wire           o_wdata0,
output wire           o_wdata1,
output wire [4+WITH_CSR:0] o_rreg0,
output wire [4+WITH_CSR:0] o_rreg1,
input wire           i_rdata0,
input wire           i_rdata1,

//Trap interface
input wire           i_trap,
input wire           i_mret,
input wire           i_mepc,
input wire           i_mem_op,
input wire           i_bufreg_q,
input wire           i_bad_pc,
output wire          o_csr_pc,
//CSR interface
input wire           i_csr_en,
input wire [1:0]      i_csr_addr,
input wire           i_csr,
output wire          o_csr,
//RD write port
input wire           i_rd_wen,
input wire [4:0]      i_rd_waddr,
input wire           i_ctrl_rd,
input wire           i_alu_rd,
input wire           i_rd_alu_en,
input wire           i_csr_rd,
input wire           i_rd_csr_en,

```

```

input wire           i_mem_rd,
//RS1 read port
input wire [4:0]    i_rs1_raddr,
output wire         o_rs1,
//RS2 read port
input wire [4:0]    i_rs2_raddr,
output wire         o_rs2);

/*
***** Write side *****
*/
wire      rd_wen = i_rd_wen & (~i_rd_waddr);
generate
if (WITH_CSR) begin
wire      rd = (i_ctrl_rd ) |
              (i_alu_rd & i_rd_alu_en) |
              (i_csr_rd & i_rd_csr_en) |
              (i_mem_rd);

wire      mtval = i_mem_op ? i_bufreg_q : i_bad_pc;
assign o_wdata0 = i_trap ? mtval : rd;
assign o_wdata1 = i_trap ? i_mepc : i_csr;

/* Port 0 handles writes to mtval during traps and rd
otherwise
 * Port 1 handles writes to mepc during traps and csr
accesses otherwise
 *

```

* GPR registers are mapped to address 0-31 (bits 0xxxxx).

* Following that are four CSR registers

* mscratch 100000

* mtvec 100001

* mepc 100010

* mtval 100011

*/

```
assign o_wreg0 = i_trap ? {6'b100011} :  
{1'b0,i_rd_waddr};
```

```
assign o_wreg1 = i_trap ? {6'b100010} :  
{4'b1000,i_csr_addr};
```

```
assign o_wen0 = i_cnt_en & (i_trap | rd_wen);  
assign o_wen1 = i_cnt_en & (i_trap | i_csr_en);
```

/*

***** Read side *****

*/

//0 : RS1

//1 : RS2 / CSR

```
assign o_rreg0 = {1'b0, i_rs1_raddr};
```

/*

The address of the second read port (o_rreg1) can get assigned from four different sources

Normal operations : i_rs2_raddr

CSR access : i_csr_addr

```
trap      : MTVEC  
mret      : MEPC
```

Address 0-31 in the RF are assigned to the GPRs. After that follows the four CSRs on addresses 32-35

```
32 MSCRATCH  
33 MTVEC  
34 MEPC  
35 MVAL
```

The expression below is an optimized version of this logic

```
/*  
wire sel_rs2 = !(i_trap | i_mret | i_csr_en);  
assign o_rreg1 = {~sel_rs2,  
                 i_rs2_raddr[4:2] & {3{sel_rs2}},  
                 {1'b0,i_trap} | {i_mret,1'b0} |  
({2{i_csr_en}} & i_csr_addr) | ({2{sel_rs2}} &  
i_rs2_raddr[1:0]);  
  
assign o_rs1 = i_rdata0;  
assign o_rs2 = i_rdata1;  
assign o_csr = i_rdata1 & i_csr_en;  
assign o_csr_pc = i_rdata1;  
  
end else begin  
    wire    rd = (i_ctrl_rd ) |  
                (i_alu_rd & i_rd_alu_en) |  
                (i_mem_rd);  
  
    assign    o_wdata0 = rd;
```

```

assign          o_wdata1 = 1'b0;

assign o_wreg0 = i_rd_waddr;
assign o_wreg1 = 5'd0;

assign      o_wen0 = i_cnt_en & rd_wen;
assign      o_wen1 = 1'b0;

/*
***** Read side *****
*/

assign o_rreg0 = i_rs1_raddr;
assign o_rreg1 = i_rs2_raddr;

assign o_rs1 = i_rdata0;
assign o_rs2 = i_rdata1;
assign o_csr = 1'b0;
assign o_csr_pc = 1'b0;
end // else: !if(WITH_CSR)
endgenerate
endmodule

```

serv_rf_ram

```

module serv_rf_ram
#(parameter width=0,
parameter csr_regs=4,
parameter depth=32*(32+csr_regs)/width)
(input wire i_clk,
input wire [$clog2(depth)-1:0] i_waddr,
input wire [width-1:0]      i_wdata,

```

```

input wire          i_wen,
input wire [$clog2(depth)-1:0] i_raddr,
output reg [width-1:0]    o_rdata);

reg [width-1:0]      memory [0:depth-1];

always @(posedge i_clk) begin
  if (i_wen)
    memory[i_waddr] <= i_wdata;
  o_rdata <= memory[i_raddr];
end

`ifdef SERV_CLEAR_RAM
integer i;
initial
  for (i=0;i<depth;i=i+1)
    memory[i] = {width{1'd0}};
`endif
endmodule

```

```

serv_rf_ram_if
`default_nettype none
module serv_rf_ram_if
#(parameter width=8,
  parameter reset_strategy="MINI",
  parameter csr_regs=4,
  parameter depth=32*(32+csr_regs)/width,
  parameter l2w = $clog2(width))
(
  //SERV side
  input wire          i_clk,

```

```

input wire           i_RST,
input wire           i_WREQ,
input wire           i_RREQ,
output wire          o_READY,
input wire [${clog2(32+CSR_REGS)-1}:0] i_WREG0,
input wire [${clog2(32+CSR_REGS)-1}:0] i_WREG1,
input wire           i_WEN0,
input wire           i_WEN1,
input wire           i_WDATA0,
input wire           i_WDATA1,
input wire [${clog2(32+CSR_REGS)-1}:0] i_RREG0,
input wire [${clog2(32+CSR_REGS)-1}:0] i_RREG1,
output wire          o_RDATA0,
output wire          o_RDATA1,
//RAM side
output wire [${clog2(DEPTH)-1}:0] o_WADDR,
output wire [WIDTH-1:0]          o_WDATA,
output wire           o_WEN,
output wire [${clog2(DEPTH)-1}:0] o_RADDR,
input wire [WIDTH-1:0] i_RDATA);

```

```

reg                  rgnt;
assign o_READY = rgnt | i_WREQ;
reg [4:0]      rcnt;

/*
***** Write side *****
*/

```

```

wire [4:0]      wcnt;
```

```

reg [WIDTH-2:0]  wdata0_r;
reg [WIDTH-1:0]  wdata1_r;
```

```

reg          wen0_r;
reg          wen1_r;
wire         wtrig0;
wire         wtrig1;

generate if (width == 2) begin
    assign wtrig0 = ~wcnt[0];
    assign wtrig1 = wcnt[0];
end else begin
    reg wtrig0_r;
    always @(posedge i_clk) wtrig0_r <= wtrig0;
    assign wtrig0 = (wcnt[12w-1:0] == {{12w-
1{1'b1}},1'b0});
    assign wtrig1 = wtrig0_r;
end
endgenerate

assign      o_wdata = wtrig1 ?
              wdata1_r :
              {i_wdata0, wdata0_r};

wire [${clog2(32+csr_regs)-1:0}] wreg = wtrig1 ?
i_wreg1 : i_wreg0;
generate if (width == 32)
assign o_waddr = wreg;
else
assign o_waddr = {wreg, wcnt[4:12w]};
endgenerate

assign o_wen = (wtrig0 & wen0_r) | (wtrig1 & wen1_r);

generate if (width > 2)

```

```

    always @(posedge i_clk) wdata0_r <= {i_wdata0,
wdata0_r[width-2:1]};
    else
        always @(posedge i_clk) wdata0_r <= i_wdata0;
endgenerate

assign wcnt = rcnt-3;

always @(posedge i_clk) begin
    wen0_r  <= i_wen0;
    wen1_r  <= i_wen1;

    wdata1_r <= {i_wdata1,wdata1_r[width-1:1]};

end

/*
***** Read side *****
*/

```



```

wire      rtrig0;
reg       rtrig1;

wire [$clog2(32+csr_regs)-1:0] rreg = rtrig0 ? i_rreg1 :
i_rreg0;
generate if (width == 32)
assign o_raddr = rreg;
else
    assign o_raddr = {rreg, rcnt[4:l2w]};
endgenerate

reg [width-1:0] rdata0;

```

```

reg [width-2:0] rdata1;

assign o_rdata0 = rdata0[0];
assign o_rdata1 = rtrig1 ? i_rdata[0] : rdata1[0];

assign rtrig0 = (rcnt[l2w-1:0] == 1);

reg          rreq_r;

generate if (width>2)
  always @(posedge i_clk) begin
    rdata1 <= {1'b0,rdata1[width-2:1]}; //Optimize?
    if (rtrig1)
      rdata1[width-2:0] <= i_rdata[width-1:1];
    end
  endelse
  always @(posedge i_clk) if (rtrig1) rdata1 <=
i_rdata[1];
endgenerate

always @(posedge i_clk) begin
  rtrig1 <= rtrig0;
  rcnt <= rcnt+5'd1;
  if (i_rreq)
    rcnt <= 5'd0;
  if (i_wreq)
    rcnt <= 5'd2;

  rreq_r <= i_rreq;
  rgnt <= rreq_r;

  rdata0 <= {1'b0,rdata0[width-1:1]};
  if (rtrig0)

```

```

rdata0 <= i_rdata;

if (i_RST) begin
    if (reset_strategy != "NONE") begin
        rgnt <= 1'b0;
        rreq_r <= 1'b0;
    end
end
end
endmodule

```

```

serv_state
module serv_state
#(parameter RESET_STRATEGY = "MINI",
  parameter [0:0] WITH_CSR = 1)
(
  input wire      i_clk,
  input wire      i_RST,
  input wire      i_new_irq,
  input wire      i_dbus_ack,
  output wire     o_ibus_cyc,
  input wire      i_ibus_ack,
  output wire     o_rf_rreq,
  output wire     o_rf_wreq,
  input wire      i_rf_ready,
  output wire     o_rf_rd_en,
  input wire      i_cond_branch,
  input wire      i_bne_or_bge,
  input wire      i_alu_cmp,

```

```
input wire      i_branch_op,
input wire      i_mem_op,
input wire      i_shift_op,
input wire      i_sh_right,
input wire      i_slt_op,
input wire      i_e_op,
input wire      i_rd_op,
output wire     o_init,
output wire     o_cnt_en,
output wire     o_cnt0,
output wire     o_cnt0to3,
output wire     o_cnt12to31,
output wire     o_cnt1,
output wire     o_cnt2,
output wire     o_cnt3,
output wire     o_cnt7,
output wire     o_ctrl_pc_en,
output reg      o_ctrl_jump,
output wire     o_ctrl_trap,
input wire      i_ctrl_misalign,
input wire      i_sh_done,
input wire      i_sh_done_r,
output wire     o_dbus_cyc,
output wire [1:0] o_mem_bytectn,
input wire      i_mem_misalign,
output reg      o_cnt_done,
output wire     o_bufreg_en);
```

```
reg      stage_two_req;
reg      init_done;
wire    misalign_trap_sync;
```

```
reg [4:2] o_cnt;
```

```

reg [3:0] o_cnt_r;

reg          ibus_cyc;
//Update PC in RUN or TRAP states
assign o_ctrl_pc_en = o_cnt_en & !o_init;

assign o_cnt_en = |o_cnt_r;

assign o_mem_bytetcnt = o_cnt[4:3];

assign o_cnt0to3  = (o_cnt[4:2] == 3'd0);
assign o_cnt12to31 = (o_cnt[4] | (o_cnt[3:2] == 2'b11));
assign o_cnt0 = (o_cnt[4:2] == 3'd0) & o_cnt_r[0];
assign o_cnt1 = (o_cnt[4:2] == 3'd0) & o_cnt_r[1];
assign o_cnt2 = (o_cnt[4:2] == 3'd0) & o_cnt_r[2];
assign o_cnt3 = (o_cnt[4:2] == 3'd0) & o_cnt_r[3];
assign o_cnt7 = (o_cnt[4:2] == 3'd1) & o_cnt_r[3];

//Take branch for jump or branch instructions (opcode
== 1x0xx) if
  //a) It's an unconditional branch (opcode[0] == 1)
  //b) It's a conditional branch (opcode[0] == 0) of type
    beq,blt,bltu (funct3[0] == 0) and ALU compare is true
  //c) It's a conditional branch (opcode[0] == 0) of type
    bne,bge,bgeu (funct3[0] == 1) and ALU compare is false
  //Only valid during the last cycle of INIT, when the
  branch condition has
  //been calculated.

  wire   take_branch = i_branch_op & (!i_cond_branch |
    (i_alu_cmp^i_bne_or_bge));

//slt*, branch/jump, shift, load/store

```

```
wire two_stage_op = i_slt_op | i_mem_op | i_branch_op  
| i_shift_op;
```

```
assign o_dbus_cyc = !o_cnt_en & init_done &  
i_mem_op & !i_mem_misalign;
```

```
//Prepare RF for reads when a new instruction is fetched  
// or when stage one caused an exception (rreq implies a  
write request too)
```

```
assign o_rf_rreq = i_ibus_ack | (stage_two_req &  
misalign_trap_sync);
```

```
//Prepare RF for writes when everything is ready to  
enter stage two
```

```
// and the first stage didn't cause a misalign exception  
assign o_rf_wreq = !misalign_trap_sync &  
((i_shift_op & (i_sh_done | !i_sh_right)) &  
!o_cnt_en & init_done) |  
((i_mem_op & i_dbus_ack) |  
(stage_two_req & (i_slt_op |  
i_branch_op)));
```

```
assign o_rf_rd_en = i_rd_op & !o_init;
```

```
/*
```

```
bufreg is used during mem. branch and shift operations
```

```
mem : bufreg is used for dbus address. Shift in data  
during phase 1.
```

```
Shift out during phase 2 if there was an  
misalignment exception.
```

branch : Shift in during phase 1. Shift out during phase
2

shift : Shift in during phase 1. Continue shifting
between phases (except
for the first cycle after init). Shift out during phase
2

*/

```
assign o_bufreg_en = (o_cnt_en & (o_init | o_ctrl_trap |  
i_branch_op)) | (i_shift_op & !stage_two_req &  
(i_sh_right | i_sh_done_r));
```

```
assign o_ibus_cyc = ibus_cyc & !i_RST;
```

```
assign o_init = two_stage_op & !i_new_irq &  
!init_done;
```

```
always @(posedge i_clk) begin  
    //ibus_cyc changes on three conditions.  
    //1. i_RST is asserted. Together with the async gating  
    above, o_ibus_cyc
```

// will be asserted as soon as the reset is released.

This is how the

// first instruction is fetched

//2. o_CNT_DONE and o_CTRL_PC_EN are asserted. This
means that SERV just

// finished updating the PC, is done with the current
instruction and

// o_IBUS_CYC gets asserted to fetch a new instruction

//3. When i_IBUS_ACK, a new instruction is fetched and
o_IBUS_CYC gets

// deasserted to finish the transaction

```
if (i_IBUS_ACK | o_CNT_DONE | i_RST)
```

```

ibus_cyc <= o_ctrl_pc_en | i_RST;
if (o_CNT_DONE) begin
    init_DONE <= o_INIT & !init_DONE;
    o_CTRL_JUMP <= o_INIT & take_BRANCH;
end
o_CNT_DONE <= (o_CNT[4:2] == 3'b111) & o_CNT_R[2];
//Need a strobe for the first cycle in the IDLE state
after INIT
stage_TWO_REQ <= o_CNT_DONE & o_INIT;
/*

```

Because SERV is 32-bit bit-serial we need a counter
than can count 0-31

to keep track of which bit we are currently
processing. o_CNT and o_CNT_R
are used together to create such a counter.

The top three bits (o_CNT) are implemented as a
normal counter, but

instead of the two LSB, o_CNT_R is a 4-bit shift
register which loops 0-3

When o_CNT_R[3] is 1, o_CNT will be increased.

The counting starts when the core is idle and the
i_RF_READY signal

comes in from the RF module by shifting in the
i_RF_READY bit as LSB of

the shift register. Counting is stopped by using
o_CNT_DONE to block the

bit that was supposed to be shifted into bit 0 of
o_CNT_R.

There are two benefit of doing the counter this way

1. We only need to check four bits instead of five when we want to check

if the counter is at a certain value. For 4-LUT architectures this means

we only need one LUT instead of two for each comparison.

2. We don't need a separate enable signal to turn on and off the counter

between stages, which saves an extra FF and a unique control signal. We

just need to check if o_cnt_r is not zero to see if the counter is

currently running

*/

```
o_cnt <= o_cnt + {2'd0,o_cnt_r[3]};  
o_cnt_r <= {o_cnt_r[2:0],(o_cnt_r[3] & !o_cnt_done)  
| (i_rf_ready & !o_cnt_en)};  
if (i_RST) begin  
    if (RESET_STRATEGY != "NONE") begin  
        o_CNT <= 3'd0;  
        init_DONE <= 1'b0;  
        o_CTRL_JUMP <= 1'b0;  
        o_CNT_R <= 4'b0000;  
    end  
end  
end
```

```
assign o_CTRL_TRAP = WITH_CSR & (i_E_OP | i_NEW_IRQ  
| misalign_trap_sync);
```

generate

```
if (WITH_CSR) begin
```

```

    reg misalign_trap_sync_r;

    //trap_pending is only guaranteed to have correct
    value during the
        // last cycle of the init stage
        wire trap_pending = WITH_CSR & ((take_branch
& i_ctrl_misalign) |
                                (i_mem_op  &
i_mem_misalign));

    always @(posedge i_clk) begin
        if (o_cnt_done)
            misalign_trap_sync_r <= trap_pending & o_init;
        if (i_RST)
            if (RESET_STRATEGY != "NONE")
                misalign_trap_sync_r <= 1'b0;
        end
        assign misalign_trap_sync = misalign_trap_sync_r;
    end else
        assign misalign_trap_sync = 1'b0;
    endgenerate
endmodule

```

serv_top

```

`default_nettype none

module serv_top
#(parameter WITH_CSR = 1,
parameter PRE_REGISTER = 1,

```

```

parameter RESET_STRATEGY = "MINI",
parameter RESET_PC = 32'd0)
(
    input wire          clk,
    input wire          i_rst,
    input wire          i_timer_irq,
`ifdef RISCV_FORMAL
    output reg          rvfi_valid = 1'b0,
    output reg [63:0]   rvfi_order = 64'd0,
    output reg [31:0]   rvfi_insn = 32'd0,
    output reg          rvfi_trap = 1'b0,
    output reg          rvfi_halt = 1'b0,
    output reg          rvfi_intr = 1'b0,
    output reg [1:0]    rvfi_mode = 2'b11,
    output reg [1:0]    rvfi_ixl = 2'b01,
    output reg [4:0]    rvfi_rs1_addr,
    output reg [4:0]    rvfi_rs2_addr,
    output reg [31:0]   rvfi_rs1_rdata,
    output reg [31:0]   rvfi_rs2_rdata,
    output reg [4:0]    rvfi_rd_addr,
    output reg [31:0]   rvfi_rd_wdata,
    output reg [31:0]   rvfi_pc_rdata,
    output reg [31:0]   rvfi_pc_wdata,
    output reg [31:0]   rvfi_mem_addr,
    output reg [3:0]    rvfi_mem_rmask,
    output reg [3:0]    rvfi_mem_wmask,
    output reg [31:0]   rvfi_mem_rdata,
    output reg [31:0]   rvfi_mem_wdata,
`endif
//RF Interface
    output wire         o_rf_rreq,
    output wire         o_rf_wreq,
    input wire          i_rf_ready,

```

```
output wire [4+WITH_CSR:0] o_wreg0,  
output wire [4+WITH_CSR:0] o_wreg1,  
output wire          o_wen0,  
output wire          o_wen1,  
output wire          o_wdata0,  
output wire          o_wdata1,  
output wire [4+WITH_CSR:0] o_rreg0,  
output wire [4+WITH_CSR:0] o_rreg1,  
input wire          i_rdata0,  
input wire          i_rdata1,  
  
output wire [31:0]      o_ibus_adr,  
output wire          o_ibus_cyc,  
input wire [31:0]      i_ibus_rdt,  
input wire          i_ibus_ack,  
output wire [31:0]      o_dbus_adr,  
output wire [31:0]      o_dbus_dat,  
output wire [3:0]       o_dbus_sel,  
output wire          o_dbus_we ,  
output wire          o_dbus_cyc,  
input wire [31:0]      i_dbus_rdt,  
input wire          i_dbus_ack);  
  
wire [4:0]   rd_addr;  
wire [4:0]   rs1_addr;  
wire [4:0]   rs2_addr;  
  
wire [3:0]   immdec_ctrl;  
wire [3:0]   immdec_en;  
  
wire      sh_right;  
wire      bne_or_bge;  
wire      cond_branch;
```

```
wire    e_op;
wire    ebreak;
wire    branch_op;
wire    mem_op;
wire    shift_op;
wire    slt_op;
wire    rd_op;

wire    rd_alu_en;
wire    rd_csr_en;
wire    ctrl_rd;
wire    alu_rd;
wire    mem_rd;
wire    csr_rd;

wire    ctrl_pc_en;
wire    jump;
wire    jal_or_jalr;
wire    utype;
wire    mret;
wire    imm;
wire    trap;
wire    pc_rel;

wire    init;
wire    cnt_en;
wire    cnt0to3;
wire    cnt12to31;
wire    cnt0;
wire    cnt1;
wire    cnt2;
wire    cnt3;
wire    cnt7;
```

```
    wire      cnt_done;  
  
    wire      bufreg_en;  
    wire      bufreg_sh_signed;  
    wire      bufreg_rs1_en;  
    wire      bufreg_imm_en;  
    wire      bufreg_clr_lsb;  
    wire      bufreg_q;  
  
    wire      alu_sub;  
    wire [1:0]   alu_bool_op;  
    wire      alu_cmp_eq;  
    wire      alu_cmp_sig;  
    wire      alu_cmp;  
    wire [2:0]   alu_rd_sel;  
  
    wire      rs1;  
    wire      rs2;  
    wire      rd_en;  
  
    wire      op_b_source;  
  
    wire      mem_signed;  
    wire      mem_word;  
    wire      mem_half;  
    wire [1:0]   mem_bytectn;  
    wire      mem_sh_done;  
    wire      mem_sh_done_r;  
  
    wire      mem_misalign;  
  
    wire      bad_pc;
```

```

wire      csr_mstatus_en;
wire      csr_mie_en;
wire      csr_mcause_en;
wire [1:0]   csr_source;
wire      csr_imm;
wire      csr_d_sel;
wire      csr_en;
wire [1:0]   csr_addr;
wire      csr_pc;
wire      csr_imm_en;
wire      csr_in;
wire      rf_csr_out;

wire      new_irq;

wire [1:0]  lsb;

wire      op_b = op_b_source ? rs2 : imm;

serv_state
#(.RESET_STRATEGY (RESET_STRATEGY),
.WITH_CSR (WITH_CSR))
state
(
  .i_clk (clk),
  .i_rst    (i_rst),
  //State
  .i_new_irq    (new_irq),
  .i_alu_cmp    (alu_cmp),
  .o_init      (init),
  .o_cnt_en    (cnt_en),
  .o_cnt0to3   (cnt0to3),

```

```
.o_cnt12to31  (cnt12to31),
.o_cnt0       (cnt0),
.o_cnt1       (cnt1),
.o_cnt2       (cnt2),
.o_cnt3       (cnt3),
.o_cnt7       (cnt7),
.o_cnt_done   (cnt_done),
.o_bufreg_en  (bufreg_en),
.o_ctrl_pc_en (ctrl_pc_en),
.o_ctrl_jump  (jump),
.o_ctrl_trap  (trap),
.i_ctrl_misalign(lsb[1]),
.i_sh_done    (mem_sh_done),
.i_sh_done_r  (mem_sh_done_r),
.o_mem_bytecnt (mem_bytecnt),
.i_mem_misalign (mem_misalign),
//Control
.i_bne_or_bge (bne_or_bge),
.i_cond_branch (cond_branch),
.i_branch_op   (branch_op),
.i_mem_op      (mem_op),
.i_shift_op    (shift_op),
.i_sh_right   (sh_right),
.i_slt_op      (slt_op),
.i_e_op        (e_op),
.i_rd_op       (rd_op),
//External
.o_dbus_cyc   (o_dbus_cyc),
.i_dbus_ack   (i_dbus_ack),
.o_ibus_cyc   (o_ibus_cyc),
.i_ibus_ack   (i_ibus_ack),
//RF Interface
.o_rf_rreq    (o_rf_rreq),
```

```

.o_rf_wreq    (o_rf_wreq),
.i_rf_ready   (i_rf_ready),
.o_rf_rd_en   (rd_en));

serv_decode
#(.PRE_REGISTER (PRE_REGISTER))
decode
(
.clk (clk),
//Input
.i_wb_rdt      (i_ibus_rdt[31:2]),
.i_wb_en       (i_ibus_ack),
//To state
.o_bne_or_bge  (bne_or_bge),
.o_cond_branch  (cond_branch),
.o_e_op         (e_op),
.o_ebreak       (ebreak),
.o_branch_op    (branch_op),
.o_mem_op       (mem_op),
.o_shift_op     (shift_op),
.o_slt_op       (slt_op),
.o_rd_op        (rd_op),
.o_sh_right    (sh_right),
//To bufreg
.o_bufreg_rs1_en (bufreg_rs1_en),
.o_bufreg_imm_en (bufreg_imm_en),
.o_bufreg_clr_lsb (bufreg_clr_lsb),
.o_bufreg_sh_signed (bufreg_sh_signed),
//To ctrl
.o_ctrl_jal_or_jalr (jal_or_jalr),
.o_ctrl_utype   (utype),
.o_ctrl_pc_rel  (pc_rel),
.o_ctrl_mret    (mret),

```

```

//To alu
.o_op_b_source    (op_b_source),
.o_alu_sub        (alu_sub),
.o_alu_bool_op   (alu_bool_op),
.o_alu_cmp_eq    (alu_cmp_eq),
.o_alu_cmp_sig   (alu_cmp_sig),
.o_alu_rd_sel    (alu_rd_sel),
//To mem IF
.o_mem_cmd        (o_dbus_we),
.o_mem_signed    (mem_signed),
.o_mem_word      (mem_word),
.o_mem_half      (mem_half),
//To CSR
.o_csr_en         (csr_en),
.o_csr_addr       (csr_addr),
.o_csr_mstatus_en (csr_mstatus_en),
.o_csr_mie_en    (csr_mie_en),
.o_csr_mcause_en (csr_mcause_en),
.o_csr_source     (csr_source),
.o_csr_d_sel      (csr_d_sel),
.o_csr_imm_en    (csr_imm_en),
//To top
.o_immdec_ctrl   (immdec_ctrl),
.o_immdec_en     (immdec_en),
.o_rd_csr_en     (rd_csr_en),
.o_rd_alu_en     (rd_alu_en));

```

```

serv_immdec immdec
(
.i_clk    (clk),
//State
.i_cnt_en  (cnt_en),
.i_cnt_done (cnt_done),

```

```
//Control
.i_immdec_en    (immdec_en),
.i_csr_imm_en (csr_imm_en),
.i_ctrl      (immdec_ctrl),
.o_rd_addr   (rd_addr),
.o_rs1_addr  (rs1_addr),
.o_rs2_addr  (rs2_addr),
//Data
.o_csr_imm  (csr_imm),
.o_imm     (imm),
//External
.i_wb_en    (i_ibus_ack),
.i_wb_rdt   (i_ibus_rdt[31:7]));
```

```
serv_bufreg bufreg
(
  .i_clk  (clk),
//State
  .i_cnt0 (cnt0),
  .i_cnt1 (cnt1),
  .i_en   (bufreg_en),
  .i_init (init),
  .o_lsb   (lsb),
//Control
  .i_sh_signed (bufreg_sh_signed),
  .i_rs1_en   (bufreg_rs1_en),
  .i_imm_en   (bufreg_imm_en),
  .i_clr_lsb  (bufreg_clr_lsb),
//Data
  .i_rs1   (rs1),
  .i_imm   (imm),
  .o_q     (bufreg_q),
//External
```

```

.o_dbus_adr (o_dbus_adr));

serv_ctrl
#(.RESET_PC (RESET_PC),
.RESET_STRATEGY (RESET_STRATEGY),
.WITH_CSR (WITH_CSR))
ctrl
(
    .clk      (clk),
    .i_rst    (i_rst),
    //State
    .i_pc_en  (ctrl_pc_en),
    .i_cnt12to31 (cnt12to31),
    .i_cnt0   (cnt0),
    .i_cnt2   (cnt2),
    //Control
    .i_jump   (jump),
    .i_jal_or_jalr (jal_or_jalr),
    .i_utype  (utype),
    .i_pc_rel (pc_rel),
    .i_trap   (trap | mret),
    //Data
    .i_imm    (imm),
    .i_buf    (bufreg_q),
    .i_csr_pc (csr_pc),
    .o_rd     (ctrl_rd),
    .o_bad_pc (bad_pc),
    //External
    .o_ibus_adr (o_ibus_adr));

serv_alu alu
(
    .clk      (clk),

```

```
//State
.i_en    (cnt_en),
.i_cnt0  (cnt0),
.o_cmp   (alu_cmp),
//Control
.i_sub   (alu_sub),
.i_bool_op (alu_bool_op),
.i_cmp_eq (alu_cmp_eq),
.i_cmp_sig (alu_cmp_sig),
.i_rd_sel (alu_rd_sel),
//Data
.i_rs1   (rs1),
.i_op_b   (op_b),
.i_buf    (bufreg_q),
.o_rd     (alu_rd));
```

serv_rf_if

#(.WITH_CSR (WITH_CSR))

rf_if

(//RF interface

```
.i_cnt_en  (cnt_en),
.o_wreg0   (o_wreg0),
.o_wreg1   (o_wreg1),
.o_wen0    (o_wen0),
.o_wen1    (o_wen1),
.o_wdata0  (o_wdata0),
.o_wdata1  (o_wdata1),
.o_rreg0   (o_rreg0),
.o_rreg1   (o_rreg1),
.i_rdata0  (i_rdata0),
.i_rdata1  (i_rdata1),
```

//Trap interface

```

.i_trap    (trap),
.i_mret    (mret),
.i_mepc    (o_ibus_adr[0]),
.i_mem_op  (mem_op),
.i_bufreg_q (bufreg_q),
.i_bad_pc  (bad_pc),
.o_csr_pc  (csr_pc),
//CSR write port
.i_csr_en  (csr_en),
.i_csr_addr (csr_addr),
.i_csr     (csr_in),
//RD write port
.i_rd_wen  (rd_en),
.i_rd_waddr (rd_addr),
.i_ctrl_rd (ctrl_rd),
.i_alu_rd   (alu_rd),
.i_rd_alu_en (rd_alu_en),
.i_csr_rd   (csr_rd),
.i_rd_csr_en (rd_csr_en),
.i_mem_rd   (mem_rd),

//RS1 read port
.i_rs1_raddr (rs1_addr),
.o_rs1      (rs1),
//RS2 read port
.i_rs2_raddr (rs2_addr),
.o_rs2      (rs2),

//CSR read port
.o_csr      (rf_csr_out));

serv_mem_if
#(.WITH_CSR (WITH_CSR))

```

```
mem_if
(
    .i_clk      (clk),
    //State
    .i_en       (cnt_en),
    .i_init     (init),
    .i_cnt_done (cnt_done),
    .i_bytectn (mem_bytectn),
    .i_lsb      (lsb),
    .o_misalign (mem_misalign),
    .o_sh_done  (mem_sh_done),
    .o_sh_done_r (mem_sh_done_r),
    //Control
    .i_mem_op   (mem_op),
    .i_shift_op (shift_op),
    .i_signed   (mem_signed),
    .i_word     (mem_word),
    .i_half     (mem_half),
    //Data
    .i_op_b     (op_b),
    .o_rd       (mem_rd),
    //External interface
    .o_wb_dat   (o_dbus_dat),
    .o_wb_sel   (o_dbus_sel),
    .i_wb_rdt   (i_dbus_rdt),
    .i_wb_ack   (i_dbus_ack));
```

```
generate
  if (WITH_CSR) begin
    serv_csr csr
    (
        .i_clk      (clk),
        //State
```

```

.i_init      (init),
.i_en       (cnt_en),
.i_cnt0to3  (cnt0to3),
.i_cnt3     (cnt3),
.i_cnt7     (cnt7),
.i_cnt_done (cnt_done),
.i_mem_op   (mem_op),
.i_mtip    (i_timer_irq),
.i_trap    (trap),
.o_new_irq (new_irq),
//Control
.i_e_op     (e_op),
.i_ebreak   (ebreak),
.i_mem_cmd  (o_dbus_we),
.i_mstatus_en (csr_mstatus_en),
.i_mie_en   (csr_mie_en ),
.i_mcause_en (csr_mcause_en ),
.i_csr_source (csr_source),
.i_mret     (mret),
.i_csr_d_sel (csr_d_sel),
//Data
.i_rf_csr_out (rf_csr_out),
.o_csr_in   (csr_in),
.i_csr_imm  (csr_imm),
.i_rs1      (rs1),
.o_q        (csr_rd));
end else begin
    assign csr_in = 1'b0;
    assign csr_rd = 1'b0;
    assign new_irq = 1'b0;
end
endgenerate

```

```

`ifdef RISCV_FORMAL
    reg [31:0] pc = RESET_PC;

    wire rs_en = (branch_op|mem_op|shift_op|slt_op) ? init
    : ctrl_pc_en;

always @(posedge clk) begin
    rvfi_valid <= cnt_done & ctrl_pc_en & !i_rst;
    rvfi_order <= rvfi_order + {63'd0,rvfi_valid};
    if (o_ibus_cyc & i_ibus_ack)
        rvfi_insn <= i_ibus_rdt;
    if (o_wen0)
        rvfi_rd_wdata <= {o_wdata0,rvfi_rd_wdata[31:1]};
    if (cnt_done & ctrl_pc_en) begin
        rvfi_pc_rdata <= pc;
        if (!(rd_en & (|rd_addr))) begin
            rvfi_rd_addr <= 5'd0;
            rvfi_rd_wdata <= 32'd0;
        end
    end
    rvfi_trap <= trap;
    if (rvfi_valid) begin
        rvfi_trap <= 1'b0;
        pc <= rvfi_pc_wdata;
    end

    rvfi_halt <= 1'b0;
    rvfi_intr <= 1'b0;
    rvfi_mode <= 2'd3;
    rvfi_ixl = 2'd1;
    if (i_rf_ready) begin
        rvfi_rs1_addr <= rs1_addr;

```

```

    rvfi_rs2_addr <= rs2_addr;
    rvfi_rd_addr <= rd_addr;
end
if (rs_en) begin
    rvfi_rs1_rdata <= {rs1,rvfi_rs1_rdata[31:1]};
    rvfi_rs2_rdata <= {rs2,rvfi_rs2_rdata[31:1]};
end

if (i_dbus_ack) begin
    rvfi_mem_addr <= o_dbus_adr;
    rvfi_mem_rmask <= o_dbus_we ? 4'b0000 :
o_dbus_sel;
    rvfi_mem_wmask <= o_dbus_we ? o_dbus_sel :
4'b0000;
    rvfi_mem_rdata <= i_dbus_rdt;
    rvfi_mem_wdata <= o_dbus_dat;
end
if (i_ibus_ack) begin
    rvfi_mem_rmask <= 4'b0000;
    rvfi_mem_wmask <= 4'b0000;
end
/* verilator lint_off COMBDLY */
always @(o_ibus_adr)
rvfi_pc_wdata <= o_ibus_adr;
/* verilator lint_on COMBDLY */

`endif

endmodule
`default_nettype wire

```

PDK Details

There are seven standard cell libraries provided directly by the SkyWater Technology foundry available for use on **SKY130** designs, which differ in intended applications and come in three separate cell heights.

Libraries:

sky130_fd_sc_hs (high speed)

sky130_fd_sc_ms (medium speed)

sky130_fd_sc_ls (low speed)

sky130_fd_sc_lp (low power)

These are compatible in size, with a 0.48 x 3.33um site, equivalent to about 11 met1 tracks.

Libraries **sky130_fd_sc_hd** (high density) and **sky130_fd_sc_hdll** (high density, low leakage) contain standard cells that are smaller, utilizing a 0.46 x 2.72um site, equivalent to 9 met1 tracks.

The **sky130_fd_sc_hvl** (high voltage) library contains 5V devices and utilizes a 0.48 x 4.07um site, or 14 met1 tracks.

sky130_fd_sc_hd - High Density Standard Cell Library

The sky130_fd_sc_hd library is designed for high density.

Compared to **sky130_fd_sc_ls**, this library enables higher routed gated density, lower dynamic power consumption, and comparable timing and leakage power. As a trade-off it has lower drive strength and does not support any drop in replacement medium or high speed library.

- **sky130_fd_sc_hd** includes clock-gating cells to reduce active power during non-sleep modes.
- Latches and flip-flops have scan equivalents to enable scan chain creation.
- Multi-voltage domain library cells are provided.
- Routed Gate Density is 160 kGates/mm² or better.
- leakage @ttleak_1.80v_25C (no body bias) is 0.86 nA / kGate
- sky130_fd_sc_XX_buf_16 max cap (ss_1.60v_-40C, in/out tran=1.5ns) is 0.746 pF
- Body Bias-able

sky130_fd_sc_hdll - High Density, Low Leakage Standard Cell Library

The sky130_fd_sc_hdll library is a low leakage high density standard cell library.

Compared to **sky130_fd_sc_hd**, this library enables 5-10X lower leakage power, but the same X, Y pin grids, routing layer pitches, and cell height.

Blocks should be DRC clean when intermingled with sky130_fd_sc_hd cells.

Raw gate density (number of **sky130_fd_sc_hdll_nand2_1** gates able to fit in 1mm²) for **sky130_fd_sc_hd** is 266kGates/mm² and 200kGates/mm² for **sky130_fd_sc_hdll**.

- Includes integrating clock-gating cells to reduce active power during non-sleep modes
- Latches and flip-flops in the library have a scan equivalent implementation to enable scan chain creation and testing supported by the synthesis tools
- Multi-voltage domain library cells are provided
- Routed Gate Density is 120 kGates/mm²
- leakage @t_{leak}_1.80v_25C (no body bias) is 0.08 nA / kGate
- sky130_fd_sc_XX_buf_16 max cap (ss_1.60v_-40C, in/out tran=1.5ns) < 1 pF
- Multi-Voltage Design Support
- Body Bias-able

sky130_fd_sc_ms - Low Voltage (<2.0V), Medium Speed, Standard Cell Library

sky130_fd_sc_ms is drop-in compatible with sky130_fd_sc_ls or sky130_fd_sc_hs libraries for cells of the same function and drive strength. sky130_fd_sc_ms cells have medium speed and leakage.

sky130_fd_sc_ms is implemented with low voltage transistors; timing and power models are valid from 1.60V up to 1.95V. All cells are functional at 1.2V.

The low to high level shifter cells are capable of shifting from 1.2V to 1.95V.

- The library supports low leakage sleep mode via state retention flops
- Includes integrating clock-gating cells to reduce active power during non-sleep modes
- Latches and flip-flops in the library have a scan equivalent implementation to enable scan chain creation and testing supported by the synthesis tools
- Library details:
 - Inverters and buffers: 48
 - AND, OR, NAND, NOR gates: 86
 - Exclusive-OR and Exclusive-NOR: 12
 - Inverted And-Or and Inverted Or-And: 71
 - And-Or and Or-And: 68
 - Adders, Comparators and Multiplexers: 33
 - Latches and filp-flops: 68
 - Low Power Flow Cells: 42
 - UDB custom cells: 17

sky130_fd_sc_lp - Low Voltage (<2.0V), Low Power, Standard Cell Library

sky130_fd_sc_lp is the largest of the **SKY130** standard cell libraries at nearly 750 cells. All logic cells are implemented with low voltage transistors and should be powered within the limits of those transistors. Specifically, the timing and power models are valid from 1.55V up to 2.0V.

- **sky130_fd_sc_lp** supports low leakage sleep mode via sleep transistors

- Includes integrating clock-gating cells to reduce active power during non-sleep modes
- Latches and flip-flops have scan equivalents to enable scan chain creation
- Larger Library size:
 - Inverters, Buffers: 108
 - AND, OR, NAND, NOR gates: 159
 - Exclusive-OR, Exclusive-NOR: 16
 - AND-OR-Inverted, OR-AND-Inverted: 138
 - AND-OR, OR-AND: 132
 - Adders, Comparators, Multiplexors: 59
 - Custom Power gating, bus cells: 43
 - Latches and flip-flops: 92

sky130_fd_sc_hvl - High Voltage (5V), Standard Cell Library

The **sky130_fd_sc_hvl** library has the smallest cell count of the **SKY130** standard cell libraries, but is the only one that enables 5V tolerant logic blocks. All logic cells are implemented with 5V tolerant transistors; timing and power models are valid from 1.65V to 5.5V. The low voltage to high voltage level shifter is functional shifting from 1.2V to 5.5V.

Raw gate density (number of **sky130_fd_sc_hvl_nand2_1** gates able to fit in 1mm²) should be 170kGates/mm².

Routed should be ≥ 100 k Gates/mm². Due to the gate length for these high voltage transistors, the actual gate density is lower than 170kGates/mm². The size of a 2-input NAND gate in this library is actually 5 grids wide,

whereas the 170k calculation is based on a gate that is 3 grids wide. With a 5 grid wide gate, the raw gate density is 102kGates/mm².

OpenLane Flow

```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane$ export PDK_ROOT=/home/badboy07/Desktop/openlane/pdks
```

The above command is for exporting Process Design Kit (PDK) ie sky130 in OpenLane.

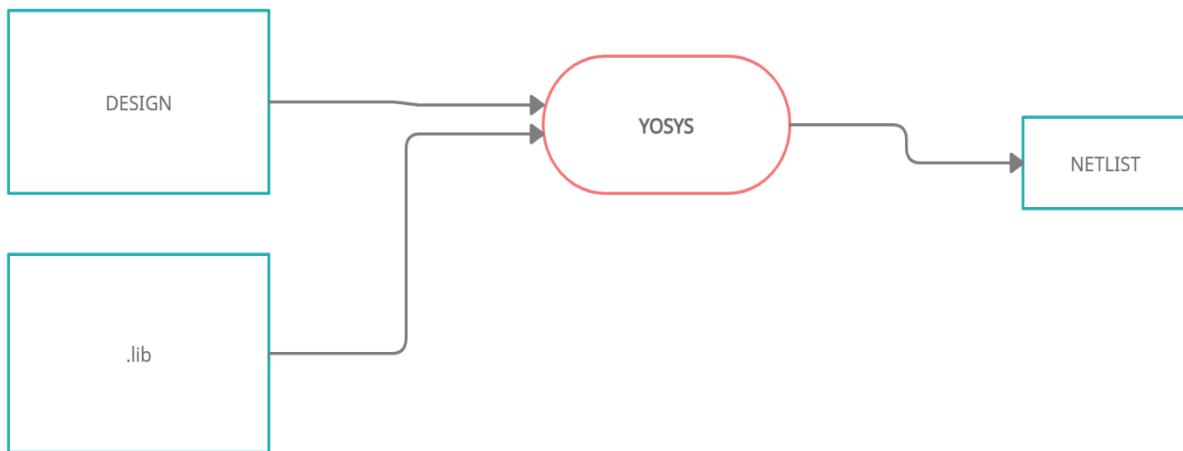
```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane$ sudo make mount
```

This command is for mounting OpenLane directory on Docker.

```
[sudo] password for badboy07:  
cd /home/badboy07/Desktop/OpenLane && \  
    docker run -it --rm -v /home/badboy07/Desktop/OpenLane:/openLANE_flow  
-v /home/badboy07/Desktop/OpenLane/pdks:/home/badboy07/Desktop/OpenLane/pdks -  
e PDK_ROOT=/home/badboy07/Desktop/OpenLane/pdks -u 0:0 efabless/openlane:2021.  
08.02_05.21.44
```

This what we see after successfully mounting OpenLane on Docker.

Logic Synthesis Tool Flow



The above image shows the flow of Yosys. Yosys takes RTL code and .lib files as input and netlist as output. Netlist is the standard cell representation of the design.

Yosys flow

```
# read design  
read_verilog serv_rf_top.v  
  
# elaborate design hierarchy  
hierarchy -check -top serv_rf_top
```

```
# the high-level stuff  
proc;  
opt;  
fsm;
```

```
opt;
memory;
opt

# mapping to internal cell library
techmap;
op

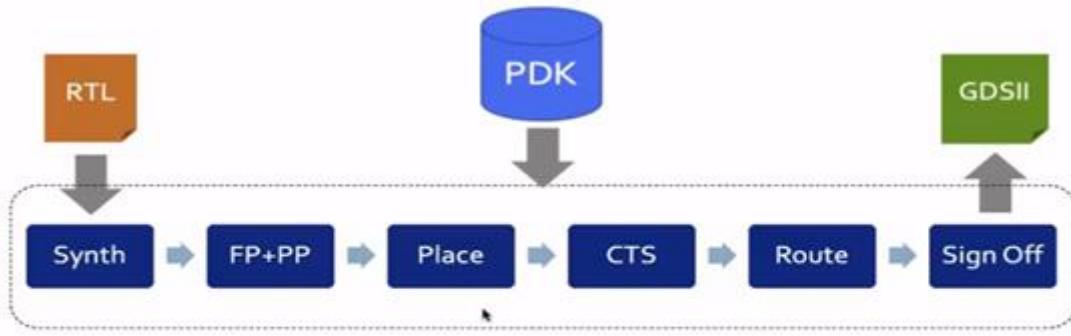
# mapping flip-flops to mycells.lib
dfflibmap -liberty
home/badboy07/Desktop/vsdflow/work/tools/openlane_working_dir/pdks/open_pdks/sky130/sky130A/libs.ref/sky130_fd_sc_hs/lib/sky130_fd_sc_hs_ff_100C_1v95.lib

# mapping logic to mycells.lib
abc -liberty
home/badboy07/Desktop/vsdflow/work/tools/openlane_working_dir/pdks/open_pdks/sky130/sky130A/libs.ref/sky130_fd_sc_hs/lib/ sky130_fd_sc_hs_ff_100C_1v95.lib

# cleanup
clean

# write synthesized design
write_verilog serv_rf_top.synthesis
```

OpenLane automated flow

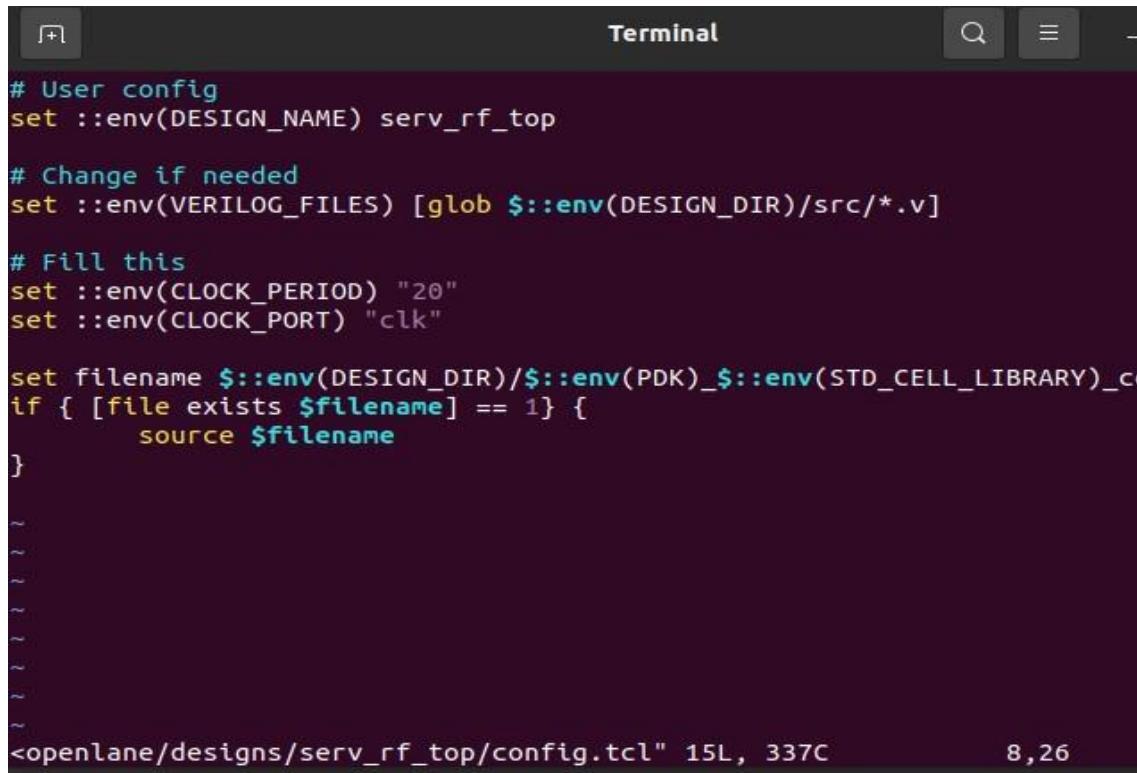


- Synthesis
- Floor/Power Planning
- Placement
- Clock Tree Synthesis
- Routing
- Sign Off

`./flow.tcl -design serv_rf_top -init_design_config`

The above command `flow.tcl` has commands for running OpenLane flow so here we run the configuration flow

Then we need to edit required variable in `config.tcl`



The screenshot shows a terminal window with a dark background and light-colored text. The title bar says "Terminal". The script content is as follows:

```
# User config
set ::env(DESIGN_NAME) serv_rf_top

# Change if needed
set ::env(VERILOG_FILES) [glob $::env(DESIGN_DIR)/src/*.v]

# Fill this
set ::env(CLOCK_PERIOD) "20"
set ::env(CLOCK_PORT) "clk"

set filename $::env(DESIGN_DIR)/$::env(PDK)_$::env(STD_CELL_LIBRARY)_c
if { [file exists $filename] == 1} {
    source $filename
}

~
~
~
~
~
~
~
~
<openlane/designs/serv_rf_top/config.tcl" 15L, 337C           8,26
```

Then we are giving the below commands to run the design automatically through ./flow.tcl command

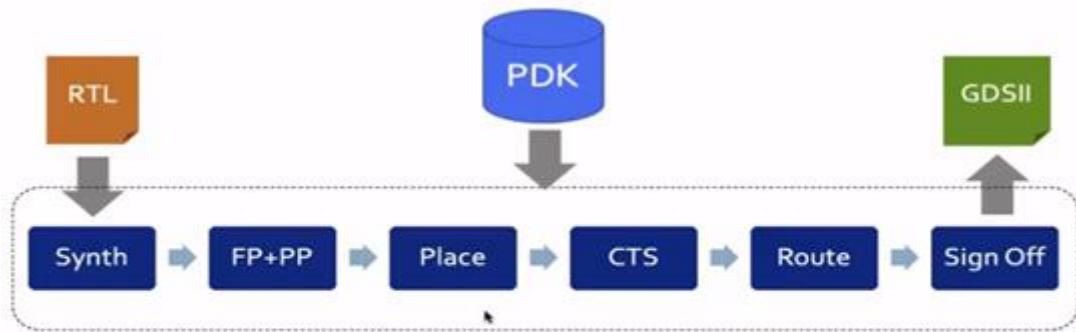
./flow.tcl -design serv_rf_top -init_design_config -src /serv_rf_top.v

In the above command we are passing source file as our input which is in Verilog.

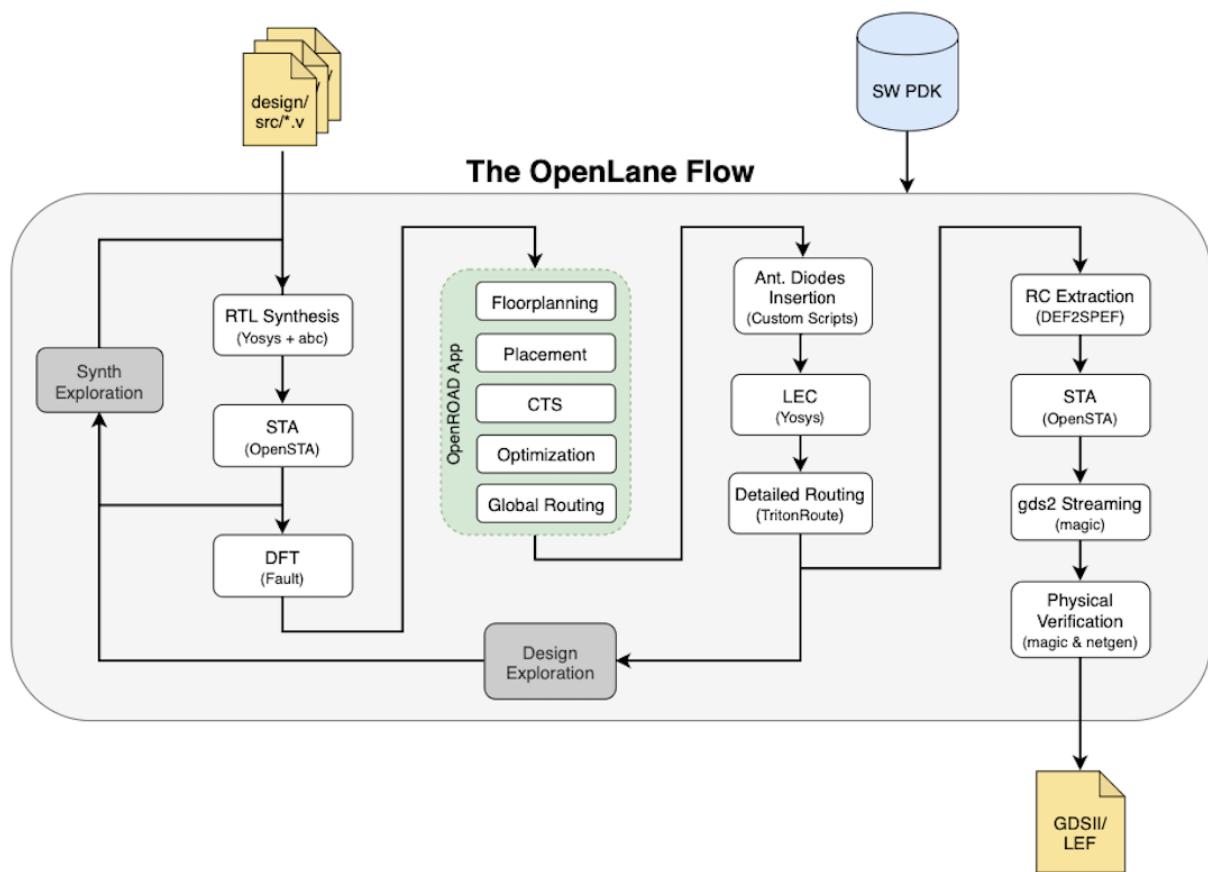
./flow.tcl -design serv_rf_top

This command runs the entire OpenLane flow from preparation of design to generating GDSII.

OpenLane Interactive flow



- Synthesis
- Floor/Power Planning
- Placement
- Clock Tree Synthesis
- Routing
- Sign Off



./flow.tcl -interactive

This command runs OpenLane in interactive flow

```
[root@d8161153276d openLANE_flow]# ./flow.tcl -interactive  
[INFO]:
```



```
[INFO]: Version: 2021.08.02_05.21.44-7-g71bf5bd  
[INFO]: Running interactively
```

Running the commands in interactive flow

package require openlane 0.9

This command checks the version of OpenLane required for the flow

```
% package require openlane 0.9  
0.9
```

prep -design serv_rf_top -tag serv_interactive

Preparing a design called serv_rf_top with tag for naming the current run.

```
M% prep -design serv_rf_top -tag serv
```

run_synthesis

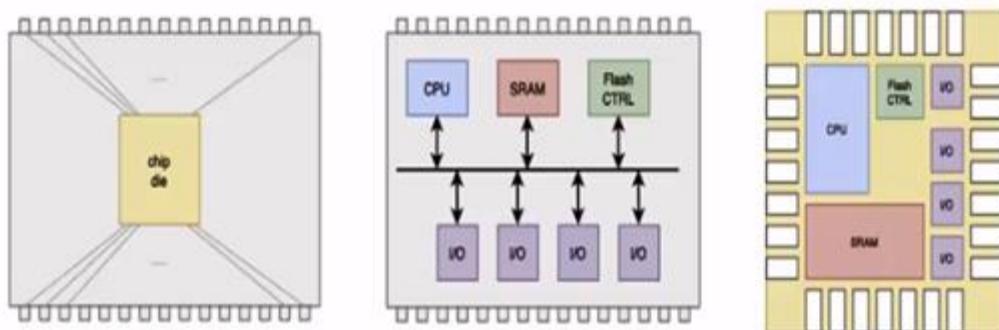
This command runs the synthesis right from Yosys till OpenSTA.

Conversion of RTL design to a circuit out of components from Standard cell library (SCL).

run_floorplan

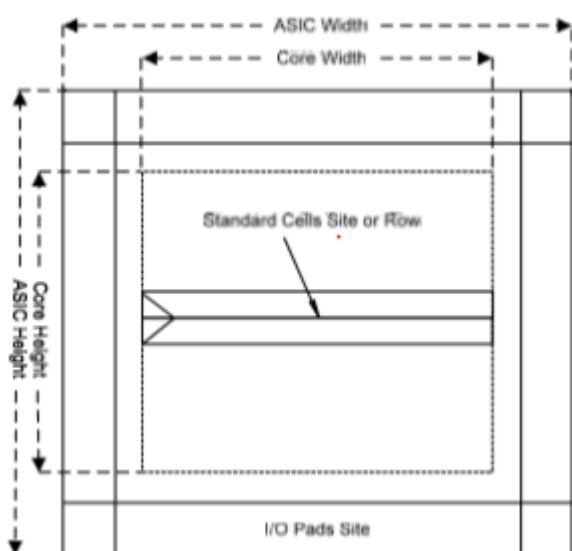
This command performs the Floor planning and Power Distribution network(PDN) of the design.

Chip Floorplanning - Partitioning the chip die between different system blocks and placing the I/O pads



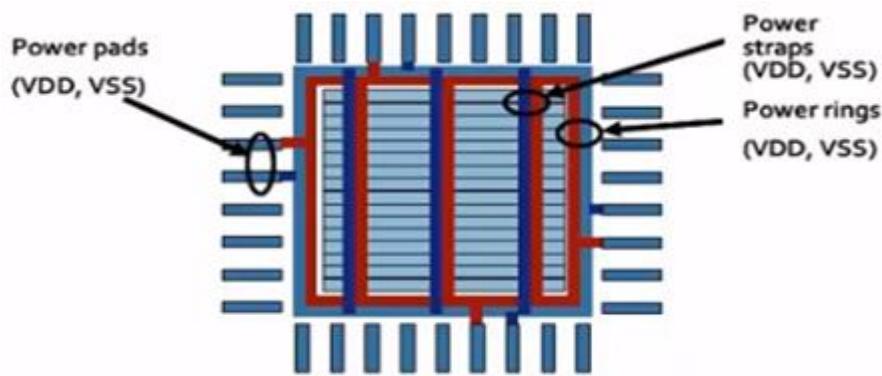
The above image describes the Chip Floorplanning

Macro Floorplanning - We define the macro dimensions, pin locations and row definitions.



Power Planning - Power network is constructed typically by multiple VDD and GND pins. the Power pins are all connected

through power rings, vertical and horizontal metal straps. Power rings- Power ring is designed around the core. Power rings contains both VDD and VSS rings. After the ring is placed a power is designed such that power reaches all the cells easily.

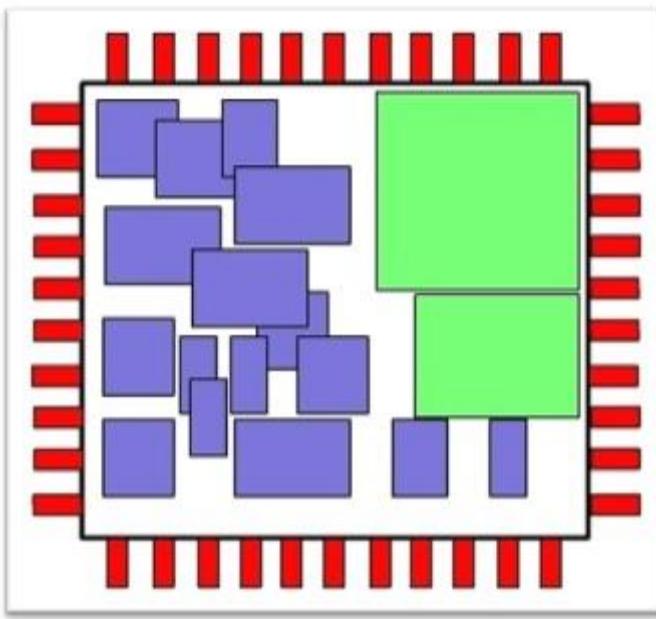


This diagram shows the power planning components. These parallel structures reduce resistance. To address electromagnetism the power distribution network uses upper metal layers as they are thicker than lower metal layers

run_placement

This command perform global placement and detailed placement of cells of the design.

Placing the cell on the floorplan rows aligned with sites. Connected cells should be placed close to each other to reduce the interconnect delay. It is done in two steps ie Global and Detailed Placement. In global placement , the approximate locations for cells is decided by placing cells in global bins. In detailed placement, cells are Placed without over lapping.

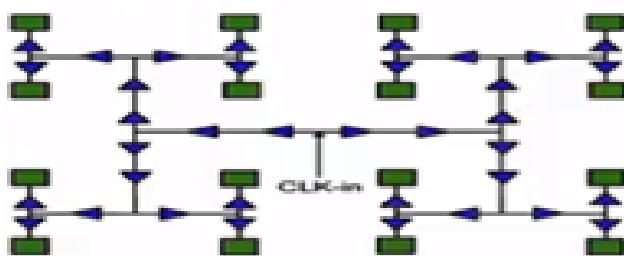


run_cts

Creating a clock distribution network :

➤ Delivering clock to all sequential elements.

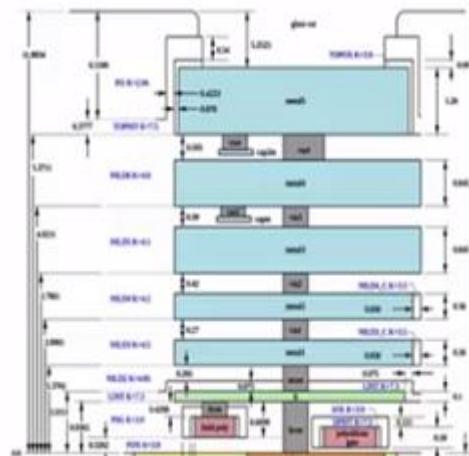
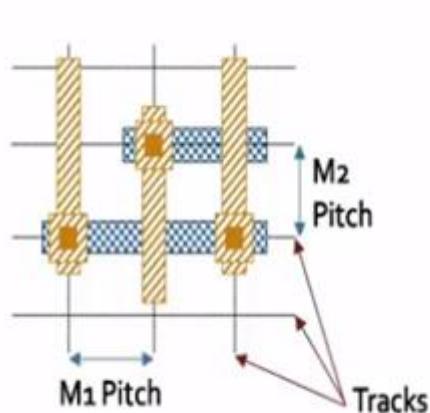
- With Minimum skew
- In Good shape
- A tree (H,X etc)



run_routing

Implements interconnects using available metal layers. Metal tracks form a routing grid. Routing uses Divide and Conquer method.

There are two types of Routing ie -Global routing: Generates routing guides -Detailed routing: Uses routing guides to implement actual wiring.



write_powered_verilog

```
set_netlist $::env(lvs_result_file.tag).powered.v
```

These commands write Verilog code from the netlist after we have set it in the file after routing.

run_magic

This command generates Layout

run_magic_drc

This command performs DRC check on the layout.

run_klayout

This command creates backup layout

scrot_klayout

This command is for generating layout in PNG format.

run_klayout_drc

Running DRC on layout generated from Klayout

run_klayout_gds_xor

This command is for generating GDS2 of our design with XOR design

run_lvs

This command checks for layout vs schematic violations

Layout Versus Schematic (LVS) checking compares the extracted netlist from the layout to the original schematic netlist to determine if they match. The comparison check is considered clean if all the devices and nets of the schematic match the devices and the nets of the layout.

run_lef_cvc

This command is used for Circuit Validity Checker on the output spice. Voltage aware ERC checker for CDL netlists.

run_antenna_check

This command is used for checking Antenna violations. Long metal lines and Vias introduce antenna violations. The antenna rule specifies the maximum tolerance for the ratio of a metal line area to the area of connected gates. VLSI process starts from the substrate, device layer and then metal layers.

Scripts for OpenLane flow

OpenLane flow scripts are mostly consisting of **Python**, **Shell scripting**, **TCL scripting** and some parts in **Perl** and **web design tools like CSS and JavaScript**. Some of the scripts are as shown below:

Report Generation Wrapper Python file

This report generator consists of all the wrappers to wrap around **report.py script** and **config.py script** and make it usable for **flow.tcl script**.

```

Open ▾ ⌂ report_generation_wrapper.py
~/Desktop/vsdlflow/work/tools/openlane_working_dir/openlane

1 # Copyright 2020 Efabless Corporation
2 #
3 # Licensed under the Apache License, Version 2.0 (the "License");
4 # you may not use this file except in compliance with the License.
5 # You may obtain a copy of the License at
6 #
7 #     http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14
15 # The only purpose of this file is to create a wrapper around report.py and config.py and make them usable by flow.tcl
16
17 import argparse
18 import os
19 import re
20 from scripts.report.report import Report
21 from scripts.config.config import ConfigHandler
22 import scripts.utils.utils as utils
23 from scripts.report.get_file_name import get_name
24
25 parser = argparse.ArgumentParser(
26     description='Creates a final summary csv report for a given design \
27     + a manufacturability report + a runtime summary report.')
28
29 parser.add_argument('--design', '-d', required=True,
30                     help='Design Path')
31
32 parser.add_argument('--design_name', '-dn', required=True,
33                     help='Design Name')
34
35 parser.add_argument('--tag', '-t', required=True,
36                     help='Run Tag')
37
38 parser.add_argument('--run_path', '-r', default=None,
39                     help='Run Path')
40
41 parser.add_argument('--output_file', '-o', required=True,
42                     help='Output Final Summary Report')
43
44 parser.add_argument('--man_report', '-m', required=True,
45                     help='Output Manufacturability Reports')
46
47 parser.add_argument('--runtime_summary', '-rs', required=True,
48                     help='Output Runtime Summary Reports')
49

50 args = parser.parse_args()
51 design = args.design
52 design_name = args.design_name
53 tag = args.tag
54 run_path=args.run_path
55 output_file = args.output_file
56 man_report = args.man_report
57 runtime_summary = args.runtime_summary
58
59 # Extracting Configurations
60 params = ConfigHandler.get_config(design, tag, run_path)
61 # Extracting Report
62 report = Report(design, tag, design_name,params,run_path).get_report()
63 # write into file
64 outputFileOpener = open(output_file,"w")
65 outputFileOpener.write(Report.get_header() + "," + ConfigHandler.get_header())
66 outputFileOpener.write("\n")
67 outputFileOpener.write(report)
68 outputFileOpener.close()
69
70 # Adding Extra Attributes computed from configs and reported statistics
71 utils.addComputedStatistics(output_file)
72
73 # Tracking Magic DRC, LVS, Antenna Logs:
74 if run_path is None:
75     run_path = utils.get_run_path(design, tag)
76 magic_drc_report=utils.get_name(str(run_path)+"/reports/magic/", "magic.drc")
77 lvs_report=utils.get_name(str(run_path)+"/results/lvs/" +design_name+".lvs_parsed.lef.log"
78 if not os.path.exists(lvs_report):
79     lvs_report=str(run_path)+"/results/lvs/" +design_name+".lvs_parsed.gds.log"
80 magic_antenna_report=utils.get_name(str(run_path)+"/reports/magic/", "magic.antenna_violators.rpt")
81 arc_antenna_report=utils.get_name(str(run_path)+"/reports/routing/", "antenna.rpt")
82
83 printArr = []
84
85 printArr.append("Design Name: " + design_name)
86 printArr.append("Run Directory: " + str(run_path))
87
88 splitLine = '-----'
89
90 # Summarizing Magic DRC
91 drcVioDict = dict()
92 cnt = 0
93 if os.path.exists(magic_drc_report):
94     drcFileOpener = open(magic_drc_report)
95     if drcFileOpener.mode == 'r':
96         drcContent = drcFileOpener.read()
97     drcFileOpener.close()

```

```

99 # design name
100 # violation message
101 # list of violations
102 # Total Count:
103 printArr.append(splitLine)
104 printArr.append("\nMagic DRC Summary:")
105 printArr.append("Source: " + str(magic_drc_report))
106 if drcContent is not None:
107     drcSections = drcContent.split(splitLine)
108     if (len(drcSections) > 2):
109         for i in range(1, len(drcSections) - 1, 2):
110             drcVioDict[drcSections[i]] = len(drcSections[i + 1].split("\n")) - 2
111             for key in drcVioDict:
112                 val = drcVioDict[key]
113                 cnt += val
114             printArr.append("Violation Message \'" + str(key.strip()) + "\' found " + str(val) + " Times.")
115     printArr.append("Total Magic DRC violations is " + str(cnt))
116 else:
117     printArr.append("Source not found.")
118 # Summarizing LVS
119 printArr.append(splitLine)
120 printArr.append("\nLVS Summary:")
121 printArr.append("Source: "+str(lvs_report))
122 if os.path.exists(lvs_report):
123     lvsFileOpener = open(lvs_report)
124     if lvsFileOpener.mode == 'r':
125         lvsContent = lvsFileOpener.read()
126     lvsFileOpener.close()
127     flag = False
128     for line in lvsContent.split("\n"):
129         if line.find("Total errors =") != -1:
130             flag = True
131             printArr.append(line)
132         elif line.find("net") != -1:
133             printArr.append(line)
134
135     if not flag:
136         printArr.append("Design is LVS clean.")
137 else:
138     printArr.append("Source not found.")
139 # Summarizing Antennas
140 printArr.append(splitLine)
141 printArr.append("\nAntenna Summary:")
142
143 if os.path.exists(arc_antenna_report):
144     printArr.append("Source: " + str(arc_antenna_report))
145     antFileOpener = open(arc_antenna_report)
146     if antFileOpener.mode == 'r':
147         antContent = antFileOpener.read().split("\n)[-5:]
148     antFileOpener.close()
149     for line in antContent:
150         if line.find("violated:") != -1:
151             printArr.append(line)
152 elif os.path.exists(magic_antenna_report):
153     printArr.append("Source: " + str(magic_antenna_report))
154     antFileOpener = open(magic_antenna_report)
155     if antFileOpener.mode == 'r':
156         antContent = antFileOpener.read().split("\n")
157     antFileOpener.close()
158     tot_cnt = 0
159     for ant in antContent:
160         if len(str(ant).strip()) > 0:
161             tot_cnt+=1
162     printArr.append(" Number of pins violated: " + str(tot_cnt))
163 else:
164     printArr.append("No antenna report found.")
165
166
167 # write into file
168 outputFileOpener = open(man_report,"w")
169 outputFileOpener.write("\n".join(printArr))
170 outputFileOpener.close()
171
172
173
174
175 def getListOffiles(dirName):
176     # create a list of file and sub directories
177     # We use os module's listdir function
178     allFiles = []
179     listOffile = os.listdir(dirName)
180     # Iterate over all the entries
181     for entry in listOffile:
182         # Create full path
183         fullPath = os.path.join(dirName, entry)
184         # If entry is a directory then get the list of files in this directory
185         if os.path.isdir(fullPath):
186             allFiles = allFiles + getListOffiles(fullPath)
187         else:
188             allFiles.append(fullPath)
189     return allFiles
190
191 def convert_to_seconds(runtime):
192     pattern = re.compile('(\d{1,2}:\d{1,2}:\d{1,2})')
193     m = pattern.match(runtime)
194     time = int(m.group(1))*60*60 + int(m.group(2))*60 + int(m.group(3)) + int(m.group(4))/1000.0
195     return str(time)
196
197 # Creating a runtime summary report
198 logs_path = "/local"
199 neededfiles = sorted([(int(os.path.basename(f).split("-")[1][0]),f) for f in getListOffiles(logs_path) if os.path.isfile(os.path.join(logs_path, f)) and len(f.split('_')) > 1 and f.split('_')[1] == "runtime.txt" and len(os.path.basename(f).split('_')) > 1 and os.path.basename(f).split('_')[0].isnumeric()])
200 runtimeArr = []
201

```

```

200 runtimeArr = []
201 prasableRuntimeArr= []
202 for (idx,f) in neededfiles:
203     stagename = os.path.basename(f).split("_runtime.txt")[0]
204     runtimeFileOpener = open(f, "r")
205     if runtimeFileOpener.mode == 'r':
206         runtimeContent = runtimeFileOpener.read().strip()
207     runtimeFileOpener.close()
208     runtimeArr.append(str(stagename)+ " "+ str(runtimeContent))
209     prasableRuntimeArr.append(str(stagename)+ " "+ conver_to_seconds(runtimeContent))
210
211 # write into file
212 outputFileOpener = open(runtime_summary,"w")
213 outputFileOpener.write("\n".join(runtimeArr))
214 outputFileOpener.close()
215
216 outputFileOpener = open(runtime_summary+".parsable","w")
217 outputFileOpener.write("\n".join(prasableRuntimeArr))
218 outputFileOpener.close()

```

synthesis.tcl (Configuration of design for Synthesis)

This configuration .tcl file consists of **Yosys and all other synthesis configuration settings** to be set for proper functioning of synthesis.

```

# Copyright 2020 Efabless Corporation
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Synth defaults
set ::env(SYNTH_BIN) yosys
set ::env(SYNTH_SCRIPT) ${::env(SCRIPTS_DIR)}/synth.tcl
set ::env(SYNTH_NO_FLAT) 0
set ::env(SYNTH_SHARE_RESOURCES) 1
set ::env(SYNTH_BUFFERING) 1
set ::env(SYNTH_SIZING) 0
set ::env(SYNTH_MAX_FANOUT) 5
set ::env(SYNTH_STRATEGY) "AREA_0"
set ::env(SYNTH_ADDER_TYPE) "YOSYS"
set ::env(CLOCK_BUFFER_FANOUT) 16
set ::env(SYNTH_READ_BLACKBOX_LIB) 0
set ::env(SYNTH_TOP_LEVEL) 0
set ::env(SYNTH_FLAT_TOP) 0
set ::env(IO_PCT) 0.2

set ::env(BASE_SDC_FILE) ${::env(OPENLANE_ROOT)}/scripts/base.sdc
~  
~  
~  
~  
~

```

general.tcl (General configurations of a design)

This configuration .tcl file consists of all the general configuration settings to be set for proper functioning of **Openlane** flow.

```
# default pdk
set ::env(PDK) "sky130A"
set ::env(STD_CELL_LIBRARY) "sky130_fd_sc_hd"
set ::env(USE_GPIO_PADS) 0

# Flow control defaults
set ::env(LEC_ENABLE) 0
set ::env(YOSYS_REWRITE_VERILOG) 0

set ::env(RUN_MAGIC) 1
set ::env(MAGIC_PAD) 0
set ::env(MAGIC_ZEROIZE_ORIGIN) 0
set ::env(MAGIC_GENERATE_GDS) 1
set ::env(MAGIC_GENERATE_LEF) 1
set ::env(MAGIC_GENERATE_MAGLEV) 1
set ::env(MAGIC_WRITE_FULL_LEF) 1
set ::env(MAGIC_DRC_USE_GDS) 0
set ::env(MAGIC_EXT_USE_GDS) 1
set ::env(MAGIC_INCLUDE_GDS_POINTERS) 0
set ::env(MAGIC_DISABLE_HIER_GDS) 1
set ::env(MAGIC_CONVERT_DRC_TO_RDB) 1

set ::env(KLAYOUT_XOR_GDS) 1
set ::env(KLAYOUT_XOR_XML) 1

set ::env(RUN_ROUTING_DETAILED) 1
set ::env(RUN_SIMPLE_CTS) 0
set ::env(CLOCK_PERIOD) "10ns"
set ::env(RUN_KLAYOUT) 1
set ::env(TAKE_LAYOUT_SCROT) 1
set ::env(RUN_KLAYOUT_DRC) 0
set ::env(KLAYOUT_DRC_KLAYOUT_GDS) 0
set ::env(RUN_KLAYOUT_XOR) 1
set ::env(USE_ARC_ANTENNA_CHECK) 1

set ::env(FILL_INSERTION) 1
set ::env(TAP_DECAP_INSERTION) 1

set ::env(WIDEN_SITE) 1
set ::env(WIDEN_SITE_IS_FACTOR) 1

set ::env(RUN_SPEF_EXTRACTION) 1
set ::env(SPEF_WIRE_MODEL) "L"
set ::env(SPEF_EDGE_CAP_FACTOR) 1

set ::env(RUN_CVC) 1

set ::env(WIRE_RC_LAYER) "met1"; # Used for estimate_parasitics
set ::env(GENERATE_FINAL_SUMMARY_REPORT) 1
```

```
set ::env(WIRE_RC_LAYER) "met1"; # Used for estimate_parasitics
set ::env(GENERATE_FINAL_SUMMARY_REPORT) 1
# 0: no diodes
# 1: spray inputs with diodes
# 2: spray inputs with fake diodes first then fix up the violators with real ones
# 3: use FR Antenna Avoidance flow
# 4: Spray diodes on design pins, and add diodes where they need to be added for each macro.
# 5: Same as 2 but behaves like 4.
set ::env(DIODE_INSERTION_STRATEGY) 3

# psn
if { [file exists /build/transforms/] } {
    set ::env(PSN_TRANSFORM_PATH) /build/transforms
} else {
    set ::env(PSN_TRANSFORM_PATH) ${::env(HOME)}/.local/transforms
}
```

Analyse.pl (Synthesis Analysis Perl file)

This Perl file is used as analysing file by declaring parameters like **min area**, **min delay**, **best area** and **best delay** etc.



```
#!/usr/bin/perl
# Copyright 2020 Mohamed Shalan
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
open (myfile, $ARGV[0]) || die "couldn't open the file!";
$aw = 0.5;
$dw = 0.5;
$minArea = 1000000000;
$minGates = $minArea;
$minDelay = $minArea;
$bestArea = "";
$bestDelay = "";
$sn = "";
while (<myfile>){
    #print chomp($line);
    if (/none/){
        m/Delay\s+\=\s+(\S+)/;
        my $delay = $1;
        m/Area\s+\=\s+(\S+)/;
        my $area = $1;
        my @data = split;
        #print "$_";
        my $factor = $aw*$area/$ascale + $dw*$delay;
        #print "$data[6]\t$area\t$delay\n";
        if ($area < $minArea) {
            $minArea = $area;
            $bestArea = $sn;
        }
        if ($delay < $minDelay) {
            $minDelay = $delay;
            $bestDelay = $sn;
        }
        if ($data[6] < $minGates) {
            $minGates = $data[6];
            $bestGates = $sn;
        }
    }
}
```

```
118
119 my $cntr =0;
120 while (<myfile>) {
121     #print chop($line);
122     if(/(none)/){
123
124         $cntr++;
125         print "<tr>\n";
126
127         m/Delay\$\+=\$s+(\$+)/;
128         my $delay = $1;
129         m/Area\$\+=\$+(\$+)/;
130         my $area = $1;
131
132         my @data = split;
133         #print "$_ ";
134
135         my $dfactor = 0.25*$area/$minArea + 0.75*$delay/$minDelay;
136         my $afactor = 0.75*$area/$minArea + 0.25*$delay/$minDelay;
137
138         $dratio = int($delay/$minDelay*1000)/1000;
139         $ratio = int($area/$minArea*1000)/1000;
140         $gratio = int(@data[6]/$minGates*1000)/1000;
141
142         #printf ("%0.0f%0.0f%0.0f%0.0f%0.0f%0.0f",@data[6],$area,$delay,$data[6]/$minGates,$area/$minArea,$delay/$minDelay);
143         print "<td>$cntr: $sn</td>\n";
144
145         if($gratio < 1.0001) {
146             print "<td bgcolor="#ccff99">$data[6]</td>\n";
147         } else {
148             print "<td>$data[6]</td>\n";
149         }
150
151         if($ratio < 1.0001) {
152             print "<td bgcolor="#ccff99">$area</td>\n";
153         } else {
154             print "<td>$area</td>\n";
155         }
156
157         if($dratio < 1.0001) {
158             print "<td bgcolor="#ccff99">$delay</td>\n";
159         } else {
160             print "<td>$delay</td>\n";
161         }
162
163     }
```

sta.tcl:

This is the main tcl file where we set **timing parameters**,
report total negative slack, worst negative slack,
checks and power parameters.

```

# Copyright 2020 Efabless Corporation
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

set_cmd_units -time ns -capacitance pF -current mA -voltage V -resistance kOhm -distance um

read_liberty -min ${::env(LIB_FASTEST)}
read_liberty -max ${::env(LIB_SLOWEST)}
read_verilog ${::env(CURRENT_NELIST)}
link_design ${::env(DESIGN_NAME)}
if { [info exists ::env(CURRENT_SPEF)] } {
    read_spef ${::env(CURRENT_SPEF)}
}

#set_units -capacitance fF
read_sdc -echo ${::env(BASE_SDC_FILE)}
#report_checks
report_tns
report_tns > ${::env(opensta_report_file_tag)}_tns.rpt
report_wns
report_wns > ${::env(opensta_report_file_tag)}_wns.rpt
report_power
report_power > ${::env(opensta_report_file_tag)}_power.rpt

report_checks -fields {capacitance slew input_pins nets fanout} -unique -slack_max -0.0 -group_count 100 > ${::env(opensta_report_file_tag)}_slew.rpt
report_checks -fields {capacitance slew input_pins nets fanout} -path_delay_min_max > ${::env(opensta_report_file_tag)}.min_max.rpt
report_checks -fields {capacitance slew input_pins nets fanout} -group_count 100 -slack_max -0.01 > ${::env(opensta_report_file_tag)}_slack.rpt
report_check_types -max_slew -max_capacitance -max_fanout -violators > ${::env(opensta_report_file_tag)}.slew.rpt
exit

```

config.py

This python extension config has all the configurations required for smooth running of OpenLane.

```
14
15 import os
16 import subprocess
17 import sys
18 from shutil import copyfile
19 from ..utils.utils import *
20 from collections import OrderedDict
21
22
23
24 class ConfigHandler():
25     config_getter_script = os.path.join(os.path.dirname(__file__), "config_get.sh")
26     configuration_values = ['CLOCK_PERIOD', 'SYNTH_STRATEGY', 'SYNTH_MAX_FANOUT', 'FP_CORE_UTIL', 'FP_ASPECT_RATIO',
27                             'FP_PDN_VPITCH', 'FP_PDN_HPITCH', 'PL_TARGET_DENSITY', 'GLB_RT_ADJUSTMENT', 'STD_CELL_LIBRARY', 'CELL_PAD', 'DIODE_IN'
28
29     base_config_values = ['DESIGN_NAME', 'VERILOG_FILES', 'CLOCK_PERIOD', 'CLOCK_PORT']
30     @classmethod
31     def update_configuration_values(cls, params, append):
32         if append:
33             cls.configuration_values = cls.configuration_values + params
34         else:
35             cls.configuration_values = params
36             cls.configuration_values = list(OrderedDict.fromkeys(cls.configuration_values))
37
38     @classmethod
39     def update_configuration_values_to_all(cls, append):
40         configFiles = ["synthesis.tcl", "floorplan.tcl", "placement.tcl", "cts.tcl", "routing.tcl"]#, "magic.tcl"]
41         config_relative_path = "configuration/"
42         config_path = os.path.join(os.getcwd(), config_relative_path)
43         if append == False:
44             cls.configuration_values = []
45
46         for configFile in configFiles:
47             tmpFile = open(config_path+configFile,"r")
48             if tmpFile.mode == 'r':
49                 configurationFileContent = tmpFile.read().split("\n")
50                 for line in configurationFileContent:
51                     start = line.find("(")
52                     end = line.find(")")
53                     if (start > -1) & (end > 0) & (line.find("SCRIPT") == -1) :
54                         cls.configuration_values.append(line[start+1:end])
55
56             except OSError:
57                 print ("Could not open/read file:", config_path)
58                 sys.exit()
59             cls.configuration_values = list( dict.fromkeys(cls.configuration_values) )
60
61
```

flow.tcl

This tcl file is the main script upon which OpenLane flow runs based on user's preference ie automatic flow or interactive flow.

```

set ::env(OPENLANE_ROOT) [file dirname [file normalize [info script]]]

lappend ::auto_path "$::env(OPENLANE_ROOT)/scripts/"
package require openlane; # provides the utils as well

proc run_non_interactive_mode {args} {
    set options {
        {-design required}
        {-save_path optional}
        {-no_lvs optional}
        {-no_drc optional}
        {-no_antennacheck optional}
    }
    set flags {-save}
    parse_key_args "run_non_interactive_mode" args arg_values $options flags_map $flags -no_consumers
    prep {*}$args

    run_synthesis
    run_floorplan
    run_placement
    run_cts
    run_resizer_timing
    run_routing

    if { ($::env(DIODE_INSERTION_STRATEGY) == 2) || ($::env(DIODE_INSERTION_STRATEGY) == 5) } {
        run_antenna_check
        heal_antenna_violators; # modifies the routed DEF
    }

    if { $::env(LVS_INSERT_POWER_PINS) } {
        write_powered_verilog
        set_netlist $::env(lvs_result_file_tag).powered.v
    }

    run_magic
    run_klayout
    run_klayout_gds_xor

    if { ! [info exists flags_map(-no_lvs)] } {
        run_magic_spice_export
    }

    if { [info exists flags_map(-save) ] } {
        if { ! [info exists arg_values(-save_path)] } {
            set arg_values(-save_path) ""
        }
    }
}

```

```

Rhythmbox
run_magic

run_klayout
run_klayout_gds_xor

if { ! [info exists flags_map(-no_lvs)] } {
    run_magic_spice_export
}

if { [info exists flags_map(-save) ] } {
    if { ! [info exists arg_values(-save_path)] } {
        set arg_values(-save_path) ""
    }
    save_views -lef_path ${:env(magic_result_file_tag).lef \
                  -def_path ${:env(tritonRoute_result_file_tag).def \
                  -gds_path ${:env(magic_result_file_tag).gds \
                  -mag_path ${:env(magic_result_file_tag).mag \
                  -maglef_path ${:env(magic_result_file_tag).lef.mag \
                  -spice_path ${:env(magic_result_file_tag).spice \
                  -verilog_path ${:env(CURRENT_NETLIST) \
                  -save_path $arg_values(-save_path) \
                  -tag ${:env(RUN_TAG)
}

# Physical verification
if { ! [info exists flags_map(-no_lvs)] } {
    run_lvs; # requires run_magic_spice_export
}

if { ! [info exists flags_map(-no_drc)] } {
    run_magic_drc
    run_klayout_drc
}

if { ! [info exists flags_map(-no_antennacheck) ] } {
    run_antenna_check
}

run_lef_cvc
calc_total_runtime
generate_final_summary_report
puts_success "Flow Completed Without Fatal Errors."
}

proc run_interactive_mode {args} {
    set options {
        {-design optional}
    }
}

```

```

proc run_interactive_mode {args} {
    set options {
        {-design optional}
    }
    set flags {}
    parse_key_args "run_interactive_mode" args arg_values $options flags_map $flags -no_consume
    if { [info exists arg_values(-design)] } {
        prep {*}$args
    }

    set ::env(TCLLIBPATH) ::auto_path
    exec tcsh >@stdout
}

proc run_magic_drc_batch {args} {
    set options {
        {-magicrc optional}
        {-tech optional}
        {-report required}
        {-design required}
        {-gds required}
    }
    set flags {}
    parse_key_args "run_magic_drc_batch" args arg_values $options flags_map $flags
    if { [info exists arg_values(-magicrc)] } {
        set magicrc [file normalize $arg_values(-magicrc)]
    }
    if { [info exists arg_values(-tech)] } {
        set ::env(TECH) [file normalize $arg_values(-tech)]
    }
    set ::env(GDS_INPUT) [file normalize $arg_values(-gds)]
    set ::env(REPORT_OUTPUT) [file normalize $arg_values(-report)]
    set ::env(DESIGN_NAME) $arg_values(-design)

    if { [info exists magicrc] } {
        exec magic \
            -noconsole \
            -dnull \
            -rcfile $magicrc \
            ::env(OPENLANE_ROOT)/scripts/magic/drc_batch.tcl \
            </dev/null |& tee /dev/tty
    } else {
        exec magic \
            -noconsole \
            -dnull \
            ::env(OPENLANE_ROOT)/scripts/magic/drc_batch.tcl \
            </dev/null |& /dev/tty
    }
}

```

```

proc run_lvs_batch {args} {
    # runs device level lvs on -gds/CURRENT_GDS and -net/CURRENT_NETLIST
    # extracts gds only if EXT_NETLIST does not exist
    set options {
        {-design required}
        {-gds optional}
        {-net optional}
    }
    set flags {}
    parse_key_args "run_lvs_batch" args arg_values $options flags_lvs $flags -no_consume
    prep {*}$args

    if { [info exists arg_values(-gds)] } {
        set ::env(CURRENT_GDS) [file normalize $arg_values(-gds)]
    } else {
        set ::env(CURRENT_GDS) ${::env(RESULTS_DIR)}/magic/${::env(DESIGN_NAME)}.gds
    }
    if { [info exists arg_values(-net)] } {
        set ::env(CURRENT_NETLIST) [file normalize $arg_values(-net)]
    }
    if { ! [file exists ${::env(CURRENT_GDS)}] } {
        puts_err "Could not find GDS file \\"${::env(CURRENT_GDS)}\""
        exit 1
    }
    if { ! [file exists ${::env(CURRENT_NETLIST)}] } {
        puts_err "Could not find NET file \\"${::env(CURRENT_NETLIST)}\""
        exit 1
    }

    set ::env(MAGIC_EXT_USE_GDS) 1
    set ::env(EXT_NETLIST) ${::env(RESULTS_DIR)}/magic/${::env(DESIGN_NAME)}.gds.spice
    if { [file exists ${::env(EXT_NETLIST)}] } {
        puts_warn "Reusing ${::env(EXT_NETLIST)}. Delete to remake."
    } else {
        run_magic_spice_export
    }

    run_lvs; # requires run_magic_spice_export
}

proc run_file {args} {
    set ::env(TCLLIBPATH) ${::auto_path}
    exec tclsh {*}$args >&stdout
}

set options {
    {-file optional}

```

Generation of SERV Utilization and Area Report

This report generates SERV utilization statistics ie **number of basic gates, wires, memories, cells and processes and flip flops** needed for SERV design. This Report was generated for default clock period of 10ns and Frequency of design as 100MHz. This was based on

Maximum Operating Frequency Calculations section.

Synthesis

- Using Automated Flow of OpenLane/ Interactive Flow of OpenLane

```
1
2 40. Printing statistics.
3
4 === serv_rf_top ===
5
6   Number of wires:          4624
7   Number of wire bits:      4782
8   Number of public wires:   1264
9   Number of public wire bits: 1422
10  Number of memories:       0
11  Number of memory bits:    0
12  Number of processes:      0
13  Number of cells:          4713
14    sky130_fd_sc_hd_a2111o_2      1
15    sky130_fd_sc_hd_a211o_2       1
16    sky130_fd_sc_hd_a211oi_2      2
17    sky130_fd_sc_hd_a21bo_2       2
18    sky130_fd_sc_hd_a21o_2        2
19    sky130_fd_sc_hd_a21oi_2       4
20    sky130_fd_sc_hd_a221o_2       3
21    sky130_fd_sc_hd_a221oi_2      1
22    sky130_fd_sc_hd_a22o_2        46
23    sky130_fd_sc_hd_a2bb2o_2      4
24    sky130_fd_sc_hd_a311o_2       1
25    sky130_fd_sc_hd_a31o_2        2
26    sky130_fd_sc_hd_a31oi_2       1
27    sky130_fd_sc_hd_a32o_2        5
28    sky130_fd_sc_hd_a41o_2       1
29    sky130_fd_sc_hd_and2_2        9
30    sky130_fd_sc_hd_and2b_2      1
31    sky130_fd_sc_hd_and3_2        4
32    sky130_fd_sc_hd_and4_2       3
33    sky130_fd_sc_hd_buf_1         646
34    sky130_fd_sc_hd_conb_1        2
35    sky130_fd_sc_hd_dfxtp_2       1335
36    sky130_fd_sc_hd_dfxtp_2      22
```

```

28    sky130_fd_sc_hd_a41o_2      1
29    sky130_fd_sc_hd_and2_2      9
30    sky130_fd_sc_hd_and2b_2     1
31    sky130_fd_sc_hd_and3_2      4
32    sky130_fd_sc_hd_and4_2      3
33    sky130_fd_sc_hd_buf_1      646
34    sky130_fd_sc_hd_conb_1      2
35    sky130_fd_sc_hd_dfxtp_2     1335
36    sky130_fd_sc_hd_inv_2       98
37    sky130_fd_sc_hd_mux2_1      106
38    sky130_fd_sc_hd_mux2_2     1171
39    sky130_fd_sc_hd_mux4_1      382
40    sky130_fd_sc_hd_nand2_2     8
41    sky130_fd_sc_hd_nand2b_2    4
42    sky130_fd_sc_hd_nor2_2      29
43    sky130_fd_sc_hd_o2111ai_2    1
44    sky130_fd_sc_hd_o211a_2      5
45    sky130_fd_sc_hd_o21a_2       9
46    sky130_fd_sc_hd_o21ai_2      18
47    sky130_fd_sc_hd_o21ba_2      2
48    sky130_fd_sc_hd_o221a_2      38
49    sky130_fd_sc_hd_o221ai_2     3
50    sky130_fd_sc_hd_o22a_2       58
51    sky130_fd_sc_hd_o2bb2a_2     3
52    sky130_fd_sc_hd_o311a_2      5
53    sky130_fd_sc_hd_o31a_2       1
54    sky130_fd_sc_hd_o32a_2       5
55    sky130_fd_sc_hd_or2_2       655
56    sky130_fd_sc_hd_or3_2       25
57    sky130_fd_sc_hd_or3b_2       4
58    sky130_fd_sc_hd_or4_2       5
59    sky130_fd_sc_hd_or4b_2       2
60
61    Chip area for module '\serv_rf_top': 62715.148800
62

```

This report generates SERV utilization statistics ie
number of basic gates, wires, memories, cells and processes and flip flops needed for SERV design. This Report was generated for clock period of 20ns and Frequency of design as 50MHz. This was based on

Maximum Operating Frequency Calculations section.

Floorplanning

The below report generated by OPENROAD tool.

```
[INFO ODB-0223]      Created 13 technology layers
[INFO ODB-0224]      Created 25 technology vias
[INFO ODB-0225]      Created 441 library cells
[INFO ODB-0226] Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-merged_unpadded.lef
[INFO IFP-0001] Added 139 rows of 823 sites.
[INFO] Extracting DIE_AREA and CORE_AREA from the floorplan
[INFO] Floorplanned on a die area of 0.0 0.0 389.91 400.63 (microns). Saving to /openLANE_flow/-designs/serv_rf_top/runs/11-08_11-58/reports/floorplan/3-verilog2def.die_area.rpt.
[INFO] Floorplanned on a core area of 5.52 10.88 384.1 388.96 (microns). Saving to /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/reports/floorplan/3-verilog2def.core_area.rpt.
```

The below report generated by ioPlacer Tool

```
[INFO ODB-0223]      Created 13 technology layers
[INFO ODB-0224]      Created 25 technology vias
[INFO ODB-0225]      Created 441 library cells
[INFO ODB-0226] Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-merged.lef
[INFO ODB-0127] Reading DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-floorplan/3-verilog2def_openroad.def
[INFO ODB-0128] Design: serv_rf_top
[INFO ODB-0130]      Created 172 pins.
[INFO ODB-0131]      Created 6209 components and 46457 component-terminals.
[INFO ODB-0133]      Created 6278 nets and 21619 connections.
[INFO ODB-0134] Finished DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-floorplan/3-verilog2def_openroad.def
Found 0 macro blocks.
Using 1u default distance from corners.
Using 2 tracks default min distance between IO pins.
[INFO PPL-0007] Random pin placement.
```

The below report was generated by tapcell tool.

```
[INFO ODB-0223]      Created 13 technology layers
[INFO ODB-0224]      Created 25 technology vias
[INFO ODB-0225]      Created 441 library cells
[INFO ODB-0226] Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/merged_unpadded.lef
[INFO ODB-0127] Reading DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/floorplan/4-ioPlacer.def
[INFO ODB-0128] Design: serv_rf_top
[INFO ODB-0130]      Created 172 pins.
[INFO ODB-0131]      Created 6209 components and 46457 component-terminals.
[INFO ODB-0133]      Created 6278 nets and 21619 connections.
[INFO ODB-0134] Finished DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/floorplan/4-ioPlacer.def
[WARNING TAP-0014] endcap_cpp option is deprecated.
[INFO TAP-0001] Macro blocks found: 0
[INFO TAP-0002] Original rows: 139
[INFO TAP-0003] Created rows: 0
[INFO TAP-0004] Endcaps inserted: 278
[INFO TAP-0005] Tapcells inserted: 2044
```

The below report was generated by pdn tool.

```
[INFO ODB-0223]      Created 13 technology layers
[INFO ODB-0224]      Created 25 technology vias
[INFO ODB-0225]      Created 441 library cells
[INFO ODB-0226] Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/merged_unpadded.lef
[INFO ODB-0127] Reading DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/results/floorplan/serv_rf_top.floorplan.def
[INFO ODB-0128] Design: serv_rf_top
[INFO ODB-0130]      Created 172 pins.
[INFO ODB-0131]      Created 8531 components and 51657 component-terminals.
[INFO ODB-0133]      Created 6278 nets and 21619 connections.
[INFO ODB-0134] Finished DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/results/floorplan/serv_rf_top.floorplan.def
[INFO PDN-0016] Power Delivery Network Generator: Generating PDN
    config: /home/badboy07/Desktop/OpenLane/pdks/sky130A/libs.tech/openlane/common_pdn.tcl
[INFO PDN-0008] Design Name is serv_rf_top
[INFO PDN-0009] Reading technology data
[INFO PDN-0011] ***** INFO *****
Type: stdcell, grid
    Stdcell Rails
        Layer: met1 - width: 0.480 pitch: 2.720 offset: 0.000
    Straps
        Layer: met4 - width: 1.600 pitch: 153.600 offset: 16.320
        Layer: met5 - width: 1.600 pitch: 153.180 offset: 16.650
    Connect: {met4 met5} {met1 met4}
Type: macro, macro_1
    Macro orientation: R0 R180 MX MY R90 R270 MXR90 MYR90
    Straps
    Connect: {met4_PIN_ver met5}
[INFO PDN-0012] **** END INFO ****
[INFO PDN-0013] Inserting stdcell grid - grid
[INFO PDN-0015] Writing to database
[WARNING PSM-0016] Voltage pad location (vsrc) file not specified, defaulting pad location to checkerboard pattern on core area.
[WARNING PSM-0017] X direction bump pitch is not specified, defaulting to 140um.
[WARNING PSM-0018] Y direction bump pitch is not specified, defaulting to 140um.
[WARNING PSM-0019] Voltage on net VDDP is not explicitly set
```

Placement

This report is generated by Replace tool which performs global placement.

```
[INFO ODB-0223]      Created 13 technology layers
[INFO ODB-0224]      Created 25 technology vias
[INFO ODB-0225]      Created 441 library cells
[INFO ODB-0226]      Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/merged_unpadded.lef
[INFO ODB-0127]      Reading DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/floorplan/6-pdn.def
[INFO ODB-0128]      Design: serv_rf_top
[INFO ODB-0130]      Created 174 pins.
[INFO ODB-0131]      Created 8531 components and 51657 component-terminals.
[INFO ODB-0132]      Created 2 special nets and 30036 connections.
[INFO ODB-0133]      Created 6278 nets and 21619 connections.
[INFO ODB-0134]      Finished DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/floorplan/6-pdn.def
[INFO GPL-0002]      DBU: 1000
[INFO GPL-0003]      SiteSize: 460 2720
[INFO GPL-0004]      CoreAreaLxLy: 5520 10880
[INFO GPL-0005]      CoreAreaUxLy: 384100 388960
[INFO GPL-0006]      NumInstances: 8531
[INFO GPL-0007]      NumPlaceInstances: 6209
[INFO GPL-0008]      NumFixedInstances: 2322
[INFO GPL-0009]      NumDummyInstances: 0
[INFO GPL-0010]      NumNets: 6278
[INFO GPL-0011]      NumPins: 21791
[INFO GPL-0012]      DieAreaLxLy: 0 0
[INFO GPL-0013]      DieAreaUxLy: 389910 400630
[INFO GPL-0014]      CoreAreaLxLy: 5520 10880
[INFO GPL-0015]      CoreAreaUxLy: 384100 388960
[INFO GPL-0016]      CoreArea: 143133526400
[INFO GPL-0017]      NonPlaceInstsArea: 3600953600
[INFO GPL-0018]      PlaceInstsArea: 71771334400
[INFO GPL-0019]      Util(%): 51.44
[INFO GPL-0020]      StdInstsArea: 71771334400
[INFO GPL-0021]      MacroInstsArea: 0
[InitialPlace] Iter: 1 CG Error: 0.00038485 HPWL: 121022940
[InitialPlace] Iter: 2 CG Error: 0.00002744 HPWL: 111504426
[InitialPlace] Iter: 3 CG Error: 0.00003462 HPWL: 112694834
[InitialPlace] Iter: 4 CG Error: 0.00001342 HPWL: 113210073
[InitialPlace] Iter: 5 CG Error: 0.00000860 HPWL: 113305296
[INFO GPL-0031] FillerInit: NumGCells: 6654
[INFO GPL-0032] FillerInit: NumGNets: 6278
[INFO GPL-0033] FillerInit: NumGPins: 21791
[TINFO GPL-0023] TargetDensity: 0.55
```

```
48 [INFO GPL-0023] TargetDensity: 0.55
49 [INFO GPL-0024] AveragePlaceInstArea: 11559242
50 [INFO GPL-0025] IdealBinArea: 21016804
51 [INFO GPL-0026] IdealBinCnt: 6810
52 [INFO GPL-0027] TotalBinArea: 143133526400
53 [INFO GPL-0028] BinCnt: 64 64
54 [INFO GPL-0029] BinSize: 5916 5908
55 [INFO GPL-0030] NumBins: 4096
56 [NesterovSolve] Iter: 1 overflow: 0.955274 HPWL: 72276022
57 [NesterovSolve] Iter: 10 overflow: 0.904408 HPWL: 80853630
58 [NesterovSolve] Iter: 20 overflow: 0.902267 HPWL: 81042851
59 [NesterovSolve] Iter: 30 overflow: 0.901961 HPWL: 81094651
60 [NesterovSolve] Iter: 40 overflow: 0.901775 HPWL: 81230585
61 [NesterovSolve] Iter: 50 overflow: 0.901759 HPWL: 81158562
62 [NesterovSolve] Iter: 60 overflow: 0.902377 HPWL: 81114106
63 [NesterovSolve] Iter: 70 overflow: 0.902366 HPWL: 81077377
64 [NesterovSolve] Iter: 80 overflow: 0.902233 HPWL: 81074848
65 [NesterovSolve] Iter: 90 overflow: 0.902421 HPWL: 81085186
66 [NesterovSolve] Iter: 100 overflow: 0.902609 HPWL: 81104394
67 [NesterovSolve] Iter: 110 overflow: 0.902685 HPWL: 81168196
68 [NesterovSolve] Iter: 120 overflow: 0.902903 HPWL: 81341531
69 [NesterovSolve] Iter: 130 overflow: 0.902453 HPWL: 81718568
70 [NesterovSolve] Iter: 140 overflow: 0.902036 HPWL: 82575234
71 [NesterovSolve] Iter: 150 overflow: 0.901186 HPWL: 84642791
72 [NesterovSolve] Iter: 160 overflow: 0.898471 HPWL: 89749234
73 [NesterovSolve] Iter: 170 overflow: 0.894573 HPWL: 95574589
74 [NesterovSolve] Iter: 180 overflow: 0.891371 HPWL: 99646893
75 [NesterovSolve] Iter: 190 overflow: 0.885019 HPWL: 104171396
76 [NesterovSolve] Iter: 200 overflow: 0.873921 HPWL: 111893077
77 [NesterovSolve] Iter: 210 overflow: 0.857049 HPWL: 121263293
78 [NesterovSolve] Iter: 220 overflow: 0.833948 HPWL: 130038575
79 [NesterovSolve] Iter: 230 overflow: 0.806923 HPWL: 139170070
80 [INFO GPL-0100] worst slack 1e+30
```

```
74 [NesterovSolve] Iter: 180 overflow: 0.891371 HPWL: 99646893
75 [NesterovSolve] Iter: 190 overflow: 0.885019 HPWL: 104171396
76 [NesterovSolve] Iter: 200 overflow: 0.873921 HPWL: 111893077
77 [NesterovSolve] Iter: 210 overflow: 0.857049 HPWL: 121263293
78 [NesterovSolve] Iter: 220 overflow: 0.833948 HPWL: 130038575
79 [NesterovSolve] Iter: 230 overflow: 0.806923 HPWL: 139170070
80 [INFO GPL-0100] worst slack 1e+30
81 [WARNING GPL-0102] No slacks found. Timing-driven mode disabled.
82 [NesterovSolve] Iter: 240 overflow: 0.773218 HPWL: 148656853
83 [NesterovSolve] Iter: 250 overflow: 0.735083 HPWL: 157075653
84 [NesterovSolve] Iter: 260 overflow: 0.694398 HPWL: 165271340
85 [NesterovSolve] Iter: 270 overflow: 0.647357 HPWL: 172410178
86 [NesterovSolve] Iter: 280 overflow: 0.601235 HPWL: 179672238
87 [NesterovSolve] Snapshot saved at iter = 280
88 [NesterovSolve] Iter: 290 overflow: 0.551777 HPWL: 186772677
89 [NesterovSolve] Iter: 300 overflow: 0.512053 HPWL: 193375290
90 [NesterovSolve] Iter: 310 overflow: 0.471647 HPWL: 199645584
91 [NesterovSolve] Iter: 320 overflow: 0.434136 HPWL: 204128524
92 [NesterovSolve] Iter: 330 overflow: 0.395216 HPWL: 207784600
93 [NesterovSolve] Iter: 340 overflow: 0.35525 HPWL: 210482471
94 [NesterovSolve] Iter: 350 overflow: 0.324241 HPWL: 211868765
95 [NesterovSolve] Iter: 360 overflow: 0.293428 HPWL: 213140461
96 [NesterovSolve] Iter: 370 overflow: 0.262035 HPWL: 214183137
97 [NesterovSolve] Iter: 380 overflow: 0.226708 HPWL: 215023384
98 [INFO GPL-0075] Routability numCall: 1 inflationIterCnt: 1 bloatIterCnt: 0
99 [INFO GRT-0020] Min routing layer: li1
100 [INFO GRT-0021] Max routing layer: met5
101 [INFO GRT-0022] Global adjustment: 0%
102 [INFO GRT-0023] Grid origin: (0, 0)
103 [WARNING GRT-0043] No OR_DEFAULT vias defined.
104 [INFO GRT-0224] Choose via L1M1_PR as default.
105 [INFO GRT-0224] Choose via M1M2_PR as default.
106 [INFO GRT-0224] Choose via M2M3_PR as default.
107 [INFO GRT-0224] Choose via M3M4_PR as default.
108 [INFO GRT-0224] Choose via M4M5_PR as default.
109 [INFO GRT-0088] Layer li1      Track-Pitch = 0.4600  line-2-Via Pitch: 0.3400
110 [INFO GRT-0088] Layer met1     Track-Pitch = 0.3400  line-2-Via Pitch: 0.3400
```

```

5          Routing      Original      Derated      Resource
7 Layer    Direction    Resources    Resources   Reduction (%)  

3 -----  

9 li1      Vertical     48720        42960       11.82%  

9 met1     Horizontal   64960        48282       25.67%  

1 met2     Vertical     48720        48222       1.02%  

2 met3     Horizontal   32480        31845       1.96%  

3 met4     Vertical     19488        19162       1.67%  

4 met5     Horizontal   6496         6270        3.48%  

5 -----  

5  

7 [INFO GRT-0111] Final number of vias: 13161  

3 [INFO GRT-0112] Final usage 3D: 75778  

9  

9 [INFO GRT-0096] Final congestion report:  

1 Layer      Resource     Demand      Usage (%)    Max H / Max V / Total Overflow  

2 -----  

3 li1        42960        12599       29.33%      0 / 0 / 0  

4 met1       48282        15572       32.25%      0 / 0 / 0  

5 met2       48222        5831        12.09%      0 / 0 / 0  

5 met3       31845        2293        7.20%       0 / 0 / 0  

7 met4       19162        0           0.00%       0 / 0 / 0  

3 met5       6270         0           0.00%       0 / 0 / 0  

9 -----  

9 Total      196741       36295       18.45%      0 / 0 / 0  

1  

2 [INFO GRT-0018] Total wirelength: 349008 um  

3 [INFO GPL-0036] TileLxLy: 0 0  

4 [INFO GPL-0037] TileSize: 6900 6900  

5 [INFO GPL-0038] TileCnt: 56 59  

5 [INFO GPL-0039] numRoutingLayers: 6  

7 [INFO GPL-0040] NumTiles: 3304  

3 [INFO GPL-0063] TotalRouteOverflowH2: 0.0  

9 [INFO GPL-0064] TotalRouteOverflowV2: 0.0  

9 [INFO GPL-0065] OverflowTileCnt2: 0  

1 [INFO GPL-0066] 0.5%RC: 1.0  

7 [INFO GPL-0067] 1.0%RC: 0.0026170216570500

```

This report was generated by resizer tool which performs optional optimizations of the design.

```
[INFO RSZ-0027] Inserted 66 input buffers.  
[INFO RSZ-0028] Inserted 103 output buffers.  
[INFO RSZ-0058] Using max wire length 4269um.  
[INFO RSZ-0034] Found 10 slew violations.  
[INFO RSZ-0036] Found 9 capacitance violations.  
[INFO RSZ-0038] Inserted 29 buffers in 10 nets.  
[INFO RSZ-0039] Resized 5208 instances.  
Placement Analysis  
-----  
total displacement      23897.2 u  
average displacement     2.7 u  
max displacement        30.7 u  
original HPWL          225130.7 u  
legalized HPWL         247337.5 u  
delta HPWL              10 %  
  
[INFO DPL-0020] Mirrored 2338 instances  
[INFO DPL-0021] HPWL before           247337.5 u  
[INFO DPL-0022] HPWL after            244008.9 u  
[INFO DPL-0023] HPWL delta            -1.3 %
```

This report was generated by opendp tool
which performs detailed placement

```
[INFO ODB-0223]      Created 13 technology layers  
[INFO ODB-0224]      Created 25 technology vias  
[INFO ODB-0225]      Created 441 library cells  
[INFO ODB-0226] Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-  
merged_unpadded.lef  
[INFO ODB-0127] Reading DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-  
placement/7-resizer.def  
[INFO ODB-0128] Design: serv_rf_top  
[INFO ODB-0130]      Created 174 pins.  
[INFO ODB-0131]      Created 8729 components and 52845 component-terminals.  
[INFO ODB-0132]      Created 2 special nets and 30828 connections.  
[INFO ODB-0133]      Created 6476 nets and 22015 connections.  
[INFO ODB-0134] Finished DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-  
placement/7-resizer.def  
Placement Analysis  
-----  
total displacement      0.0 u  
average displacement     0.0 u  
max displacement        0.0 u  
original HPWL          244008.9 u  
legalized HPWL         247337.5 u  
delta HPWL              1 %  
  
[INFO DPL-0020] Mirrored 2338 instances  
[INFO DPL-0021] HPWL before           247337.5 u  
[INFO DPL-0022] HPWL after            244008.9 u  
[INFO DPL-0023] HPWL delta            -1.3 %
```

CTS

```
[INFO ODB-0223]      Created 13 technology layers
[INFO ODB-0224]      Created 25 technology vias
[INFO ODB-0225]      Created 441 library cells
[INFO ODB-0226] Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-merged_unpadded.lef
[INFO ODB-0127] Reading DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/results/placement/serv_rf_top.placement.def
[INFO ODB-0128] Design: serv_rf_top
[INFO ODB-0130]      Created 174 pins.
[INFO ODB-0131]      Created 8729 components and 52845 component-terminals.
[INFO ODB-0132]      Created 2 special nets and 30828 connections.
[INFO ODB-0133]      Created 6476 nets and 22015 connections.
[INFO ODB-0134] Finished DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/-results/placement/serv_rf_top.placement.def
[INFO]: Setting output delay to: 4.0
[INFO]: Setting input delay to: 4.0
[INFO]: Setting load to: 0.01765
[INFO]: Configuring cts characterization...
[INFO]: Performing clock tree synthesis...
[INFO]: Looking for the following net(s): clk
[INFO]: Running Clock Tree Synthesis...
.....
```

```
*****
* Create characterization *
*****
[INFO CTS-0038] Number of created patterns = 50000.
[INFO CTS-0038] Number of created patterns = 100000.
[INFO CTS-0038] Number of created patterns = 150000.
[INFO CTS-0038] Number of created patterns = 200000.
[INFO CTS-0038] Number of created patterns = 250000.
[INFO CTS-0038] Number of created patterns = 300000.
[INFO CTS-0039] Number of created patterns = 313632.
[INFO CTS-0084] Compiling LUT
Min. len   Max. len   Min. cap   Max. cap   Min. slew   Max. slew
2          8          1           35          1           295
[WARNING CTS-0043] 6336 wires are pure wire and no slew degradation.
TritonCTS forced slew degradation on these wires.
[INFO CTS-0046]      Num wire segments: 313632
[INFO CTS-0047]      Num keys in characterization LUT: 2030
[INFO CTS-0048]      Actual min input cap: 1
*****
* Find clock roots *
*****
[INFO CTS-0001]  Running TritonCTS with user-specified clock roots: clk
*****
* Populate TritonCTS *
*****
Initializing clock nets
Looking for clock nets in the design
[INFO CTS-0095] Net "clk" found
[INFO CTS-0009] Initializing clock net for : "clk"
```

```

[INFO CTS-0010] Clock net "clk" has 1335 sinks
[INFO CTS-0008] TritonCTS found 1 clock nets.
*****
* Check characterization *
*****
The chacterization used 4 buffer(s) types. All of them are in the loaded DB.
*****
* Build clock trees *
*****
[INFO CTS-0027] Generating H-Tree topology for net clk
[INFO CTS-0028] Tot. number of sinks: 1335
[INFO CTS-0029] Sinks will be clustered in groups of 20 and a maximum diameter of 50.0 um
[INFO CTS-0030] Number of static layers: 0
[INFO CTS-0020] Wire segment unit: 13000 dbu (13 um)
[INFO CTS-0019] Tot. number of sinks after clustering: 139
[INFO CTS-0024] Normalized sink region: [(1.20931, 1.75346), (28.2305, 29.1377)]
[INFO CTS-0025] Width: 27.0212
[INFO CTS-0026] Height: 27.3842
Level 1
    Direction: Vertical
    # sinks per sub-region: 70
    Sub-region size: 27.0212 X 13.6921
[INFO CTS-0034] Segment length (rounded): 6
    Key: 432 outSlew: 2 load: 1 length: 6 isBuffered: false
Level 2
    Direction: Horizontal
    # sinks per sub-region: 35
    Sub-region size: 13.5106 X 13.6921
[INFO CTS-0034] Segment length (rounded): 6
    Key: 432 outSlew: 2 load: 1 length: 6 isBuffered: false
Level 3
    Direction: Vertical
    # sinks per sub-region: 18
    Sub-region size: 13.5106 X 6.8461
[INFO CTS-0034] Segment length (rounded): 4

```

```

Level 4
    Direction: Horizontal
    # sinks per sub-region: 9
    Sub-region size: 6.7553 X 6.8461
[INFO CTS-0034] Segment length (rounded): 4
    Key: 138 outSlew: 11 load: 1 length: 4 isBuffered: true
[INFO CTS-0032] Stop criterion found. Max number of sinks is (15)
Building clock sub nets...
[INFO CTS-0035] Number of sinks covered: 139
[INFO CTS-0033] Clock topology of net "clk" done.
*****
* Post CTS opt *
*****
[INFO CTS-0036] Avg. source sink dist: 20086.06 dbu.
[INFO CTS-0037] Num outlier sinks: 0
*****
* Write data to DB *
*****
Writing clock net "clk" to DB
[INFO CTS-0018] Created 156 clock buffers.
[INFO CTS-0012] Minimum number of buffers in the clock path: 3.
[INFO CTS-0013] Maximum number of buffers in the clock path: 3.
[INFO CTS-0015] Created 156 clock nets.
[INFO CTS-0016] Fanout distribution for the current clock = 3:2, 4:1, 5:3, 6:8, 7:13,
B:28, 9:29, 10:21, 11:18, 12:12, 13:12, 14:4, 15:3, 16:1.
[INFO CTS-0017] Max level of the clock tree: 4.
Post DB write clock net "clk" report
[INFO CTS-0091] Sinks after db write = 1335 (Leaf Buffers = 139)
[INFO CTS-0092] Avg Sink Wire Length = 596.0 um
[INFO CTS-0094] Min path depth = 3 Max path depth = 3
... End of TritonCTS execution.
[INFO]: Repairing long wires on clock nets...
[INFO RSZ-0058] Using max wire length 3048um.
[INFO]: Legalizing...
Placement Analysis

```

```
-----  
total displacement      1846.6 u  
average displacement     0.2 u  
max displacement        9.6 u  
original HPWL          254289.4 u  
legalized HPWL         258782.6 u  
delta HPWL              2 %  
  
[INFO DPL-0020] Mirrored 2429 instances  
[INFO DPL-0021] HPWL before           258782.6 u  
[INFO DPL-0022] HPWL after            254964.9 u  
[INFO DPL-0023] HPWL delta           -1.5 %  
timing_report
```

Routing

This report was generated by fastroute tool which performs global routing to generate a guide file for the detailed router.

Layer	Routing Direction	Original Resources	Derated Resources	Resource Reduction (%)
li1	Vertical	48720	1129	97.68%
met1	Horizontal	64960	48416	25.47%
met2	Vertical	48720	48222	1.02%
met3	Horizontal	32480	31845	1.96%
met4	Vertical	19488	19162	1.67%
met5	Horizontal	6496	6270	3.48%

```
[INFO GRT-0191] Wirelength: 41012, Wirelength1: 0
[INFO GRT-0192] Number of segments: 14556
[INFO GRT-0193] Number of shifts: 0
[INFO GRT-0097] First L Route.
[INFO GRT-0191] Wirelength: 41023, Wirelength1: 41023
[INFO GRT-0192] Number of segments: 14488
[INFO GRT-0193] Number of shifts: 493
[INFO GRT-0135] Overflow Report:
[INFO GRT-0136] Total hCap : 86531
[INFO GRT-0137] Total vCap : 68513
[INFO GRT-0138] Total usage : 41023
[INFO GRT-0139] Max H overflow : 0
[INFO GRT-0140] Max V overflow : 0
[INFO GRT-0141] Max overflow : 0
[INFO GRT-0142] Num overflow edges: 0
[INFO GRT-0143] H overflow : 0
[INFO GRT-0144] V overflow : 0
[INFO GRT-0145] Final overflow : 0

[INFO GRT-0098] Second L Route.
[INFO GRT-0135] Overflow Report:
[INFO GRT-0136] Total hCap : 86531
[INFO GRT-0137] Total vCap : 68513
[INFO GRT-0138] Total usage : 41023
[INFO GRT-0139] Max H overflow : 0
[INFO GRT-0140] Max V overflow : 0
[INFO GRT-0141] Max overflow : 0
[INFO GRT-0142] Num overflow edges: 0
[INFO GRT-0143] H overflow : 0
[INFO GRT-0144] V overflow : 0
[INFO GRT-0145] Final overflow : 0

[INFO GRT-0099] First Z Route.
[INFO GRT-0135] Overflow Report:
```

```
[INFO GRT-0099] First Z Route.  
[INFO GRT-0135] Overflow Report:  
[INFO GRT-0136] Total hCap : 86531  
[INFO GRT-0137] Total vCap : 68513  
[INFO GRT-0138] Total usage : 41023  
[INFO GRT-0139] Max H overflow : 0  
[INFO GRT-0140] Max V overflow : 0  
[INFO GRT-0141] Max overflow : 0  
[INFO GRT-0142] Num overflow edges: 0  
[INFO GRT-0143] H overflow : 0  
[INFO GRT-0144] V overflow : 0  
[INFO GRT-0145] Final overflow : 0  
  
[INFO GRT-0100] LV routing round 0, enlarge 10.  
[INFO GRT-0182] 10 threshold, 10 expand.  
[INFO GRT-0126] Overflow report:  
[INFO GRT-0127] Total usage : 41023  
[INFO GRT-0128] Max H overflow : 0  
[INFO GRT-0129] Max V overflow : 3  
[INFO GRT-0130] Max overflow : 3  
[INFO GRT-0131] Number overflow edges: 2  
[INFO GRT-0132] H overflow : 0  
[INFO GRT-0133] V overflow : 5  
[INFO GRT-0134] Final overflow : 5  
  
[INFO GRT-0100] LV routing round 1, enlarge 15.  
[INFO GRT-0182] 5 threshold, 15 expand.  
[INFO GRT-0126] Overflow report:  
[INFO GRT-0127] Total usage : 41039  
[INFO GRT-0128] Max H overflow : 0  
[INFO GRT-0129] Max V overflow : 0  
[INFO GRT-0130] Max overflow : 0  
[INFO GRT-0131] Number overflow edges: 0  
[INFO GRT-0132] H overflow : 0  
[INFO GRT-0133] V overflow : 0  
[INFO GRT-0134] Final overflow : 0  
  
[INFO GRT-0100] LV routing round 2, enlarge 20.  
[INFO GRT-0182] 1 threshold, 20 expand.  
[INFO GRT-0126] Overflow report:  
[INFO GRT-0127] Total usage : 41039  
[INFO GRT-0128] Max H overflow : 0  
[INFO GRT-0129] Max V overflow : 0  
[INFO GRT-0130] Max overflow : 0  
[INFO GRT-0131] Number overflow edges: 0  
[INFO GRT-0132] H overflow : 0  
[INFO GRT-0133] V overflow : 0  
[INFO GRT-0134] Final overflow : 0
```

This report was generated by tritonroute tool

[INFO DRT-0149] Reading Tech And Libs

```
units:      1000
#layers:    13
#macros:    441
#vias:      25
#viarulegen: 25
```

[INFO DRT-0150] Reading Design

```
design:      serv_rf_top
die area:   ( 0 0 ) ( 389910 400630 )
trackPts:    12
defvias:     4
#components: 19962
#terminals:  174
#snets:      2
#nets:       6632
```

[INFO DRT-0151] Reading Guide

```
#guides:    49641
```

[INFO DRT-0167] List of default vias:

```
Layer mcon
  default via: L1M1_PR_MR
Layer via
  default via: M1M2_PR
Layer via2
  default via: M2M3_PR_M
Layer via3
  default via: M3M4_PR_M
Layer via4
  default via: M4M5_PR
```

[INFO DRT-0162] libcell analysis ...

[INFO DRT-0163] instance analysis ...
 complete 10000 instances

[INFO DRT-0164] # unique instances = 175

[INFO DRT-0168] Init region query ...

[INFO DRT-0018] complete 10000 insts

[INFO DRT-0024] complete FR_MASTERSLICE

[INFO DRT-0024] complete FR_VIA

[INFO DRT-0024] complete li1

[INFO DRT-0024] complete mcon

INFO DRT-0024] complete mcon

```
[INFO DRT-0164] # unique instances = 175
[INFO DRT-0168] Init region query ...
[INFO DRT-0018]   complete 10000 insts
[INFO DRT-0024]   complete FR_MASTERSLICE
[INFO DRT-0024]   complete FR_VIA
[INFO DRT-0024]   complete li1
[INFO DRT-0024]   complete mcon
[INFO DRT-0024]   complete met1
[INFO DRT-0024]   complete via
[INFO DRT-0024]   complete met2
[INFO DRT-0024]   complete via2
[INFO DRT-0024]   complete met3
[INFO DRT-0024]   complete via3
[INFO DRT-0024]   complete met4
[INFO DRT-0024]   complete via4
[INFO DRT-0024]   complete met5
[INFO DRT-0033] FR_MASTERSLICE shape region query size = 0
[INFO DRT-0033] FR_VIA shape region query size = 0
[INFO DRT-0033] li1 shape region query size = 276094
[INFO DRT-0033] mcon shape region query size = 239898
[INFO DRT-0033] met1 shape region query size = 67563
[INFO DRT-0033] via shape region query size = 1400
[INFO DRT-0033] met2 shape region query size = 802
[INFO DRT-0033] via2 shape region query size = 1400
[INFO DRT-0033] met3 shape region query size = 770
[INFO DRT-0033] via3 shape region query size = 1400
[INFO DRT-0033] met4 shape region query size = 368
[INFO DRT-0033] via4 shape region query size = 13
[INFO DRT-0033] met5 shape region query size = 23
[INFO DRT-0165] start pin access
[INFO DRT-0076]   complete 100 pins
[INFO DRT-0076]   complete 200 pins
[INFO DRT-0076]   complete 300 pins
[INFO DRT-0076]   complete 400 pins
[INFO DRT-0076]   complete 500 pins
[INFO DRT-0078]   complete 596 pins
[INFO DRT-0079]   complete 100 unique inst patterns
[INFO DRT-0081]   complete 169 unique inst patterns
[INFO DRT-0082]   complete 1000 groups
[INFO DRT-0082]   complete 2000 groups
[INFO DRT-0082]   complete 3000 groups
[INFO DRT-0082]   complete 4000 groups
[INFO DRT-0082]   complete 5000 groups
[INFO DRT-0082]   complete 6000 groups
[INFO DRT-0084]   complete 6304 groups
#scanned instances      = 19962
```

```

completing 100% with 0 violations
elapsed time = 00:00:00, memory = 600.51 (MB)
[INFO DRT-0199] number of violations = 0
[INFO DRT-0267] cpu time = 00:00:00, elapsed time = 00:00:00, memory = 600.51 (MB), peak = 600.51 (MB)
total wire length = 304802 um
total wire length on LAYER li1 = 1818 um
total wire length on LAYER met1 = 131452 um
total wire length on LAYER met2 = 141866 um
total wire length on LAYER met3 = 26677 um
total wire length on LAYER met4 = 2987 um
total wire length on LAYER met5 = 0 um
total number of vias = 53352
up-via summary (total 53352):

-----
FR_MASTERSLICE      0
    li1    22779
    met1   27516
    met2   2850
    met3    207
    met4     0
-----
53352

[INFO DRT-0198] complete detail routing
total wire length = 304802 um
total wire length on LAYER li1 = 1818 um
total wire length on LAYER met1 = 131452 um
total wire length on LAYER met2 = 141866 um
total wire length on LAYER met3 = 26677 um
total wire length on LAYER met4 = 2987 um
total wire length on LAYER met5 = 0 um
total number of vias = 53352
up-via summary (total 53352):

-----
FR_MASTERSLICE      0
    li1    22779
    met1   27516
    met2   2850
    met3    207
    met4     0
-----
53352

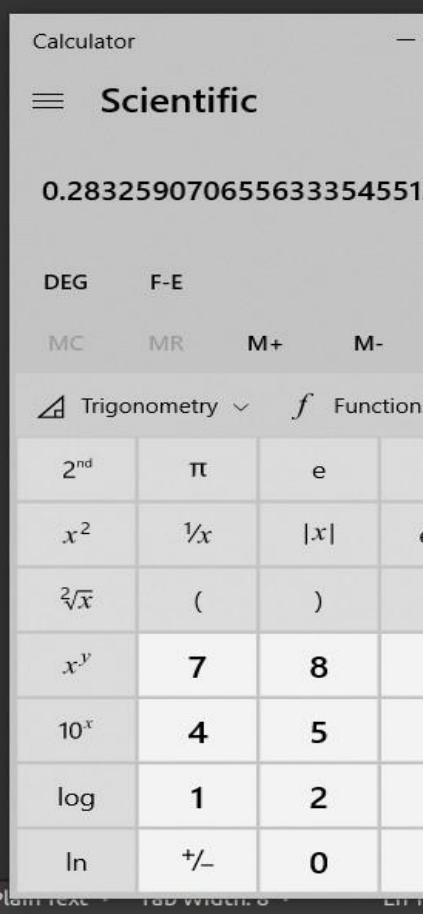
[INFO DRT-0267] cpu time = 00:16:07, elapsed time = 00:17:52, memory = 600.51 (MB), peak = 600.51 (MB)
[INFO DRT-0180] post processing ...
Saving to /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/results/routing/15-serv_rf_top.def

```

Antenna report

```
.OpenROAD 1 4d4d7205fd0292dbf3fae55fad9109b3f0bd5786
! This program is licensed under the BSD-3 license. See the LICENSE file for details.
! Components of this program may be licensed under more restrictive licenses which must be
! honored.
! [INFO ODB-0222] Reading LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-
! merged_unpadded.lef
! [WARNING ODB-0220] WARNING (LEFPARS-2036): SOURCE statement is obsolete in version 5.6 and
! later.
! The LEF parser will ignore this statement.
! To avoid this warning in the future, remove this statement from the LEF file with version 5.6
! or later. See file /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/merged_unpadded.lef
! at line 794.
!
! [INFO ODB-0223]      Created 13 technology layers
! [INFO ODB-0224]      Created 25 technology vias
! [INFO ODB-0225]      Created 441 library cells
! [INFO ODB-0226] Finished LEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/tmp/-
! merged_unpadded.lef
! [WARNING ORD-0033] -order_wires is deprecated.
! [INFO ODB-0127] Reading DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/results/-/
! routing/15-serv_rf_top.def
! [INFO ODB-0128] Design: serv_rf_top
! [INFO ODB-0130]      Created 174 pins.
! [INFO ODB-0131]      Created 19962 components and 98209 component-terminals.
! [INFO ODB-0132]      Created 2 special nets and 75760 connections.
! [INFO ODB-0133]      Created 6632 nets and 22447 connections.
! [INFO ODB-0134] Finished DEF file: /openLANE_flow/designs/serv_rf_top/runs/11-08_11-58/results/-/
! routing/15-serv_rf_top.def
! Notice 0: Split top of 2286 T shapes.
! [INFO ANT-0001] Found 21 pin violations.
! [INFO ANT-0002] Found 20 net violations in 6632 nets.
```

DFF Ratio



```

5 Number of wires: 4624
6 Number of wire bits: 4782
7 Number of public wires: 1264
8 Number of public wire bits: 1422
9 Number of memories: 0
10 Number of memory bits: 0
11 Number of processes: 0
12 Number of cells: 4713
13   sky130_fd_sc_hd_a2111o_2 1
14   sky130_fd_sc_hd_a211o_2 1
15   sky130_fd_sc_hd_a211oi_2 2
16   sky130_fd_sc_hd_a21bo_2 2
17   sky130_fd_sc_hd_a21o_2 2
18   sky130_fd_sc_hd_a21oi_2 4
19   sky130_fd_sc_hd_a221o_2 3
20   sky130_fd_sc_hd_a221oi_2 1
21   sky130_fd_sc_hd_a22o_2 46
22   sky130_fd_sc_hd_a2bb2o_2 4
23   sky130_fd_sc_hd_a311o_2 1
24   sky130_fd_sc_hd_a31o_2 2
25   sky130_fd_sc_hd_a31oi_2 1
26   sky130_fd_sc_hd_a32o_2 5
27   sky130_fd_sc_hd_a41o_2 1
28   sky130_fd_sc_hd_and2_2 9
29   sky130_fd_sc_hd_and2b_2 1
30   sky130_fd_sc_hd_and3_2 4
31   sky130_fd_sc_hd_and4_2 3
32   sky130_fd_sc_hd_buf_1 646
33   sky130_fd_sc_hd_comb_1 2
34   sky130_fd_sc_hd_dfxtcp_2 1335
35   sky130_fd_sc_hd_inv_2 98
36   sky130_fd_sc_hd_mux2_1 106
37   sky130_fd_sc_hd_mux2_2 1171
38   sky130_fd_sc_hd_mux4_1 382
39   sky130_fd_sc_hd_nand2_2 8
40   sky130_fd_sc_hd_nand2b_2 4
41   sky130_fd_sc_hd_nor2_2 29
42

```

DFF Ratio = No of DFF/ No of Cells

Generation of Power Report

This report consists of power parameters like **Internal power, Switching Power, Leakage Power and Total power consumed by Sequential, Combinational, macro and pads.**

This Report was generated for default clock period of 20ns and Frequency of design as 50MHz. This was based on

Maximum Operating Frequency Calculations section.

Synthesis

- Using Automated Flow of OpenLane/ Interactive Flow of OpenLane

1 Group	Internal Power	Switching Power	Leakage Power	Total Power	
2					
3 -----					
4 Sequential	1.22e-03	1.10e-04	2.05e-05	1.35e-03	71.5%
5 Combinational	3.44e-04	1.70e-04	2.27e-05	5.36e-04	28.5%
6 Macro	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.0%
7 Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.0%
8 -----					
9 Total	1.56e-03	2.80e-04	4.32e-05	1.88e-03	100.0%
10	82.8%	14.9%	2.3%		

Here we see the **Generation of Power Report**

$$\text{Power of the Design} = \alpha \cdot C \cdot V^2 \cdot F = C \cdot V^2 \cdot F$$

C= Load Capacitance, V= Supply Voltage,
 α =Switching activity. So, from here F which is frequency
of design is $F \propto 1/T$.

Generation of Timing Report

STA (Static Timing Analysis)

This report consists of timing analysis which shows the **maximum and minimum slack time** which is calculated by:

$$\text{Slack time} = \text{Data arrival time} - \text{Data required time}$$

This Report was generated for default clock period of 20ns and Frequency of design as 50MHz. This was based on

Maximum Operating Frequency Calculations section.

Synthesis

Min-max slack

```

1 Startpoint: _6742_ (rising edge-triggered flip-flop clocked by clk)
2 Endpoint: _6742_ (rising edge-triggered flip-flop clocked by clk)
3 Path Group: clk
4 Path Type: min
5
6 Fanout      Cap      Slew      Delay      Time      Description
7 -----
8          0.00      0.00      0.00      0.00      clock clk (rise edge)
9          0.00      0.00      0.00      0.00      clock network delay (ideal)
10         0.00      0.00      0.00 ^ _6743 /CLK (sky130_fd_sc_hd_dfxtp_2)
11         0.02      0.18      0.18      0.18 ^ _6743/_Q (sky130_fd_sc_hd_dfxtp_2)
12         2    0.00      0.02      0.00      0.18 rf_ram_if.wdata1_r[1] (net)
13          0.02      0.00      0.18 ^ _6742/_D (sky130_fd_sc_hd_dfxtp_2)
14          0.18      0.18      0.18      0.18      data arrival time
15
16         0.00      0.00      0.00      0.00      clock clk (rise edge)
17         0.00      0.00      0.00      0.00      clock network delay (ideal)
18         0.00      0.00      0.00      0.00      clock reconvergence pessimism
19         0.00      0.00      0.00 ^ _6742 /CLK (sky130_fd_sc_hd_dfxtp_2)
20          -0.02     -0.02      0.00      -0.02      library hold time
21          -0.02     -0.02      0.00      -0.02      data required time
22
23          -0.02     -0.02      0.00      -0.02      data required time
24          -0.18     -0.18      0.00      -0.18      data arrival time
25
26          0.20      0.20      0.00      0.20      slack (MET)
27
28
29 Startpoint: _8030_ (rising edge-triggered flip-flop clocked by clk)
30 Endpoint: _6738_ (rising edge-triggered flip-flop clocked by clk)
31 Path Group: clk
32 Path Type: max
33
34 Fanout      Cap      Slew      Delay      Time      Description
35 -----
36         0.00      0.00      0.00      0.00      clock clk (rise edge)
37         0.00      0.00      0.00      0.00      clock network delay (ideal)
38         0.00      0.00      0.00 ^ _8030 /CLK (sky130_fd_sc_hd_dfxtp_2)
39         8.74      6.78      6.78      6.78 ^ _8030/_Q (sky130_fd_sc_hd_dfxtp_2)
40         290    1.13      8.74      0.00      6.78 rf_ram_i_raddr[0] (net)
41          8.74      0.00      6.78 ^ _6680 /S0 (sky130_fd_sc_hd_mux4_1)
42          0.17      2.54      9.32 v _6680 /X (sky130_fd_sc_hd_mux4_1)
43         1    0.00      0.17      0.00      9.32 _0147 (net)
44          0.17      0.00      9.32 v _6683 /A1 (sky130_fd_sc_hd_mux4_1)
45          0.14      1.13      10.45 v _6683 /X (sky130_fd_sc_hd_mux4_1)
46         1    0.00      0.14      0.00      10.45 _0150 (net)
47          0.14      0.00      10.45 v _6694 /A1 (sky130_fd_sc_hd_mux4_1)
48          0.13      1.11      11.56 v _6694 /X (sky130_fd_sc_hd_mux4_1)
49         1    0.00      0.13      0.00      11.56 _0156 (net)

```

```

29 Startpoint: _8030_ (rising edge-triggered flip-flop clocked by clk)
30 Endpoint: _6738_ (rising edge-triggered flip-flop clocked by clk)
31 Path Group: clk
32 Path Type: max
33
34 Fanout      Cap      Slew      Delay      Time      Description
35 -----
36           0.00      0.00      0.00      0.00      clock clk (rise edge)
37           0.00      0.00      0.00      0.00      clock network delay (ideal)
38           0.00      0.00      0.00      0.00      ^ _8030_/CLK (sky130_fd_sc_hd_
39           8.74      6.78      6.78      6.78      ^ _8030_/Q (sky130_fd_sc_hd_d
40   290     1.13
41           8.74      0.00      6.78      6.78      rf_ram.i_raddr[0] (net)
42           0.17      2.54      9.32      v _6680_/X (sky130_fd_sc_hd_m
43   1     0.00
44           0.17      0.00      9.32      v _6683_/A1 (sky130_fd_sc_hd_r
45           0.14      1.13     10.45      v _6683_/X (sky130_fd_sc_hd_m
46   1     0.00
47           0.14      0.00     10.45      v _6694_/A1 (sky130_fd_sc_hd_r
48           0.13      1.11     11.56      v _6694_/X (sky130_fd_sc_hd_m
49   1     0.00
50           0.13      0.00     11.56      v _6737_/A1 (sky130_fd_sc_hd_r
51           0.14      1.12     12.68      v _6737_/X (sky130_fd_sc_hd_m
52   1     0.00
53           0.14      0.00     12.68      v _6354_/A1 (sky130_fd_sc_hd_r
54           0.10      0.66     13.34      v _6354_/X (sky130_fd_sc_hd_m
55   1     0.00
56           0.10      0.00     13.34      v _6355_/A1 (sky130_fd_sc_hd_r
57           0.10      0.64     13.98      v _6355_/X (sky130_fd_sc_hd_m
58   1     0.00
59           0.10      0.00     13.98      v _6738_/D (sky130_fd_sc_hd_d
60           0.00      0.00     13.98      data arrival time
61
62           0.00     20.00     20.00      clock clk (rise edge)
63           0.00     20.00     20.00      clock network delay (ideal)
64           0.00     20.00     20.00      clock reconvergence pessimism
65           20.00      -0.31    19.69      ^ _6738_/CLK (sky130_fd_sc_hd_
66           19.69      library setup time
67           19.69      data required time
68 -----
69           19.69      data required time
70           -13.98     data arrival time
71 -----
72           5.71      slack (MET)

```

tns – refers to total negative slack



wns -refers to worst negative slack



Placement

Min-max slack

```
Startpoint: _09733_ (rising edge-triggered flip-flop clocked by clk)
Endpoint: _09732_ (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min

Fanout      Cap      Slew      Delay      Time      Description
-----
2          0.00
           0.00      0.00      0.00  clock clk (rise edge)
                   0.00      0.00  clock network delay (ideal)
           0.00      0.00  0.00 ^ _09733_/_CLK (sky130_fd_sc_hd_dfxtpl_2)
           0.02      0.19  0.19 v _09733_/_Q (sky130_fd_sc_hd_dfxtpl_2)
                           rf_ram_if.wdata1_r[1] (net)
           0.02      0.00  0.19 v _09732_/_D (sky130_fd_sc_hd_dfxtpl_2)
                           0.19      data arrival time

           0.00      0.00      0.00  clock clk (rise edge)
                   0.00      0.00  clock network delay (ideal)
                   0.00      0.00  clock reconvergence pessimism
                   0.00 ^ _09732_/_CLK (sky130_fd_sc_hd_dfxtpl_2)
           -0.19     -0.19  library hold time
                           -0.19      data required time
-----
```

-0.19 data required time
-0.19 data arrival time

```
-----
```

0.39 slack (MET)

Fanout	Cap	Slew	Delay	Time	Description
				0.00	clock clk (rise edge)
				0.00	clock network delay (ideal)
				0.00	0.00 ^ _11020_/_CLK (sky130_fd_sc_hd_dfxtp_2)
293	1.63		12.64	11.19	11.19 ^ _11020_/_Q (sky130_fd_sc_hd_dfxtp_2) rf_ram.i_raddr[0] (net)
			12.64	0.11	11.29 ^ _09659_/_S0 (sky130_fd_sc_hd_mux4_1)
1	0.01		0.26	3.13	14.43 v _09659_/_X (sky130_fd_sc_hd_mux4_1) _00352_ (net)
1	0.01		0.26	0.00	14.43 v _09662_/_A1 (sky130_fd_sc_hd_mux4_1)
1	0.01		0.23	1.31	15.74 v _09662_/_X (sky130_fd_sc_hd_mux4_1) _00355_ (net)
1	0.02		0.23	0.00	15.74 v _09663_/_A3 (sky130_fd_sc_hd_mux4_1)
1	0.01		0.35	1.40	17.14 v _09663_/_X (sky130_fd_sc_hd_mux4_1) _00371_ (net)
1	0.01		0.35	0.00	17.14 v _09727_/_A0 (sky130_fd_sc_hd_mux4_1)
1	0.01		0.22	1.31	18.45 v _09727_/_X (sky130_fd_sc_hd_mux4_1) _00372_ (net)
1	0.01		0.22	0.00	18.45 v _09242_/_A1 (sky130_fd_sc_hd_mux2_1)
1	0.01		0.19	0.81	19.26 v _09242_/_X (sky130_fd_sc_hd_mux2_1) _00458_ (net)
1	0.00		0.19	0.00	19.26 v _09243_/_A1 (sky130_fd_sc_hd_mux2_1)
1	0.00		0.11	0.68	19.94 v _09243_/_X (sky130_fd_sc_hd_mux2_1) rf_ram.memory\$rdreg[0]\$d[1] (net)
			0.11	0.00	19.94 v _09729_/_D (sky130_fd_sc_hd_dfxtp_2) 19.94 data arrival time
			0.00	20.00	clock clk (rise edge)
			0.00	20.00	clock network delay (ideal)
			0.00	20.00	clock reconvergence pessimism
			0.00	20.00	^ _09729_/_CLK (sky130_fd_sc_hd_dfxtp_2)

0.55	20.55	library setup time
	20.55	data required time
	20.55	data required time
-19.94		data arrival time
0.61		slack (MET)

```
min_max_report_end
check_report
No paths found.
check_report_end
wns 0.00
tns 0.00
```

CTS

Min-max slack

Startpoint: _09733_ (rising edge-triggered flip-flop clocked by clk)
Endpoint: _09732_ (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min

Fanout	Cap	Slew	Delay	Time	Description
2	0.00	0.00	0.00	0.00	clock clk (rise edge)
		0.00	0.00	0.00	clock network delay (ideal)
		0.00	0.00	0.00	$\wedge _09733_/{\text{CLK}}$ (sky130_fd_sc_hd_dfxtpl_1)
		0.04	0.18	0.18	$\wedge _09733_{/Q}$ (sky130_fd_sc_hd_dfxtpl_1)
					rf_ram_if.wdata1_r[1] (net)
	0.04	0.04	0.00	0.18	$\wedge _09732_/{\text{D}}$ (sky130_fd_sc_hd_dfxtpl_1)
				0.18	data arrival time
		0.00	0.00	0.00	clock clk (rise edge)
		0.00	0.00	0.00	clock network delay (ideal)
		0.00	0.00	0.00	clock reconvergence pessimism
	-0.02	0.00	0.00	0.00	$\wedge _09732_/{\text{CLK}}$ (sky130_fd_sc_hd_dfxtpl_1)
		-0.02	-0.02	-0.02	library hold time
				-0.02	data required time
				-0.02	data required time
				-0.18	data arrival time
				0.20	slack (MET)

					Startpoint: _10763_ (rising edge-triggered flip-flop clocked by clk)
					Endpoint: _10975_ (rising edge-triggered flip-flop clocked by clk)
					Path Group: clk
					Path Type: max
Fanout	Cap	Slew	Delay	Time	Description

					0.00 0.00 0.00 clock clk (rise edge)
					0.00 0.00 0.00 clock network delay (ideal)
					0.00 0.00 0.00 ^ _10763_/CLK (sky130_fd_sc_hd_dfxtpl_1)
					0.14 0.62 0.62 v _10763/_Q (sky130_fd_sc_hd_dfxtpl_1)
					cpu.csr_d_sel (net)
					0.14 0.00 0.62 v _05072_/A (sky130_fd_sc_hd_nor3_2)
					0.43 0.49 1.11 ^ _05072_/Y (sky130_fd_sc_hd_nor3_2)
					02035 (net)
4	0.01				0.43 0.00 1.11 ^ _05073_/A (sky130_fd_sc_hd_clkbuf_2)
					0.20 0.43 1.54 ^ _05073_/X (sky130_fd_sc_hd_clkbuf_2)
					02036 (net)
					0.20 0.00 1.54 ^ _05077_/A (sky130_fd_sc_hd_nand3_2)
					0.22 0.26 1.81 v _05077_/Y (sky130_fd_sc_hd_nand3_2)
					02040 (net)
					0.22 0.00 1.81 v _05078_/A (sky130_fd_sc_hd_clkbuf_2)
					0.16 0.35 2.16 v _05078_/X (sky130_fd_sc_hd_clkbuf_2)
					02041 (net)
					0.16 0.00 2.16 v _06033_/D1 (sky130_fd_sc_hd_o2111ai_1)
					0.30 0.16 2.32 ^ _06033_/Y (sky130_fd_sc_hd_o2111ai_1)
					02739 (net)
1	0.00				0.30 0.00 2.32 ^ _06034_/B1 (sky130_fd_sc_hd_a21oi_1)
					0.19 0.21 2.53 v _06034_/Y (sky130_fd_sc_hd_a21oi_1)
					02740 (net)
					0.19 0.00 2.53 v _06037_/A2 (sky130_fd_sc_hd_o21bai_4)
					0.38 0.41 2.94 ^ _06037_/Y (sky130_fd_sc_hd_o21bai_4)
					02743 (net)
					0.38 0.00 2.94 ^ _06038_/A (sky130_fd_sc_hd_dlymetal6s2s_1)
					0.26 0.42 3.37 ^ _06038_/X (sky130_fd_sc_hd_dlymetal6s2s_1)
					02744 (net)
5	0.02				0.26 0.00 3.37 ^ _06039/A (skv130_fd_sc_hd_buf_2)

5	0.02				_02744_ (net)
		0.26	0.00	3.37 ^	_06039_/A (sky130_fd_sc_hd_buf_2)
		0.26	0.42	3.79 ^	_06039_/X (sky130_fd_sc_hd_buf_2)
5	0.03				_02745_ (net)
		0.26	0.00	3.79 ^	_09121_/A (sky130_fd_sc_hd_and2_1)
		0.18	0.38	4.17 ^	_09121_/X (sky130_fd_sc_hd_and2_1)
2	0.01				_00492_ (net)
		0.18	0.00	4.17 ^	_09122_/A (sky130_fd_sc_hd_inv_2)
		0.04	0.08	4.25 v	_09122_/Y (sky130_fd_sc_hd_inv_2)
1	0.00				_00481_ (net)
		0.04	0.00	4.25 v	_09265_/A0 (sky130_fd_sc_hd_mux2_1)
		0.13	0.64	4.89 v	_09265_/X (sky130_fd_sc_hd_mux2_1)
2	0.00				_00488_ (net)
		0.13	0.00	4.89 v	_09136_/A_N (sky130_fd_sc_hd_nand3b_1)
		0.25	0.45	5.33 v	_09136_/Y (sky130_fd_sc_hd_nand3b_1)
3	0.01				_04798_ (net)
		0.25	0.00	5.33 v	_09138_/B1 (sky130_fd_sc_hd_o211a_1)
		0.10	0.35	5.68 v	_09138_/X (sky130_fd_sc_hd_o211a_1)
2	0.01				_00490_ (net)
		0.10	0.00	5.68 v	_09249_/A0 (sky130_fd_sc_hd_mux2_1)
		0.11	0.63	6.31 v	_09249_/X (sky130_fd_sc_hd_mux2_1)
1	0.00				_00506_ (net)
		0.11	0.00	6.31 v	_09250_/A1 (sky130_fd_sc_hd_mux2_1)
		0.16	0.73	7.04 v	_09250_/X (sky130_fd_sc_hd_mux2_1)
2	0.01				cpu.o_wdata0 (net)
		0.16	0.00	7.04 v	_09245_/A1 (sky130_fd_sc_hd_mux2_2)
		0.19	0.72	7.75 v	_09245_/X (sky130_fd_sc_hd_mux2_2)
5	0.03				rf_ram.i_wdata[1] (net)
		0.19	0.00	7.76 v	_05185_/A (sky130_fd_sc_hd_inv_2)
		0.24	0.27	8.03 ^	_05185_/Y (sky130_fd_sc_hd_inv_2)
4	0.03				_02135_ (net)
		0.24	0.00	8.03 ^	_05186_/A (sky130_fd_sc_hd_buf_2)
		0.26	0.41	8.44 ^	_05186_/X (sky130_fd_sc_hd_buf_2)
5	0.03				_02136_ (net)
		0.26	0.00	8.44 ^	_05187_/A (sky130_fd_sc_hd_dlymetal6s2s_1)
		0.28	0.40	8.84 ^	_05187_/X (sky130_fd_sc_hd_dlymetal6s2s_1)
5	0.02				02137 (net)

```

244      5  0.02          0.28  0.00   8.84 ^ _02137_ (net)
245                  0.23  0.46   9.30 ^ _05197_/A (sky130_fd_sc_hd__clkbuf_4)
246                  0.23  0.01   9.31 ^ _05197_/X (sky130_fd_sc_hd__clkbuf_4)
247      5  0.05          0.29  0.40   9.70 ^ _02145_ (net)
248                  0.29  0.01   9.71 ^ _05309_/A (sky130_fd_sc_hd__dlymetal6s2s_1)
249                  0.29  0.40   9.70 ^ _05309_/X (sky130_fd_sc_hd__dlymetal6s2s_1)
250      5  0.02          0.29  0.00   9.91 v _02237_ (net)
251                  0.09  0.21   9.91 v _05322_/A1 (sky130_fd_sc_hd_o21ai_1)
252                  0.09  0.21   9.91 v _05322_/Y (sky130_fd_sc_hd_o21ai_1)
253      1  0.00          0.09  0.00   9.91 v _01782_ (net)
254                  0.09  0.00   9.91 v _10975_/D (sky130_fd_sc_hd_dfxtip_1)
255                  0.09          data arrival time
256
257          0.00  20.00  20.00  clock clk (rise edge)
258          0.00  20.00  20.00  clock network delay (ideal)
259          0.00  20.00  20.00  clock reconvergence pessimism
260          20.00 ^ _10975_/CLK (sky130_fd_sc_hd_dfxtip_1)
261          -0.30  19.70  19.70  library setup time
262          19.70  19.70  19.70  data required time
263 -----
264          19.70  19.70  19.70  data required time
265          -9.91  -9.91  -9.91  data arrival time
266 -----
267          9.79   9.79   9.79   slack (MET)
268
269
270 min_max_report_end
271 check_report
272 No paths found.
273 check_report_end
274 wns 0.00
275 tns 0.00

```

Routing

Startpoint: _09733_ (rising edge-triggered flip-flop clocked by clk)

Endpoint: _09732_ (rising edge-triggered flip-flop clocked by clk)

Path Group: clk

Path Type: min

Fanout	Cap	Slew	Delay	Time	Description

		0.00	0.00	0.00	clock clk (rise edge)
		0.00	0.00	0.00	clock network delay (ideal)
		0.00	0.00	0.00 ^ _09733_/CLK (sky130_fd_sc_hd_dfxtip_1)	
2	0.01	0.05	0.21	0.21 ^ _09733_/Q (sky130_fd_sc_hd_dfxtip_1)	
		0.05	0.00	0.21 ^ rrf_ram_if.wdata1_r[1] (net)	
2	0.01	0.05	0.00	0.21 ^ _09732_/D (sky130_fd_sc_hd_dfxtip_1)	
		0.05	0.00	0.21 data arrival time	
		0.00	0.00	0.00	clock clk (rise edge)
		0.00	0.00	0.00	clock network delay (ideal)
		0.00	0.00	0.00	clock reconvergence pessimism
		0.00	0.00	0.00 ^ _09732_/CLK (sky130_fd_sc_hd_dfxtip_1)	
		-0.02	-0.02	-0.02	library hold time
		-0.02	-0.02	-0.02	data required time
		-0.02	-0.02	-0.02	data arrival time
		0.22	0.22	0.22	slack (MET)

Startpoint: _10763_ (rising edge-triggered flip-flop clocked by clk)
 Endpoint: _10985_ (rising edge-triggered flip-flop clocked by clk)
 Path Group: clk
 Path Type: max

Fanout	Cap	Slew	Delay	Time	Description
4	0.01	0.00	0.00	0.00	clock clk (rise edge)
		0.00	0.00	0.00	clock network delay (ideal)
		0.00	0.00	0.00	^ _10763_/_CLK (sky130_fd_sc_hd_dfxtpl_1)
		0.14	0.67	0.67	v _10763_/_Q (sky130_fd_sc_hd_dfxtpl_1)
					cpu.csr_d_sel (net)
		0.14	0.00	0.67	v _05072_/_A (sky130_fd_sc_hd_nor3_2)
		0.47	0.52	1.19	^ _05072_/_Y (sky130_fd_sc_hd_nor3_2)
					02035 (net)
		0.47	0.01	1.20	^ _05073_/_A (sky130_fd_sc_hd_clkbuf_2)
		0.21	0.45	1.65	^ _05073_/_X (sky130_fd_sc_hd_clkbuf_2)
	0.02				_02036_ (net)
		0.21	0.01	1.66	^ _05077_/_A (sky130_fd_sc_hd_nand3_2)
		0.24	0.28	1.94	v _05077_/_Y (sky130_fd_sc_hd_nand3_2)
					02040 (net)
		0.24	0.00	1.94	v _05078_/_A (sky130_fd_sc_hd_clkbuf_2)
	0.02	0.17	0.35	2.29	v _05078_/_X (sky130_fd_sc_hd_clkbuf_2)
					02041 (net)
		0.17	0.01	2.30	v _06033_/_D1 (sky130_fd_sc_hd_o2111ai_1)
		0.31	0.17	2.47	^ _06033_/_Y (sky130_fd_sc_hd_o2111ai_1)
					02739 (net)
1	0.00	0.31	0.00	2.47	^ _06034_/_B1 (sky130_fd_sc_hd_a21oi_1)
		0.20	0.22	2.69	v _06034_/_Y (sky130_fd_sc_hd_a21oi_1)
					02740 (net)
1	0.01	0.20	0.01	2.70	v _06037_/_A2 (sky130_fd_sc_hd_o21bai_4)
		0.37	0.39	3.09	^ _06037_/_Y (sky130_fd_sc_hd_o21bai_4)
2	0.03				_02743_ (net)
		0.37	0.02	3.11	^ _06038_/_A (sky130_fd_sc_hd_dlymetal6s2s_1)
		0.28	0.43	3.54	^ _06038_/_X (sky130_fd_sc_hd_dlymetal6s2s_1)
					02744 (net)
5	0.02	0.28	0.00	3.54	^ _06039_/_A (sky130_fd_sc_hd_buf_2)

		0.25	0.41	3.95 ^ _06039_ /X (sky130_fd_sc_hd_buf_2)	
5	0.03			_02745_ (net)	
		0.25	0.01	3.97 ^ _09121_ /A (sky130_fd_sc_hd_and2_1)	
		0.18	0.38	4.34 ^ _09121_ /X (sky130_fd_sc_hd_and2_1)	
2	0.01			_00492_ (net)	
		0.18	0.00	4.35 ^ _09122_ /A (sky130_fd_sc_hd_inv_2)	
		0.04	0.08	4.43 v _09122_ /Y (sky130_fd_sc_hd_inv_2)	
1	0.00			_00481_ (net)	
		0.04	0.00	4.43 v _09265_ /A0 (sky130_fd_sc_hd_mux2_1)	
		0.14	0.66	5.08 v _09265_ /X (sky130_fd_sc_hd_mux2_1)	
2	0.01			_00488_ (net)	
		0.14	0.00	5.09 v _09136_ /A_N (sky130_fd_sc_hd_nand3b_1)	
		0.27	0.47	5.56 v _09136_ /Y (sky130_fd_sc_hd_nand3b_1)	
3	0.01			_04798_ (net)	
		0.27	0.00	5.56 v _09138_ /B1 (sky130_fd_sc_hd_o211a_1)	
		0.11	0.36	5.92 v _09138_ /X (sky130_fd_sc_hd_o211a_1)	
2	0.01			_00490_ (net)	
		0.11	0.00	5.93 v _09249_ /A0 (sky130_fd_sc_hd_mux2_1)	
		0.11	0.64	6.56 v _09249_ /X (sky130_fd_sc_hd_mux2_1)	
1	0.00			_00506_ (net)	
		0.11	0.00	6.56 v _09250_ /A1 (sky130_fd_sc_hd_mux2_1)	
		0.17	0.74	7.30 v _09250_ /X (sky130_fd_sc_hd_mux2_1)	
2	0.01			cpu.o_wdata0 (net)	
		0.17	0.00	7.31 v _09245_ /A1 (sky130_fd_sc_hd_mux2_2)	
		0.19	0.72	8.03 v _09245_ /X (sky130_fd_sc_hd_mux2_2)	
5	0.03			rf_ram.i_wdata[1] (net)	
		0.20	0.01	8.04 v _05185_ /A (sky130_fd_sc_hd_inv_2)	
		0.28	0.30	8.33 ^ _05185_ /Y (sky130_fd_sc_hd_inv_2)	
4	0.04			_02135_ (net)	
		0.29	0.01	8.34 ^ _05186_ /A (sky130_fd_sc_hd_buf_2)	
		0.26	0.43	8.77 ^ _05186_ /X (sky130_fd_sc_hd_buf_2)	
5	0.03			_02136_ (net)	
		0.26	0.00	8.78 ^ _05187_ /A (sky130_fd_sc_hd_dlymetal6s2s_1)	
		0.30	0.41	9.19 ^ _05187_ /X (sky130_fd_sc_hd_dlymetal6s2s_1)	
5	0.02			_02137_ (net)	
		0.30	0.01	9.19 ^ _05197_ /A (sky130_fd_sc_hd_clkbuf_4)	
		0.23	0.46	9.65 ^ _05197_ /X (sky130_fd_sc_hd_clkbuf_4)	
c	o or			_02145_ (net)	

				<u>o21ai_1</u> (net)
4	0.04	0.29	0.01	8.34 ^ _05186_/A (sky130_fd_sc_hd_buf_2)
		0.26	0.43	8.77 ^ _05186_/X (sky130_fd_sc_hd_buf_2)
5	0.03			_02136_ (net)
		0.26	0.00	8.78 ^ _05187_/A (sky130_fd_sc_hd_dlymetal6s2s_1)
		0.30	0.41	9.19 ^ _05187_/X (sky130_fd_sc_hd_dlymetal6s2s_1)
5	0.02			_02137_ (net)
		0.30	0.01	9.19 ^ _05197_/A (sky130_fd_sc_hd_clkbuf_4)
		0.23	0.46	9.65 ^ _05197_/X (sky130_fd_sc_hd_clkbuf_4)
6	0.05			_02145_ (net)
		0.23	0.02	9.67 ^ _05263_/A (sky130_fd_sc_hd_dlymetal6s2s_1)
		0.30	0.40	10.07 ^ _05263_/X (sky130_fd_sc_hd_dlymetal6s2s_1)
5	0.02			_02201_ (net)
		0.30	0.00	10.08 ^ _05284_/A1 (sky130_fd_sc_hd_o21ai_1)
		0.10	0.22	10.30 v _05284_/Y (sky130_fd_sc_hd_o21ai_1)
1	0.00			_01792_ (net)
		0.10	0.00	10.30 v _10985_/D (sky130_fd_sc_hd_dfxtpl_1)
				10.30 data arrival time
		0.00	20.00	clock clk (rise edge)
		0.00	20.00	clock network delay (ideal)
		0.00	20.00	clock reconvergence pessimism
			20.00 ^ _10985_/CLK (sky130_fd_sc_hd_dfxtpl_1)	
		-0.29	19.71	library setup time
			19.71	data required time
			19.71	data required time
			-10.30	data arrival time
			9.42	slack (MET)

```

min_max_report_end
check_report
No paths found.
check_report_end
wns 0.00
tns 0.00

```

Maximum Operating Frequency Calculations

The SERV CPU core requires a reasonably precise **50MHZ** on iCE40. This is our clock's frequency approximately for the mentioned boards in previous statement.

Here we are taking 50MHZ as our operating frequency based on SERV specifications

$$\text{Time period (T)} = \frac{1}{\text{Frequency (F)}}$$

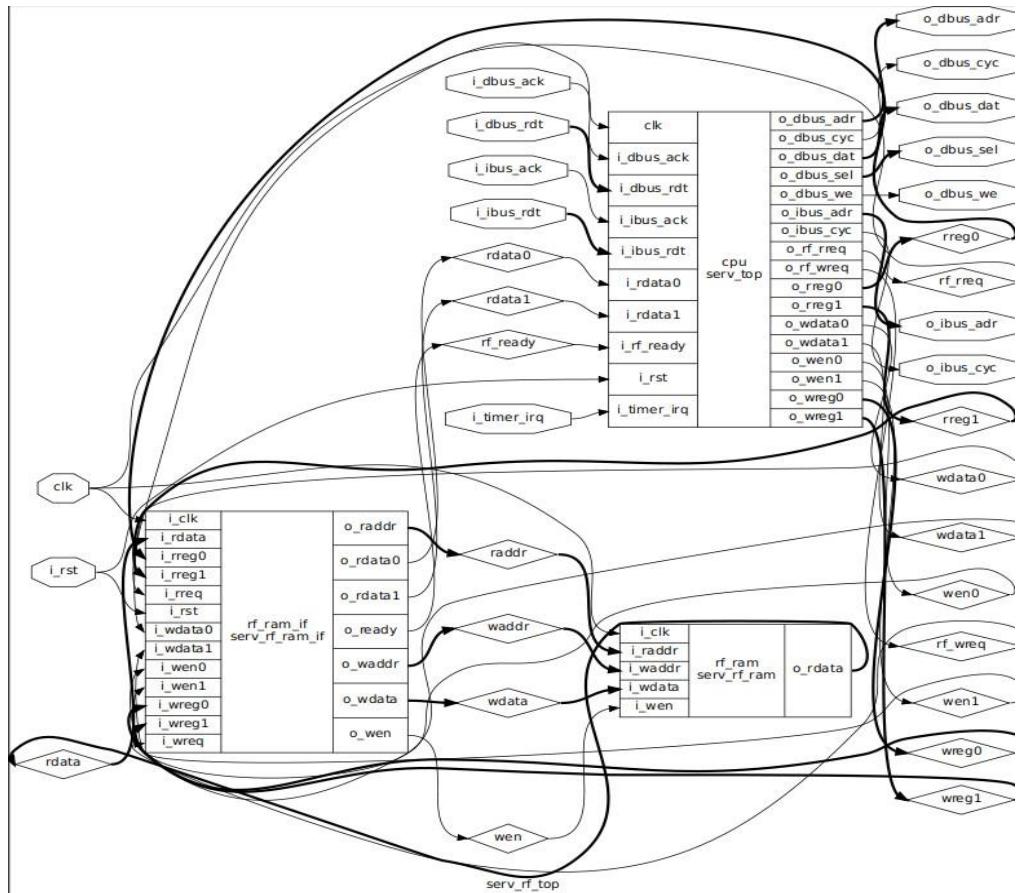
$$= \frac{1}{50 \text{ MHZ}}$$

$$= 20 \text{ ns}$$

Sky130 PDK's Operating Frequency

Description	Active Mode	Deep Sleep Mode
Output: Fmax Freq	33MHz	33MHz
Output: Fmax Freq	33MHz	33MHz
SE Input: Fmax Freq	66MHz	66MHz

Schematic of Synthesized Netlist



Synthesized Netlist File

This netlist file shows the inputs, outputs, wires and gates from sky130_fd_sc_hd libraries with different flavours according to user's designs.

Post Synthesis

```
/* Generated by Yosys 0.9+3621 (git sha1 84e9fa7, gcc 8.3.1 -fPIC -Os) */
2 module serv_rf_top(clk, i_rst, i_timer_irq, o_ibus_adr, o_ibus_cyc, i_ibus_rdt, i_ibus_ack, o_dbus_adr, o_dbus_dat, o_dbus_sel, o_dbus_we, o_dbus
3   wire _0000;
4   wire _0001;
5   wire _0002;
6   wire _0003;
7   wire _0004;
8   wire _0005;
9   wire _0006;
10  wire _0007;
11  wire _0008;
12  wire _0009;
13  wire _0010;
14  wire _0011;
15  wire _0012;
16  wire _0013;
17  wire _0014;
18  wire _0015;
19  wire _0016;
20  wire _0017;
21  wire _0018;
22  wire _0019;
23  wire _0020;
24  wire _0021;
25  wire _0022;
26  wire _0023;
27  wire _0024;
28  wire _0025;
29  wire _0026;
30  wire _0027;
31  wire _0028;
32  wire _0029;
33  wire _0030;
34  wire _0031;
35  wire _0032;
36  wire _0033;
37  wire _0034;
38  wire _0035;
39  wire _0036;
40  wire _0037;
41  wire _0038;
42  wire _0039;
43  wire _0040;
44  wire _0041;
45  wire _0042;
46  wire _0043;
47  wire _0044;
48  wire _0045;
```

```
47  wire _0043_;  
48  wire _0044_;  
49  wire _0045_;  
50  wire _0046_;  
51  wire _0047_;  
52  wire _0048_;  
53  wire _0049_;  
54  wire _0050_;  
55  wire _0051_;  
56  wire _0052_;  
57  wire _0053_;  
58  wire _0054_;  
59  wire _0055_;  
60  wire _0056_;  
61  wire _0057_;  
62  wire _0058_;  
63  wire _0059_;  
64  wire _0060_;  
65  wire _0061_;  
66  wire _0062_;  
67  wire _0063_;  
68  wire _0064_;  
69  wire _0065_;  
70  wire _0066_;  
71  wire _0067_;  
72  wire _0068_;  
73  wire _0069_;  
74  wire _0070_;  
75  wire _0071_;  
76  wire _0072_;  
77  wire _0073_;  
78  wire _0074_;  
79  wire _0075_;  
80  wire _0076_;  
81  wire _0077_;  
82  wire _0078_;  
83  wire _0079_;  
84  wire _0080_;  
85  wire _0081_;  
86  wire _0082_;  
87  wire _0083_;  
88  wire _0084_;  
89  wire _0085_;  
90  wire _0086_;  
91  wire _0087_;  
92  wire _0088_;  
93  wire _0089_;  
94  wire _0090_;  
95  wire _0091_;
```

```
93   wire _0089_ ;
94   wire _0090_ ;
95   wire _0091_ ;
96   wire _0092_ ;
97   wire _0093_ ;
98   wire _0094_ ;
99   wire _0095_ ;
100  wire _0096_ ;
101  wire _0097_ ;
102  wire _0098_ ;
103  wire _0099_ ;
104  wire _0100_ ;
105  wire _0101_ ;
106  wire _0102_ ;
107  wire _0103_ ;
108  wire _0104_ ;
109  wire _0105_ ;
110  wire _0106_ ;
111  wire _0107_ ;
112  wire _0108_ ;
113  wire _0109_ ;
114  wire _0110_ ;
115  wire _0111_ ;
116  wire _0112_ ;
117  wire _0113_ ;
118  wire _0114_ ;
119  wire _0115_ ;
120  wire _0116_ ;
121  wire _0117_ ;
122  wire _0118_ ;
123  wire _0119_ ;
124  wire _0120_ ;
125  wire _0121_ ;
126  wire _0122_ ;
127  wire _0123_ ;
128  wire _0124_ ;
129  wire _0125_ ;
130  wire _0126_ ;
131  wire _0127_ ;
132  wire _0128_ ;
133  wire _0129_ ;
134  wire _0130_ ;
135  wire _0131_ ;
136  wire _0132_ ;
137  wire _0133_ ;
138  wire _0134_ ;
139  wire _0135_ ;
140  wire _0136_ ;
141  wire _0137 ;
```

```
3364    wire \cpu.alu.i_en,
3365    wire \cpu.alu.add_cy_r ;
3366    wire \cpu.alu.cmp_r ;
3367    wire \cpu.alu.i_en ;
3368    wire \cpu.alu.i_rs1 ;
3369    wire \cpu.bne_or_bge ;
3370    wire \cpu.bufreg.c_r ;
3371    wire \cpu.bufreg.i_en ;
3372    wire \cpu.bufreg.i_sh_signed ;
3373    wire \cpu.bufreg.o_lsb[0] ;
3374    wire \cpu.bufreg.o_lsb[1] ;
3375    wire \cpu.cnt_done ;
3376    wire \cpu.csr.csr_in ;
3377    wire \cpu.csr.i_csr_d_sel ;
3378    wire \cpu.csr.i_csr_imm ;
3379    wire \cpu.csr.i_ebreak ;
3380    wire \cpu.csr.mcause31 ;
3381    wire \cpu.csr.mcause3_0[0] ;
3382    wire \cpu.csr.mcause3_0[1] ;
3383    wire \cpu.csr.mcause3_0[2] ;
3384    wire \cpu.csr.mcause3_0[3] ;
3385    wire \cpu.csr.mie_mtie ;
3386    wire \cpu.csr.mstatus_mie ;
3387    wire \cpu.csr.mstatus_mpie ;
3388    wire \cpu.csr.o_new_irq ;
3389    wire \cpu.csr.timer_irq_r ;
3390    wire \cpu.ctrl.i_jump ;
3391    wire \cpu.ctrl.new_pc ;
3392    wire \cpu.ctrl.pc_plus_4_cy_r ;
3393    wire \cpu.ctrl.pc_plus_offset_cy_r ;
3394    wire \cpu.decode.co_immdec_ctrl[3] ;
3395    wire \cpu.decode.co_mem_word ;
3396    wire \cpu.decode.op21 ;
3397    wire \cpu.decode.op22 ;
3398    wire \cpu.decode.op26 ;
3399    wire \cpu.decode.opcode[0] ;
3400    wire \cpu.decode.opcode[1] ;
3401    wire \cpu.decode.opcode[2] ;
3402    wire \cpu.immdec.imm11_7[0] ;
3403    wire \cpu.immdec.imm11_7[1] ;
3404    wire \cpu.immdec.imm11_7[2] ;
3405    wire \cpu.immdec.imm11_7[3] ;
3406    wire \cpu.immdec.imm11_7[4] ;
3407    wire \cpu.immdec.imm19_12_20[0] ;
3408    wire \cpu.immdec.imm19_12_20[1] ;
3409    wire \cpu.immdec.imm19_12_20[2] ;
3410    wire \cpu.immdec.imm19_12_20[3] ;
3411    wire \cpu.immdec.imm19_12_20[5] ;
3412    wire \cpu.immdec.imm19_12_20[6] ;
```

```
3410    wire  \cpu.immdec.imm12_12_20[5] ;
3411    wire  \cpu.immdec.imm19_12_20[5] ;
3412    wire  \cpu.immdec.imm19_12_20[6] ;
3413    wire  \cpu.immdec.imm19_12_20[7] ;
3414    wire  \cpu.immdec.imm19_12_20[8] ;
3415    wire  \cpu.immdec.imm24_20[0] ;
3416    wire  \cpu.immdec.imm24_20[1] ;
3417    wire  \cpu.immdec.imm24_20[2] ;
3418    wire  \cpu.immdec.imm24_20[3] ;
3419    wire  \cpu.immdec.imm24_20[4] ;
3420    wire  \cpu.immdec.imm30_25[0] ;
3421    wire  \cpu.immdec.imm30_25[1] ;
3422    wire  \cpu.immdec.imm30_25[2] ;
3423    wire  \cpu.immdec.imm30_25[3] ;
3424    wire  \cpu.immdec.imm30_25[4] ;
3425    wire  \cpu.immdec.imm30_25[5] ;
3426    wire  \cpu.immdec.imm7 ;
3427    wire  \cpu.immdec.signbit ;
3428    wire  \cpu.mem_bytect[0] ;
3429    wire  \cpu.mem_bytect[1] ;
3430    wire  \cpu.mem_if.signbit ;
3431    wire  \cpu.o_wdata0 ;
3432    wire  \cpu.o_wdata1 ;
3433    wire  \cpu.o_wen0 ;
3434    wire  \cpu.o_wen1 ;
3435    wire  \cpu.state.ibus_cyc ;
3436    wire  \cpu.state.init_done ;
3437    wire  \cpu.state.misalign_trap_sync ;
3438    wire  \cpu.state.o_cnt[2] ;
3439    wire  \cpu.state.o_cnt_r[0] ;
3440    wire  \cpu.state.o_cnt_r[1] ;
3441    wire  \cpu.state.o_cnt_r[2] ;
3442    wire  \cpu.state.o_cnt_r[3] ;
3443    wire  \cpu.state.stage_two_req ;
3444    input i_dbus_ack;
3445    input [31:0] i_dbus_rdt;
3446    input i_ibus_ack;
3447    input [31:0] i_ibus_rdt;
3448    input i_RST;
3449    input i_timer_irq;
3450    output [31:0] o_dbus_adr;
3451    output o_dbus_cyc;
3452    output [31:0] o_dbus_dat;
3453    output [3:0] o_dbus_sel;
3454    output o_dbus_we;
3455    output [31:0] o_ibus_adr;
3456    output o_ibus_cyc;
3457    wire  \rf_ram.i_raddr[0] ;
3458    wire  \rf_ram.i_raddr[1] ;
```

```
3437  wire \cpu.state.misalign_trap_sync ;
3438  wire \cpu.state.o_cnt[2] ;
3439  wire \cpu.state.o_cnt_r[0] ;
3440  wire \cpu.state.o_cnt_r[1] ;
3441  wire \cpu.state.o_cnt_r[2] ;
3442  wire \cpu.state.o_cnt_r[3] ;
3443  wire \cpu.state.stage_two_req ;
3444  input i_dbus_ack;
3445  input [31:0] i_dbus_rdt;
3446  input i_ibus_ack;
3447  input [31:0] i_ibus_rdt;
3448  input i_RST;
3449  input i_timer_irq;
3450  output [31:0] o_dbus_adr;
3451  output o_dbus_cyc;
3452  output [31:0] o_dbus_dat;
3453  output [3:0] o_dbus_sel;
3454  output o_dbus_we;
3455  output [31:0] o_ibus_adr;
3456  output o_ibus_cyc;
3457  wire \rf_ram.i_raddr[0] ;
3458  wire \rf_ram.i_raddr[1] ;
3459  wire \rf_ram.i_raddr[2] ;
3460  wire \rf_ram.i_raddr[3] ;
3461  wire \rf_ram.i_wdata[0] ;
3462  wire \rf_ram.i_wdata[1] ;
3463  wire \rf_ram.memory[0][0] ;
3464  wire \rf_ram.memory[0][1] ;
3465  wire \rf_ram.memory[100][0] ;
3466  wire \rf_ram.memory[100][1] ;
3467  wire \rf_ram.memory[101][0] ;
3468  wire \rf_ram.memory[101][1] ;
3469  wire \rf_ram.memory[102][0] ;
3470  wire \rf_ram.memory[102][1] ;
3471  wire \rf_ram.memory[103][0] ;
3472  wire \rf_ram.memory[103][1] ;
3473  wire \rf_ram.memory[104][0] ;
3474  wire \rf_ram.memory[104][1] ;
3475  wire \rf_ram.memory[105][0] ;
3476  wire \rf_ram.memory[105][1] ;
3477  wire \rf_ram.memory[106][0] ;
3478  wire \rf_ram.memory[106][1] ;
3479  wire \rf_ram.memory[107][0] ;
3480  wire \rf_ram.memory[107][1] ;
3481  wire \rf_ram.memory[108][0] ;
3482  wire \rf_ram.memory[108][1] ;
3483  wire \rf_ram.memory[109][0] ;
3484  wire \rf_ram.memory[109][1] ;
3485  wire \rf_ram.memory[10][0] .
```

```
4628     sky130_fd_sc_hd__buf_1 _3360_ (
4629         .A(i_rst),
4630         .X(_1872_)
4631     );
4632     sky130_fd_sc_hd__inv_2 _3361_ (
4633         .A(\cpu.cnt_done ),
4634         .Y(_1873_)
4635     );
4636     sky130_fd_sc_hd__buf_1 _3362_ (
4637         .A(_1873_),
4638         .X(_1874_)
4639     );
4640     sky130_fd_sc_hd__inv_2 _3363_ (
4641         .A(\cpu.decode.opcode[2] ),
4642         .Y(_1875_)
4643     );
4644     sky130_fd_sc_hd__buf_1 _3364_ (
4645         .A(_1875_),
4646         .X(_1876_)
4647     );
4648     sky130_fd_sc_hd__or3_2 _3365_ (
4649         .A(\cpu.decode.co_immdec_ctrl[3] ),
4650         .B(_1876_),
4651         .C(\cpu.decode.opcode[0] ),
4652         .X(_1877_)
4653     );
4654     sky130_fd_sc_hd__inv_2 _3366_ (
4655         .A(\cpu.bne_or_bge ),
4656         .Y(_1878_)
4657     );
4658     sky130_fd_sc_hd__or2_2 _3367_ (
4659         .A(\cpu.decode.co_mem_word ),
4660         .B(_1878_),
4661         .X(_0071_)
4662     );
4663     sky130_fd_sc_hd__or2_2 _3368_ (
4664         .A(_1877_),
4665         .B(_0071_),
4666         .X(_1879_)
4667     );
4668     sky130_fd_sc_hd__or3_2 _3369_ (
4669         .A(\cpu.decode.co_immdec_ctrl[3] ),
```

```
4655     .A(\cpu.bne_or_bge),
4656     .Y(_1878_)
4657 );
4658 sky130_fd_sc_hd_or2_2_3367_ (
4659     .A(\cpu.decode.co_mem_word),
4660     .B(_1878_),
4661     .X(_0071_)
4662 );
4663 sky130_fd_sc_hd_or2_2_3368_ (
4664     .A(_1877_),
4665     .B(_0071_),
4666     .X(_1879_)
4667 );
4668 sky130_fd_sc_hd_or3_2_3369_ (
4669     .A(\cpu.decode.co_immdec_ctrl[3]),
4670     .B(\cpu.decode.opcode[2]),
4671     .C(\cpu.decode.opcode[0]),
4672     .X(_1880_)
4673 );
4674 sky130_fd_sc_hd_inv2_3370_ (
4675     .A(\cpu.decode.co_mem_word),
4676     .Y(_1881_)
4677 );
4678 sky130_fd_sc_hd_inv2_3371_ (
4679     .A(\cpu.decode.co_immdec_ctrl[3]),
4680     .Y(_1882_)
4681 );
4682 sky130_fd_sc_hd_o32a2_3372_ (
4683     .A1(\cpu.csr.i_csr_d_sel),
4684     .A2(_1881_),
4685     .A3(_1877_),
4686     .B1(_1882_),
4687     .B2(\cpu.decode.opcode[2]),
4688     .X(_1883_)
4689 );
4690 sky130_fd_sc_hd_a311o2_3373_ (
4691     .A1(_1879_),
4692     .A2(_1880_),
4693     .A3(_1883_),
4694     .B1(\cpu.state.init_done),
4695     .C1(\cpu.csr.o_new_irq),
4696     .X(_1884_)
4697 );
4698 sky130_fd_sc_hd_buf1_3374_ (
4699     .A(_1884_),
4700     .X(_0074_)
4701 );
4702 sky130_fd_sc_hd_or2_2_3375_ (
```

```
4800 sky130_fd_sc_hd__inv_2 _3394_ (
4801     .A(_1884_),
4802     .Y(_1902_)
4803 );
4804 sky130_fd_sc_hd__or4_2 _3395_ (
4805     .A(\cpu.state.o_cnt_r[1]),
4806     .B(\cpu.state.o_cnt_r[0]),
4807     .C(\cpu.state.o_cnt_r[2]),
4808     .D(\cpu.state.o_cnt_r[3]),
4809     .X(_1903_)
4810 );
4811 sky130_fd_sc_hd__inv_2 _3396_ (
4812     .A(_1903_),
4813     .Y(_1904_)
4814 );
4815 sky130_fd_sc_hd__buf_1 _3397_ (
4816     .A(_1904_),
4817     .X(_1905_)
4818 );
4819 sky130_fd_sc_hd__or2_2 _3398_ (
4820     .A(_1902_),
4821     .B(_1905_),
4822     .X(_1906_)
4823 );
4824 sky130_fd_sc_hd__inv_2 _3399_ (
4825     .A(_1906_),
4826     .Y(_1907_)
4827 );
4828 sky130_fd_sc_hd__buf_1 _3400_ (
4829     .A(_1907_),
4830     .X(_1908_)
4831 );
4832 sky130_fd_sc_hd__buf_1 _3401_ (
4833     .A(_1908_),
4834     .X(_1909_)
4835 );
4836 sky130_fd_sc_hd__buf_1 _3402_ (
4837     .A(_1906_),
4838     .X(_1910_)
4839 );
4840 sky130_fd_sc_hd__buf_1 _3403_ (
4841     .A(_1910_),
4842     .X(_1911_)
4843 );
4844 sky130_fd_sc_hd__inv_2 _3404_ (
4845     .A(i_RST),
4846     .Y(_1912_)
4847 );
```

```
30728     sky130_fd_sc_hd__dfxtp_2 _8063_ (
30729         .CLK(clk),
30730         .D(_1862_),
30731         .Q(o_ibus_adr[25])
30732     );
30733     sky130_fd_sc_hd__dfxtp_2 _8064_ (
30734         .CLK(clk),
30735         .D(_1863_),
30736         .Q(o_ibus_adr[26])
30737     );
30738     sky130_fd_sc_hd__dfxtp_2 _8065_ (
30739         .CLK(clk),
30740         .D(_1864_),
30741         .Q(o_ibus_adr[27])
30742     );
30743     sky130_fd_sc_hd__dfxtp_2 _8066_ (
30744         .CLK(clk),
30745         .D(_1865_),
30746         .Q(o_ibus_adr[28])
30747     );
30748     sky130_fd_sc_hd__dfxtp_2 _8067_ (
30749         .CLK(clk),
30750         .D(_1866_),
30751         .Q(o_ibus_adr[29])
30752     );
30753     sky130_fd_sc_hd__dfxtp_2 _8068_ (
30754         .CLK(clk),
30755         .D(_1867_),
30756         .Q(o_ibus_adr[30])
30757     );
30758     sky130_fd_sc_hd__dfxtp_2 _8069_ (
30759         .CLK(clk),
30760         .D(_1868_),
30761         .Q(o_ibus_adr[31])
30762     );
30763     sky130_fd_sc_hd__dfxtp_2 _8070_ (
30764         .CLK(clk),
30765         .D(_1869_),
30766         .Q(\cpu.state.misalign_trap_sync )
30767     );
30768     sky130_fd_sc_hd__dfxtp_2 _8071_ (
30769         .CLK(clk),
30770         .D(_1870_),
30771         .Q(\cpu.ctrl.i_jump )
30772     );
30773     sky130_fd_sc_hd__dfxtp_2 _8072_ (
30774         .CLK(clk),
30775         .D(_1871_),
30776         .Q(\cpu.state.init_done )
30777     );
30778 endmodule
```

Post CTS

```
module serv_rf_top (VGND,
    VPWR,
    clk,
    i_dbus_ack,
    i_ibus_ack,
    i_RST,
    i_timer_irq,
    o_dbus_cyc,
    o_dbus_we,
    o_ibus_cyc,
    i_dbus_rdt,
    i_ibus_rdt,
    o_dbus_adr,
    o_dbus_dat,
    o_dbus_sel,
    o_ibus_adr);
    input VGND;
    input VPWR;
    input clk;
    input i_dbus_ack;
    input i_ibus_ack;
    input i_RST;
    input i_timer_irq;
    output o_dbus_cyc;
    output o_dbus_we;
    output o_ibus_cyc;
    input [31:0] i_dbus_rdt;
    input [31:0] i_ibus_rdt;
    output [31:0] o_dbus_adr;
    output [31:0] o_dbus_dat;
    output [3:0] o_dbus_sel;
    output [31:0] o_ibus_adr;

    wire _00000_;
    wire _00001_;
    wire _00002_;
    wire _00003_;
```

```
wire _00001_;  
wire _00002_;  
wire _00003_;  
wire _00004_;  
wire _00005_;  
wire _00006_;  
wire _00007_;  
wire _00008_;  
wire _00009_;  
wire _00010_;  
wire _00011_;  
wire _00012_;  
wire _00013_;  
wire _00014_;  
wire _00015_;  
wire _00016_;  
wire _00017_;  
wire _00018_;  
wire _00019_;  
wire _00020_;  
wire _00021_;  
wire _00022_;  
wire _00023_;  
wire _00024_;  
wire _00025_;  
wire _00026_;  
wire _00027_;  
wire _00028_;  
wire _00029_;  
wire _00030_;  
wire _00031_;  
wire _00032_;  
wire _00033_;  
wire _00034_;  
wire _00035_;  
wire _00036_;  
wire _00037_;
```

```
wire clknet_leaf_90_clk;
wire clknet_leaf_91_clk;
wire clknet_leaf_92_clk;
wire clknet_leaf_93_clk;
wire clknet_leaf_94_clk;
wire clknet_leaf_95_clk;
wire clknet_leaf_96_clk;
wire clknet_leaf_97_clk;
wire clknet_leaf_98_clk;
wire clknet_leaf_99_clk;
wire clknet_leaf_9_clk;
wire \cpu.alu.add_cy_r ;
wire \cpu.alu.cmp_r ;
wire \cpu.alu.i_en ;
wire \cpu.alu.i_rs1 ;
wire \cpu.bne_or_bge ;
wire \cpu.bufreg.c_r ;
wire \cpu.bufreg.i_en ;
wire \cpu.bufreg.i_sh_signed ;
wire \cpu.bufreg.o_lsb[0] ;
wire \cpu.bufreg.o_lsb[1] ;
wire \cpu.cnt_done ;
wire \cpu.csr_d_sel ;
wire \cpu.csr_imm ;
wire \cpu.csr_in ;
wire \cpu.ctrl.i_jump ;
wire \cpu.ctrl.new_pc ;
wire \cpu.ctrl.pc_plus_4_cy_r ;
wire \cpu.ctrl.pc_plus_offset_cy_r ;
wire \cpu.decode.co_ebreak ;
wire \cpu.decode.co_immdec_ctrl[3] ;
wire \cpu.decode.co_mem_word ;
wire \cpu.decode.op21 ;
wire \cpu.decode.op22 ;
wire \cpu.decode.op26 ;
wire \cpu.decode.opcode[0] ;
wire \cpu.decode.opcode[1] ;
```

```
.....,
sky130_fd_sc_hd_buf_12 repeater193 (.A(net194),
    .X(net193),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd_buf_12 repeater194 (.A(\rf_ram.i_r
    .X(net194),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd_clkbuf_8 repeater195 (.A(net196),
    .X(net195),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd_buf_6 repeater196 (.A(net34),
    .X(net196),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd_buf_6 repeater197 (.A(net1),
    .X(net197),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd_clkbuf_8 repeater198 (.A(net1),
    .X(net198),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
endmodule
```

Post Optimization

```
module serv_rf_top (VGND,
    VPWR,
    clk,
    i_dbus_ack,
    i_ibus_ack,
    i_RST,
    i_timer_irq,
    o_dbus_cyc,
    o_dbus_we,
    o_ibus_cyc,
    i_dbus_rdt,
    i_ibus_rdt,
    o_dbus_addr,
    o_dbus_dat,
    o_dbus_sel,
    o_ibus_addr);
    input VGND;
    input VPWR;
    input clk;
    input i_dbus_ack;
    input i_ibus_ack;
    input i_RST;
    input i_timer_irq;
    output o_dbus_cyc;
    output o_dbus_we;
    output o_ibus_cyc;
    input [31:0] i_dbus_rdt;
    input [31:0] i_ibus_rdt;
    output [31:0] o_dbus_addr;
    output [31:0] o_dbus_dat;
    output [3:0] o_dbus_sel;
    output [31:0] o_ibus_addr;

    wire _00000_;
    wire _00001_;
    wire _00002_;
    wire _00003_;
```

```
wire _00001_;  
wire _00002_;  
wire _00003_;  
wire _00004_;  
wire _00005_;  
wire _00006_;  
wire _00007_;  
wire _00008_;  
wire _00009_;  
wire _00010_;  
wire _00011_;  
wire _00012_;  
wire _00013_;  
wire _00014_;  
wire _00015_;  
wire _00016_;  
wire _00017_;  
wire _00018_;  
wire _00019_;  
wire _00020_;  
wire _00021_;  
wire _00022_;  
wire _00023_;  
wire _00024_;  
wire _00025_;  
wire _00026_;  
wire _00027_;  
wire _00028_;  
wire _00029_;  
wire _00030_;  
wire _00031_;  
wire _00032_;  
wire _00033_;  
wire _00034_;  
wire _00035_;  
wire _00036_;  
wire _00037_;
```

```
wire \rf_ram.memory[366][1] ;
wire \rf_ram.memory[367][0] ;
wire \rf_ram.memory[367][1] ;
wire \rf_ram.memory[368][0] ;
wire \rf_ram.memory[368][1] ;
wire \rf_ram.memory[369][0] ;
wire \rf_ram.memory[369][1] ;
wire \rf_ram.memory[36][0] ;
wire \rf_ram.memory[36][1] ;
wire \rf_ram.memory[370][0] ;
wire \rf_ram.memory[370][1] ;
wire \rf_ram.memory[371][0] ;
wire \rf_ram.memory[371][1] ;
wire \rf_ram.memory[372][0] ;
wire \rf_ram.memory[372][1] ;
wire \rf_ram.memory[373][0] ;
wire \rf_ram.memory[373][1] ;
wire \rf_ram.memory[374][0] ;
wire \rf_ram.memory[374][1] ;
wire \rf_ram.memory[375][0] ;
wire \rf_ram.memory[375][1] ;
wire \rf_ram.memory[376][0] ;
wire \rf_ram.memory[376][1] ;
wire \rf_ram.memory[377][0] ;
wire \rf_ram.memory[377][1] ;
wire \rf_ram.memory[378][0] ;
wire \rf_ram.memory[378][1] ;
wire \rf_ram.memory[379][0] ;
wire \rf_ram.memory[379][1] ;
wire \rf_ram.memory[37][0] ;
wire \rf_ram.memory[37][1] ;
wire \rf_ram.memory[380][0] ;
wire \rf_ram.memory[380][1] ;
wire \rf_ram.memory[381][0] ;
wire \rf_ram.memory[381][1] ;
wire \rf_ram.memory[382][0] ;
wire \rf_ram.memory[382][1] ;
```

```
sky130_ta_sc_nd__decap_3 PHY_109 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_11 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_110 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_111 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_112 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_113 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_114 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_115 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_116 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 PHY_117 (.VGND(VGND),
    .VNR(VGND))
```

```

sky130_fd_sc_hd__buf_12 repeater193 (.A(net194),
  .X(net193),
  .VGND(VGND),
  .VNB(VGND),
  .VPB(VPWR),
  .VPWR(VPWR));
sky130_fd_sc_hd__buf_12 repeater194 (.A(\rf_ram.i_raddr[1] ),
  .X(net194),
  .VGND(VGND),
  .VNB(VGND),
  .VPB(VPWR),
  .VPWR(VPWR));
sky130_fd_sc_hd__clkbuf_8 repeater195 (.A(net196),
  .X(net195),
  .VGND(VGND),
  .VNB(VGND),
  .VPB(VPWR),
  .VPWR(VPWR));
sky130_fd_sc_hd__buf_6 repeater196 (.A(net34),
  .X(net196),
  .VGND(VGND),
  .VNB(VGND),
  .VPB(VPWR),
  .VPWR(VPWR));
sky130_fd_sc_hd__buf_6 repeater197 (.A(net1),
  .X(net197),
  .VGND(VGND),
  .VNB(VGND),
  .VPB(VPWR),
  .VPWR(VPWR));
sky130_fd_sc_hd__clkbuf_8 repeater198 (.A(net1),
  .X(net198),
  .VGND(VGND),
  .VNB(VGND),
  .VPB(VPWR),
  .VPWR(VPWR));
endmodule

```

Post Initial Routing

```
module serv_rf_top (VGND,
    VPWR,
    clk,
    i_dbus_ack,
    i_ibus_ack,
    i_RST,
    i_timer_irq,
    o_dbus_cyc,
    o_dbus_we,
    o_ibus_cyc,
    i_dbus_rdt,
    i_ibus_rdt,
    o_dbus_addr,
    o_dbus_dat,
    o_dbus_sel,
    o_ibus_addr);
    input VGND;
    input VPWR;
    input clk;
    input i_dbus_ack;
    input i_ibus_ack;
    input i_RST;
    input i_timer_irq;
    output o_dbus_cyc;
    output o_dbus_we;
    output o_ibus_cyc;
    input [31:0] i_dbus_rdt;
    input [31:0] i_ibus_rdt;
    output [31:0] o_dbus_addr;
    output [31:0] o_dbus_dat;
    output [3:0] o_dbus_sel;
    output [31:0] o_ibus_addr;

    wire _00000_;
    wire _00001_;
```

```
) .VNB(VGND),
L .VPB(VPWR),
? .VPWR(VPWR));
} sky130_fd_sc_hd__decap_12 FILLER_101_11 (.VGND(VGND),
+ .VNB(VGND),
; .VPB(VPWR),
; .VPWR(VPWR));
' sky130_fd_sc_hd__decap_4 FILLER_101_119 (.VGND(VGND),
) .VNB(VGND),
) .VPB(VPWR),
) .VPWR(VPWR));
L sky130_fd_sc_hd__decap_12 FILLER_101_126 (.VGND(VGND),
? .VNB(VGND),
; .VPB(VPWR),
+ .VPWR(VPWR));
; sky130_fd_sc_hd__decap_6 FILLER_101_138 (.VGND(VGND),
; .VNB(VGND),
' .VPB(VPWR),
) .VPWR(VPWR));
) sky130_fd_sc_hd__decap_8 FILLER_101_160 (.VGND(VGND),
) .VNB(VGND),
L .VPB(VPWR),
? .VPWR(VPWR));
; sky130_fd_sc_hd__fill_1 FILLER_101_169 (.VGND(VGND),
; .VNB(VGND),
; .VPB(VPWR),
; .VPWR(VPWR));
' sky130_fd_sc_hd__decap_8 FILLER_101_174 (.VGND(VGND),
; .VNB(VGND),
; .VPB(VPWR),
) .VPWR(VPWR));
L sky130_fd_sc_hd__decap_3 FILLER_101_182 (.VGND(VGND),
; .VNB(VGND),
; .VPB(VPWR),
; .VPWR(VPWR));
```

```
sky130_fd_sc_hd__fill_1 FILLER_120_819 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_3 FILLER_120_85 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_8 FILLER_121_102 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_12 FILLER_121_11 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__fill_2 FILLER_121_110 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_6 FILLER_121_113 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_12 FILLER_121_122 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__decap_8 FILLER_121_134 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR));
sky130_fd_sc_hd__fill_1 FILLER_121_142 (.VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR))
```

```

.VGND(VGND),
.VNB(VGND),
.VPB(VPWR),
.VPWR(VPWR));
sky130_fd_sc_hd_buf_12 repeater194 (.A(\rf_ram.i_raddr[1] ),
.X(net194),
.VGND(VGND),
.VNB(VGND),
.VPB(VPWR),
.VPWR(VPWR));
sky130_fd_sc_hd_clkbuf_8 repeater195 (.A(net196),
.X(net195),
.VGND(VGND),
.VNB(VGND),
.VPB(VPWR),
.VPWR(VPWR));
sky130_fd_sc_hd_buf_6 repeater196 (.A(net34),
.X(net196),
.VGND(VGND),
.VNB(VGND),
.VPB(VPWR),
.VPWR(VPWR));
sky130_fd_sc_hd_buf_6 repeater197 (.A(net1),
.X(net197),
.VGND(VGND),
.VNB(VGND),
.VPB(VPWR),
.VPWR(VPWR));
sky130_fd_sc_hd_clkbuf_8 repeater198 (.A(net1),
.X(net198),
.VGND(VGND),
.VNB(VGND),
.VPB(VPWR),
.VPWR(VPWR));
endmodule

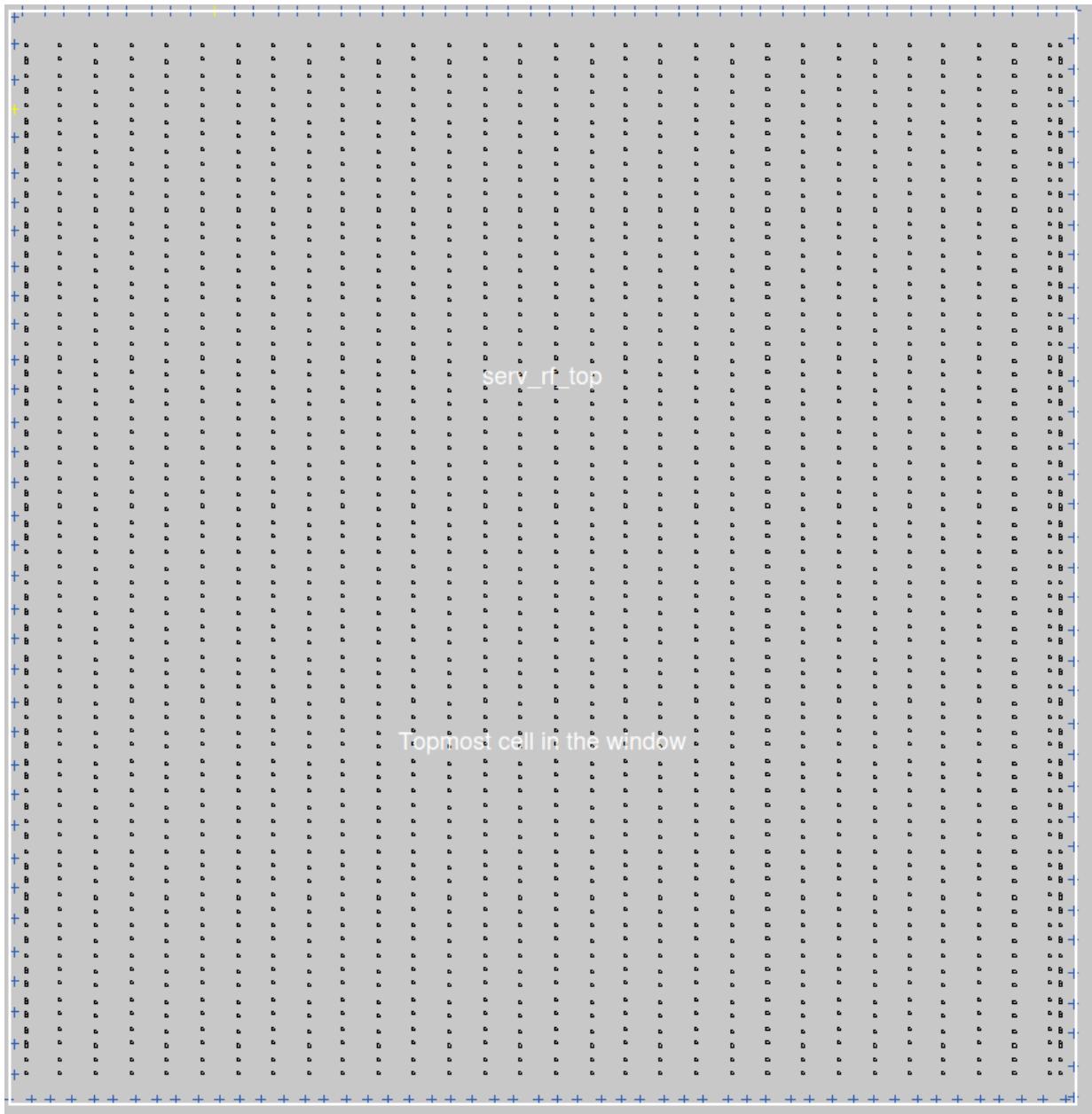
```

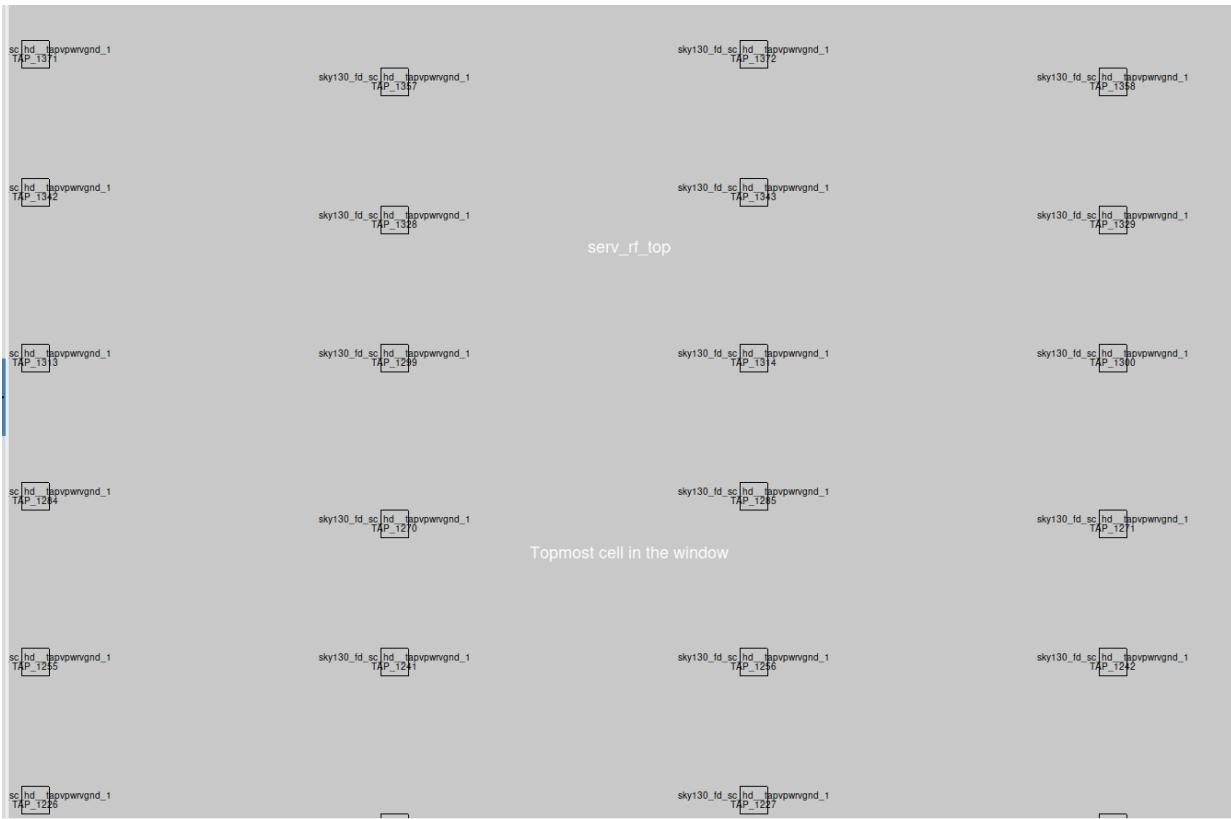
Layout Generation by Magic Tool

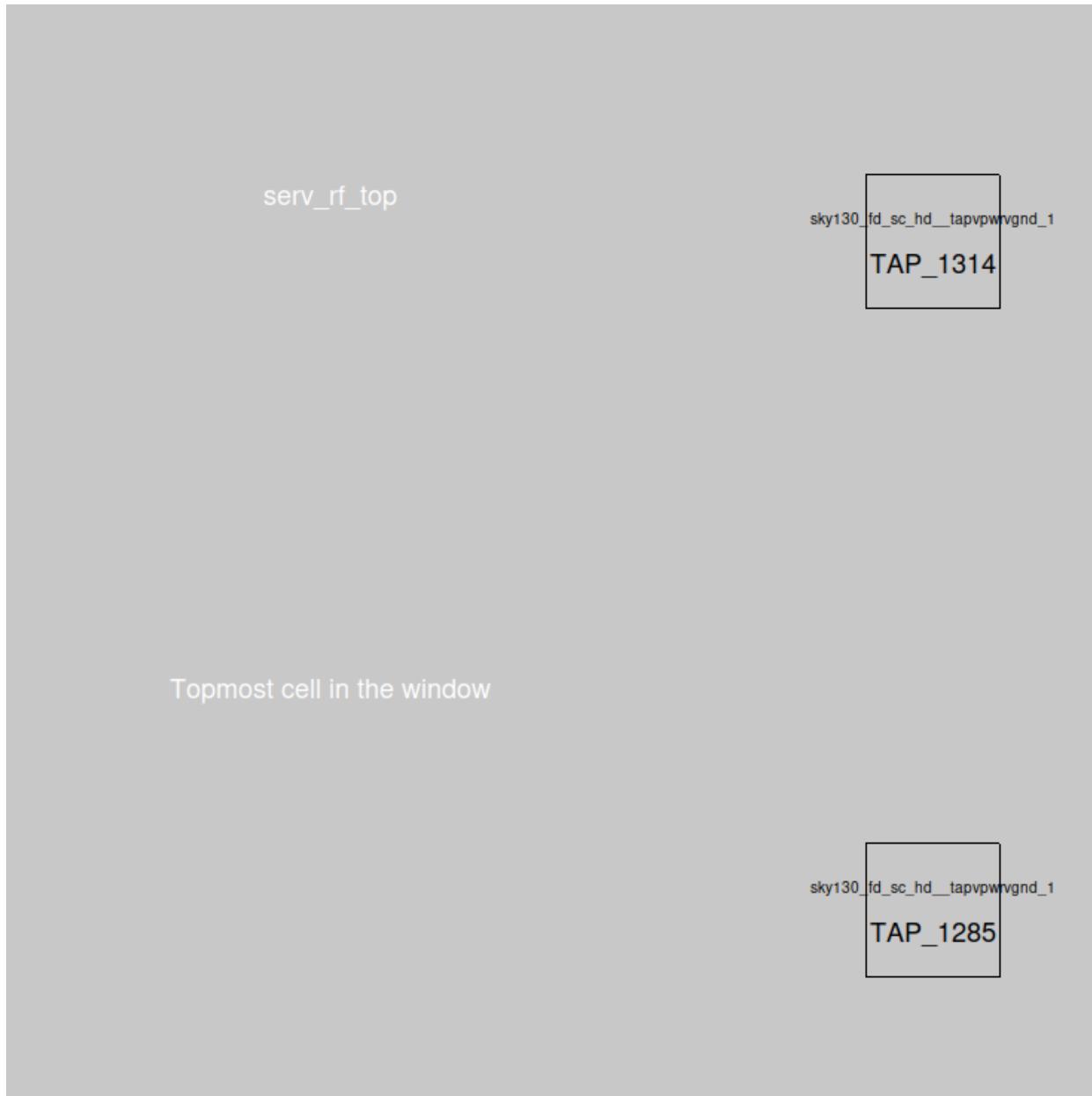
Post Floorplanning

The below command is for generating layout after Floorplanning. All the images after the below command are from what was generated during Floorplanning.

```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane/designs/serv_rf_top/runs/11-08_11-58/results/floorplan$ magic -T /home/badboy07/Desktop/OpenLane/pdks/open_pdks/sky130/libs.tech/magic/sky130.tech def read ../../tmp/merged.lef def read serv_rf_top.floorplan.def &
```



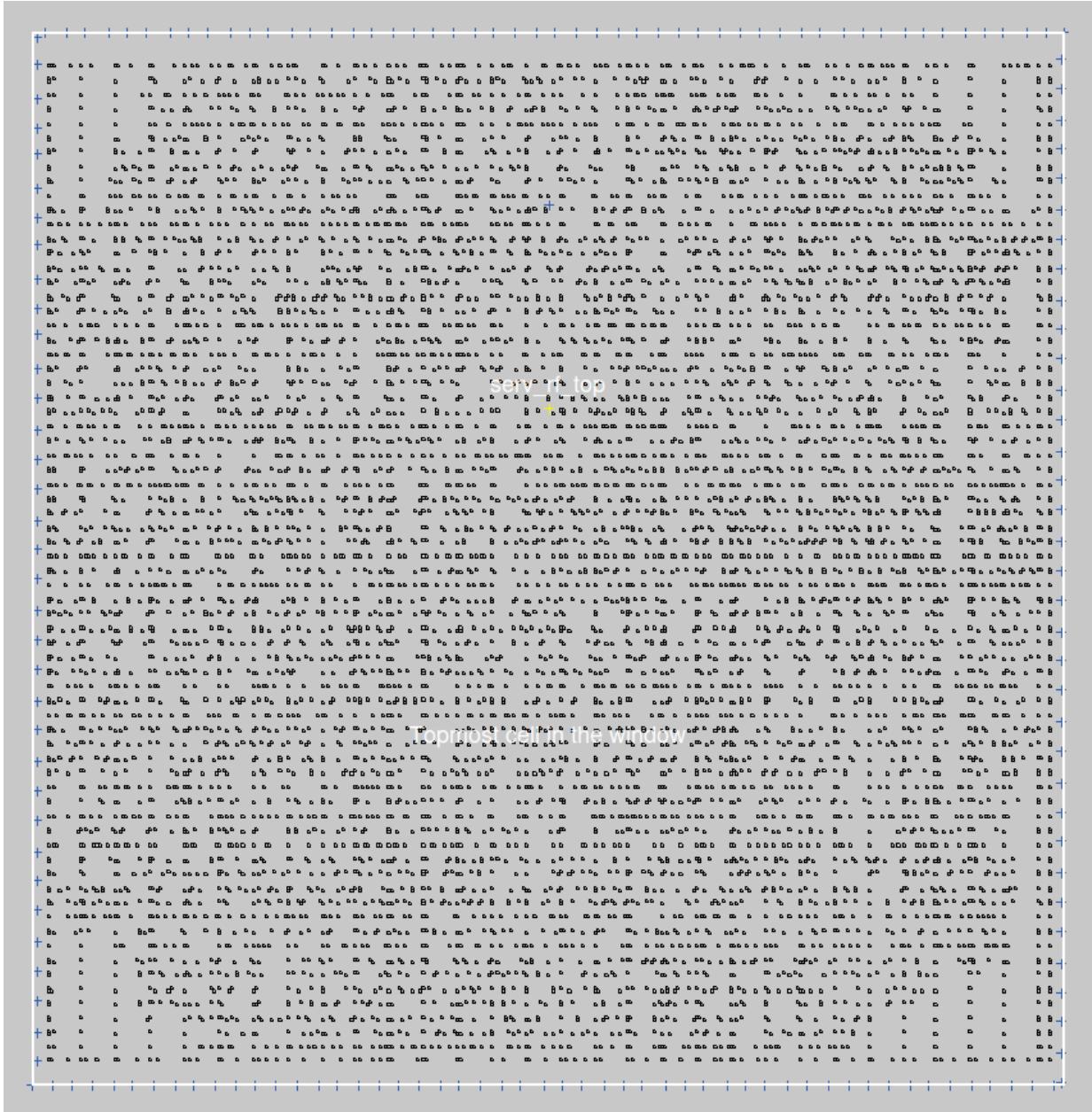


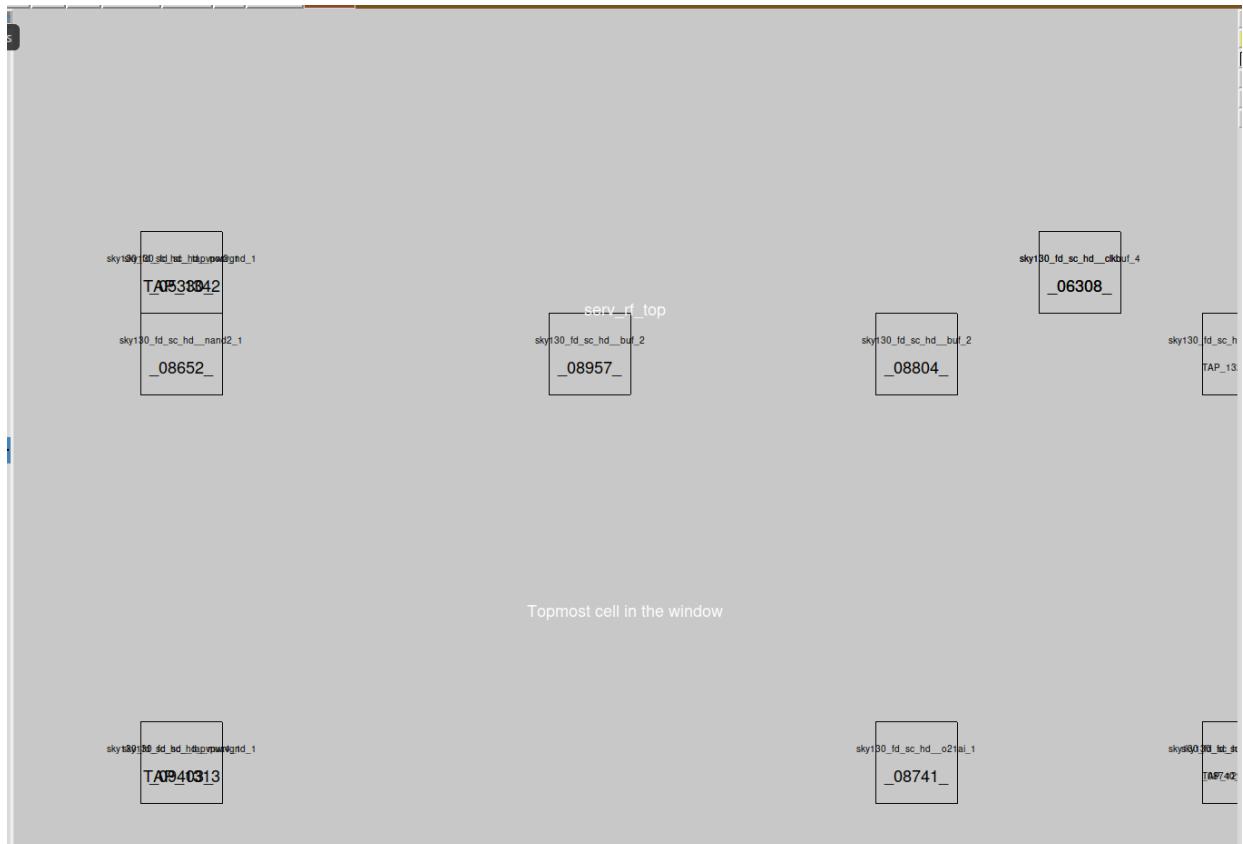
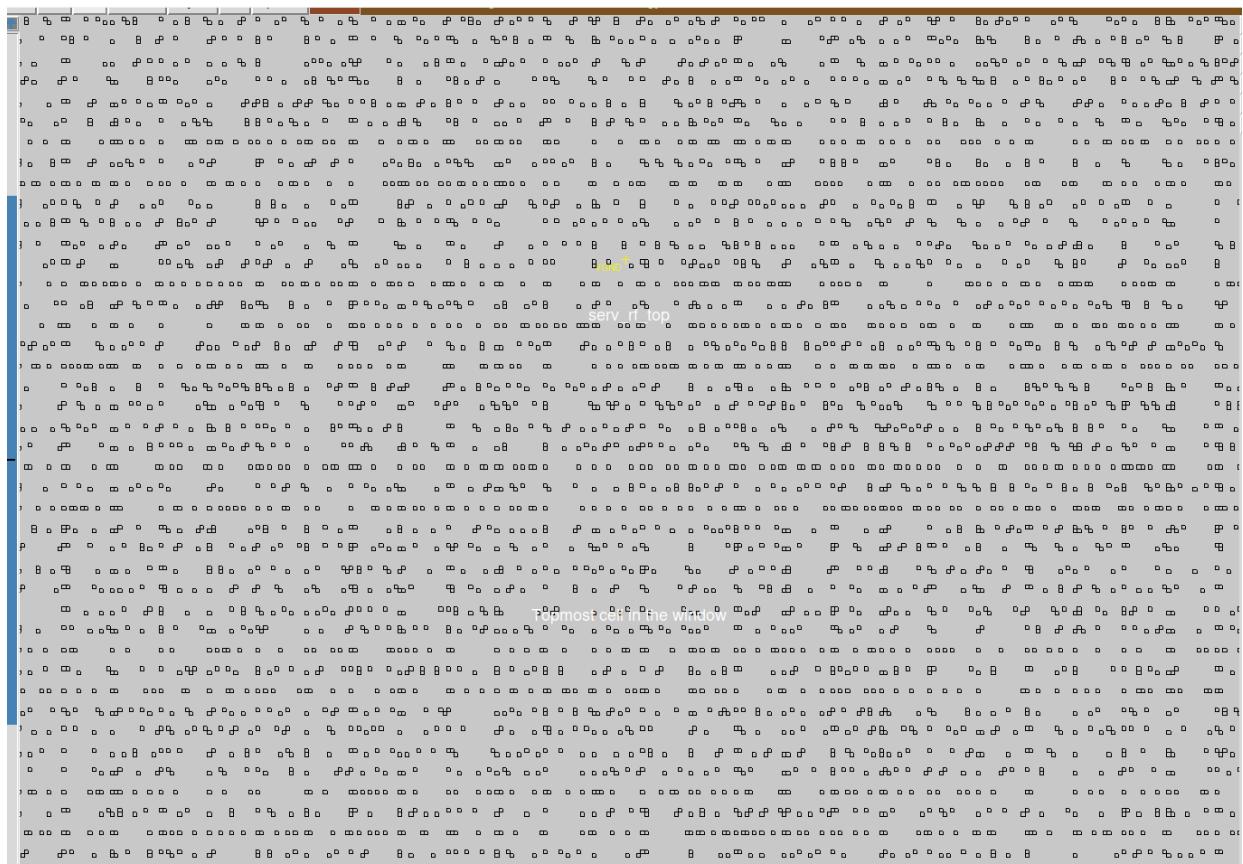


Post Placement

The below command is for viewing layout after placement. All the images after the below command are from what was generated during Placement.

```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane/designs/serv_rf_top/runs/11-08_11-58/results/placement$ magic -T /home/badboy07/Desktop/OpenLane/pdks/open_pdks/sky130/libs.tech/magic/sky130.tech def read ../../tmp/merged.lef def read serv_rf_top.placement.def &
```



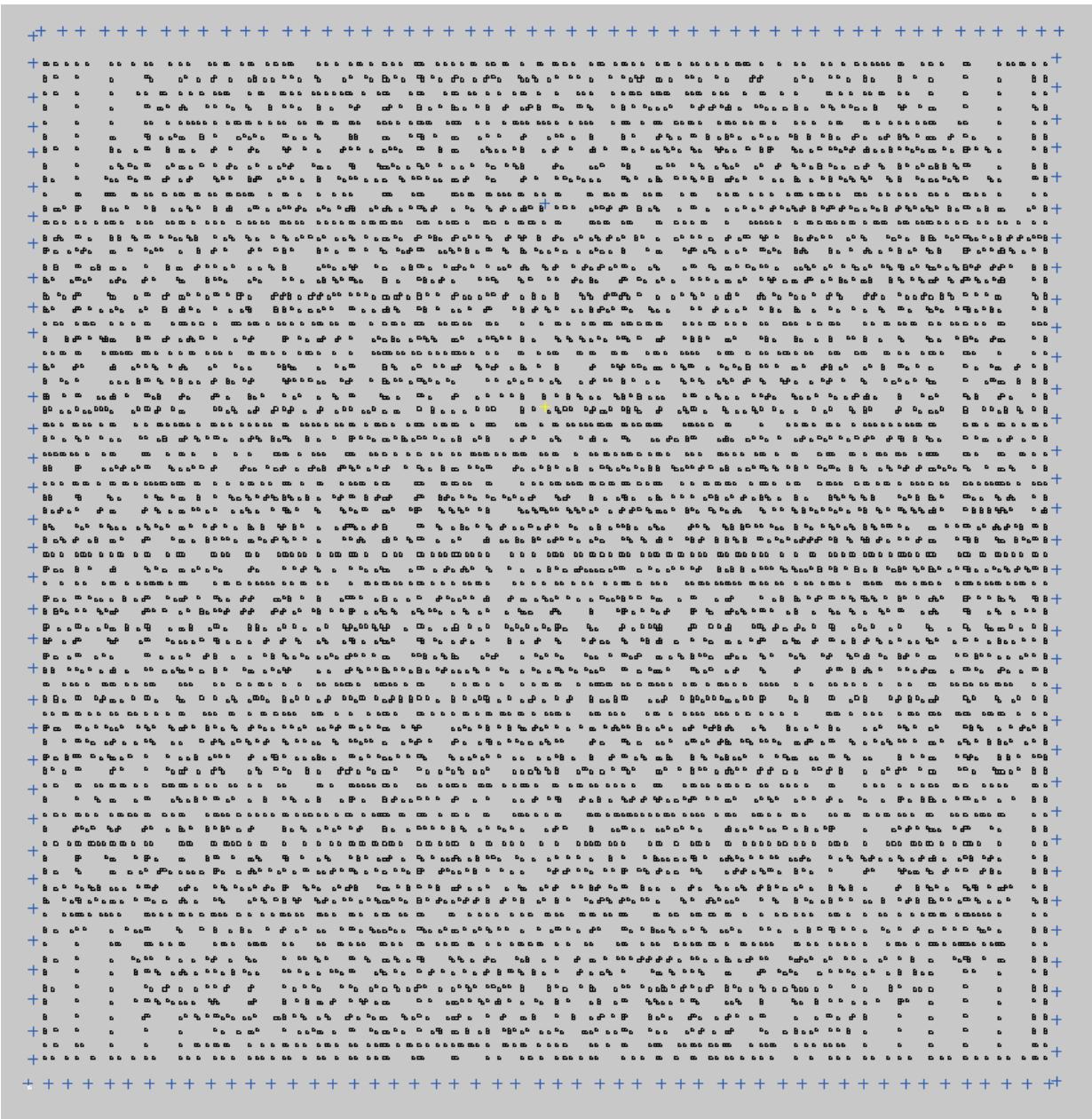


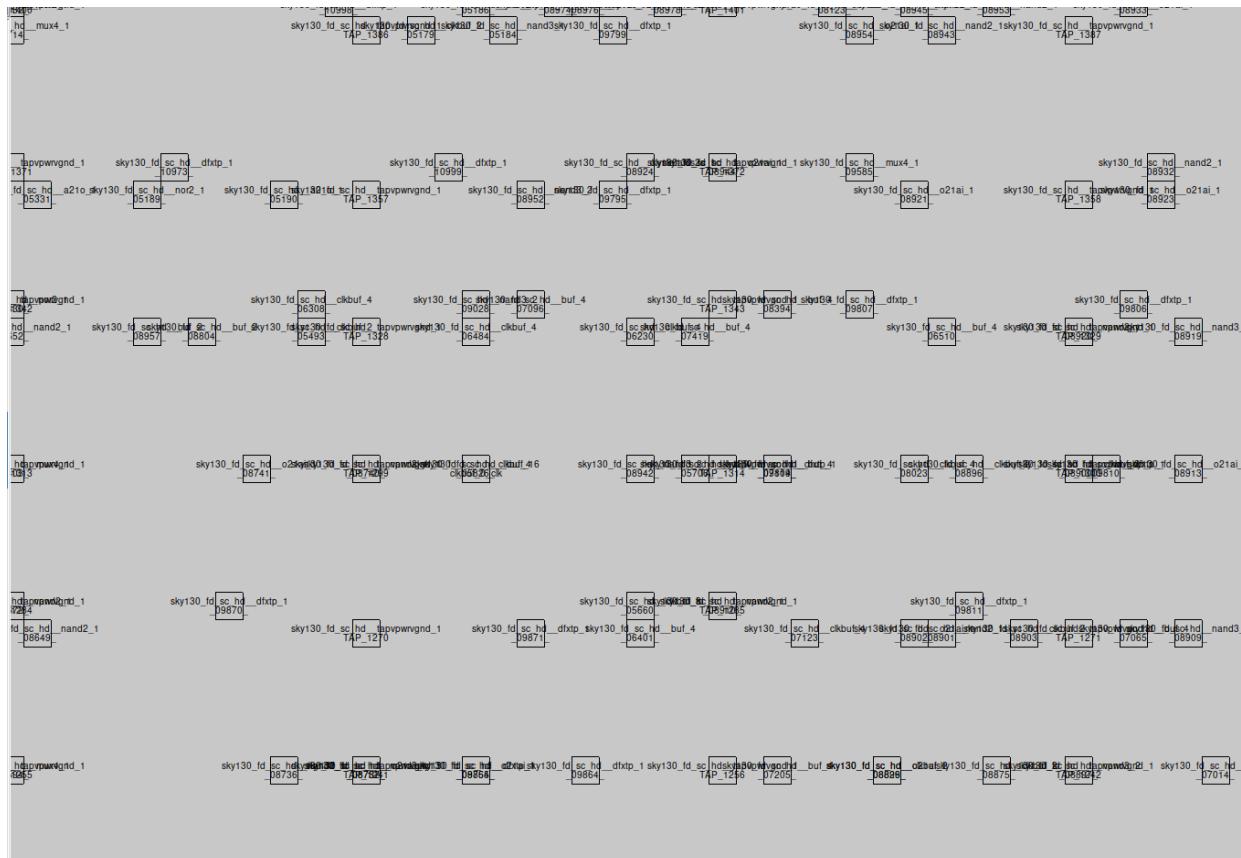
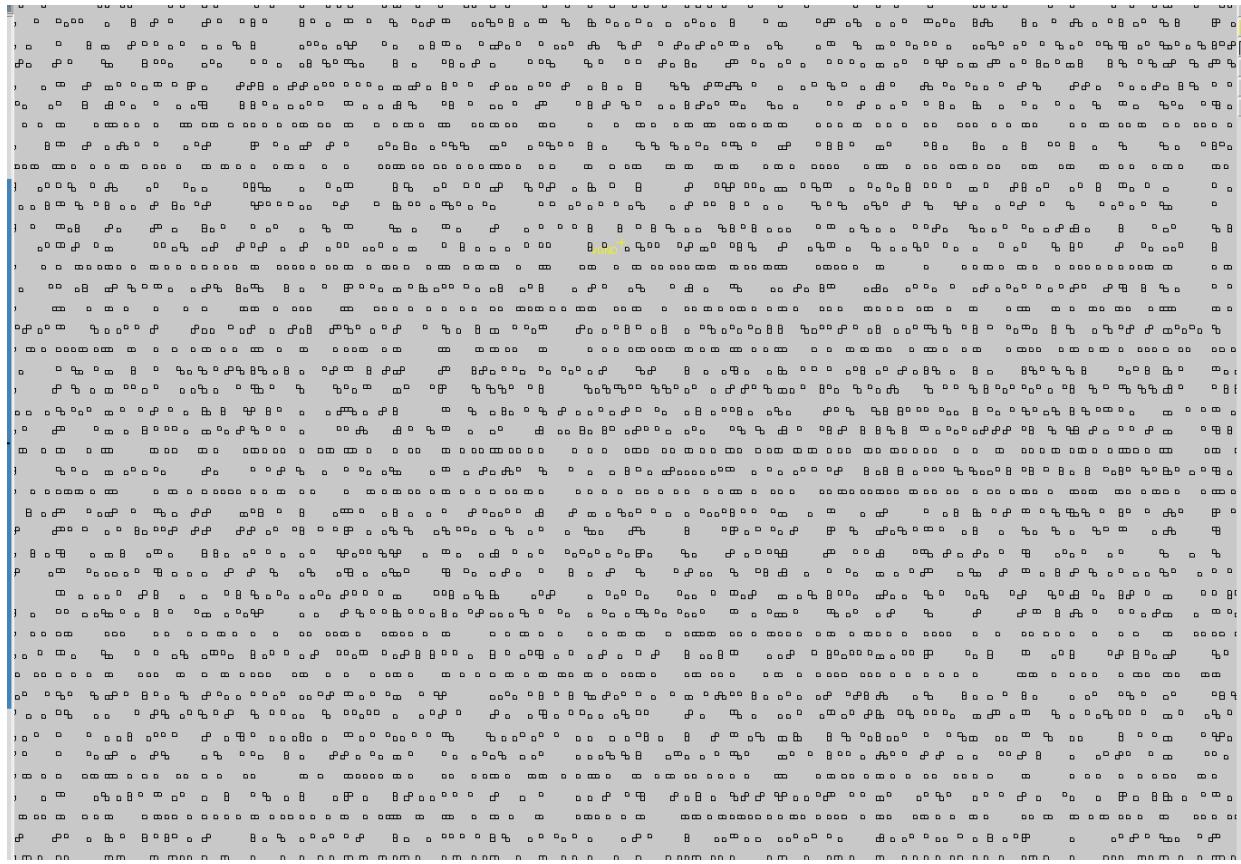


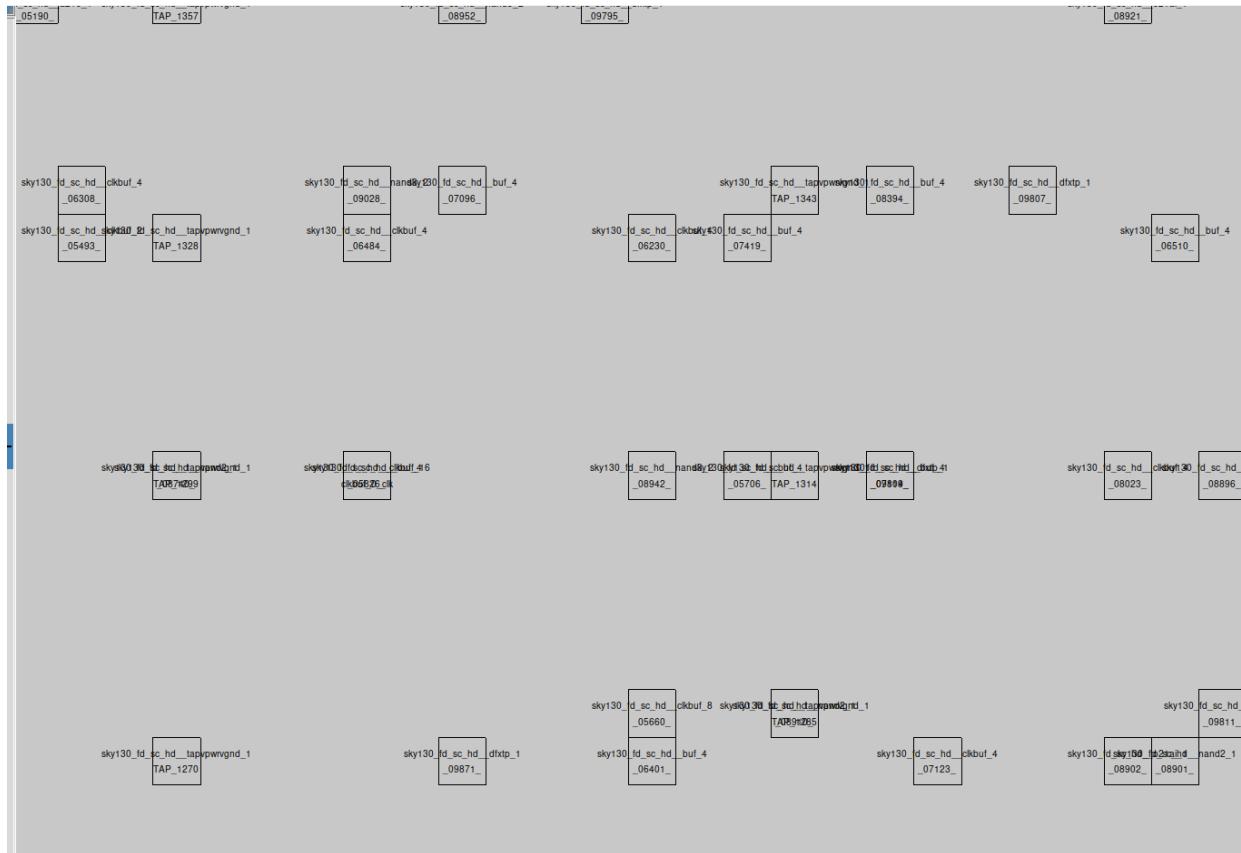
Post CTS

The below command is for viewing layout after CTS. All the images after the below command are from what was generated during CTS.

```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane/designs/serv_rf_top/runs/11-08_11-58/results/cts$ magic -T /home/badboy07/Desktop/OpenLane/pdks/open_pdks/sky130/libs.tech/magic/sky130.tech def read ../../tmp/merged.lef def read serv_rf_top.cts.def &
```





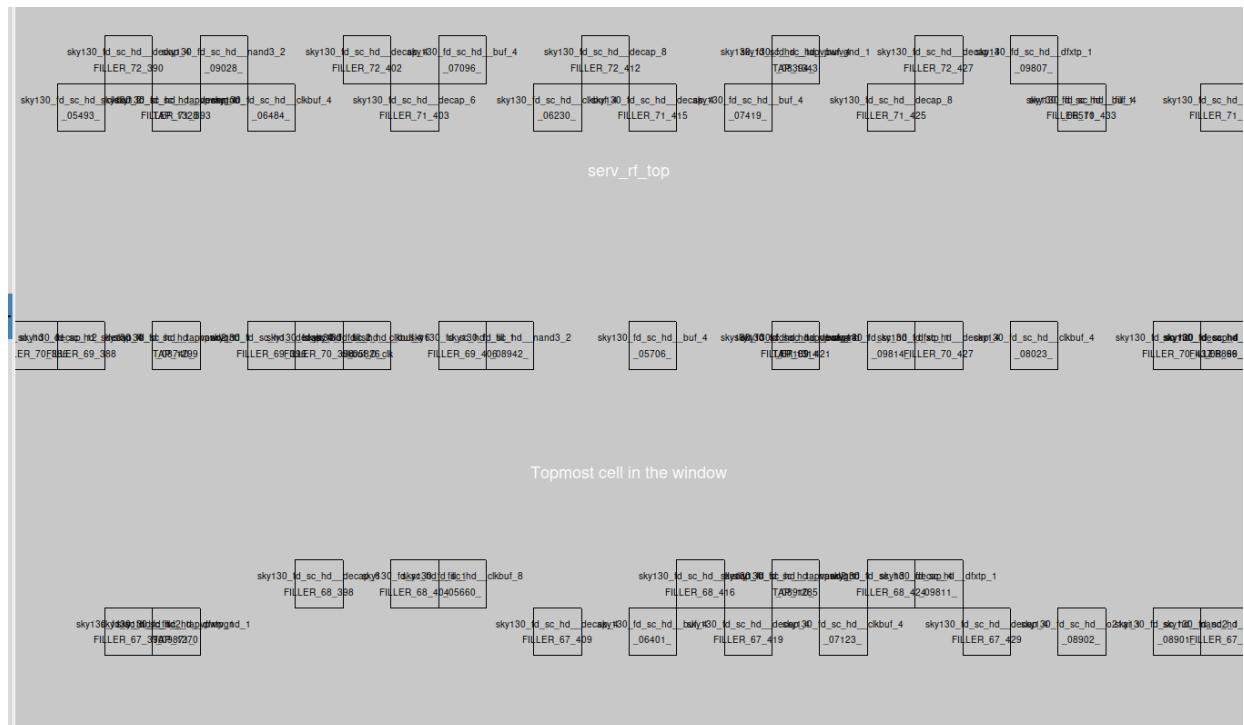
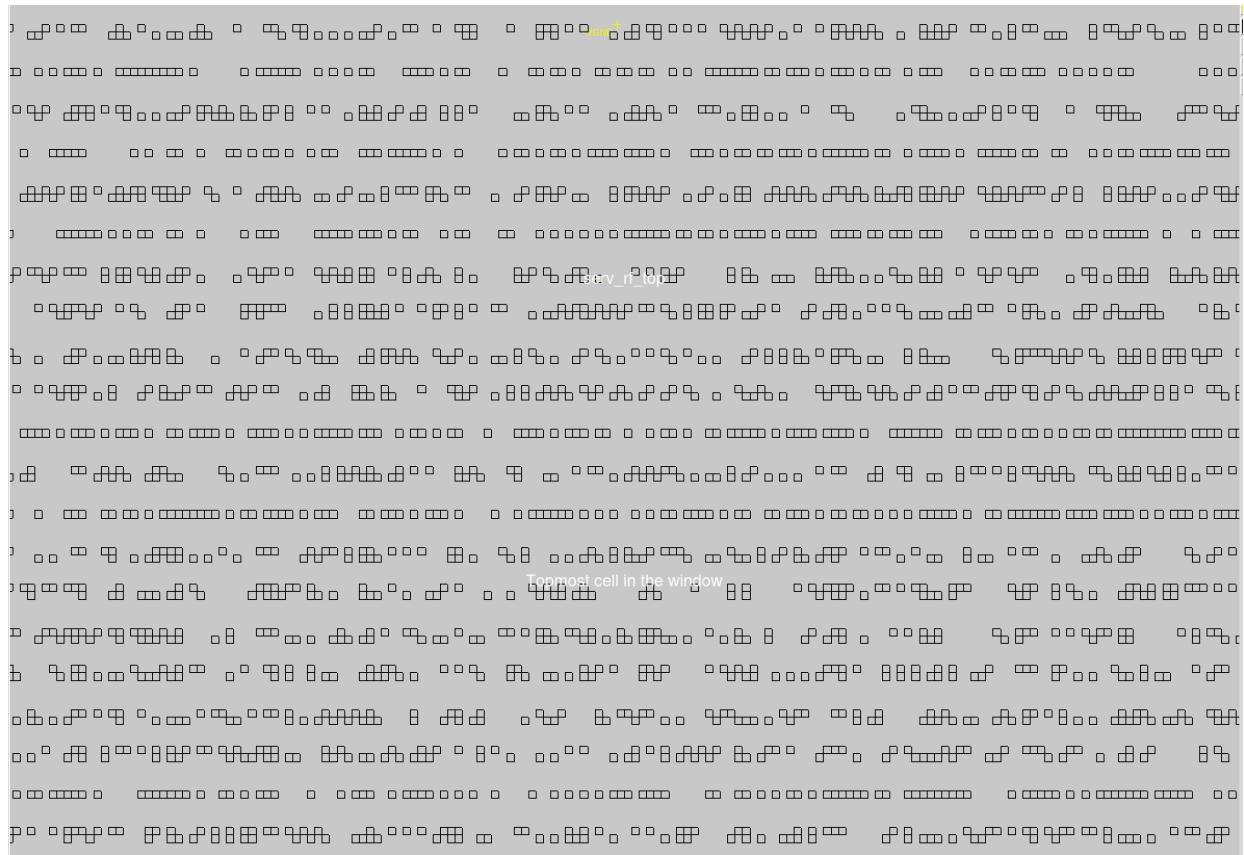


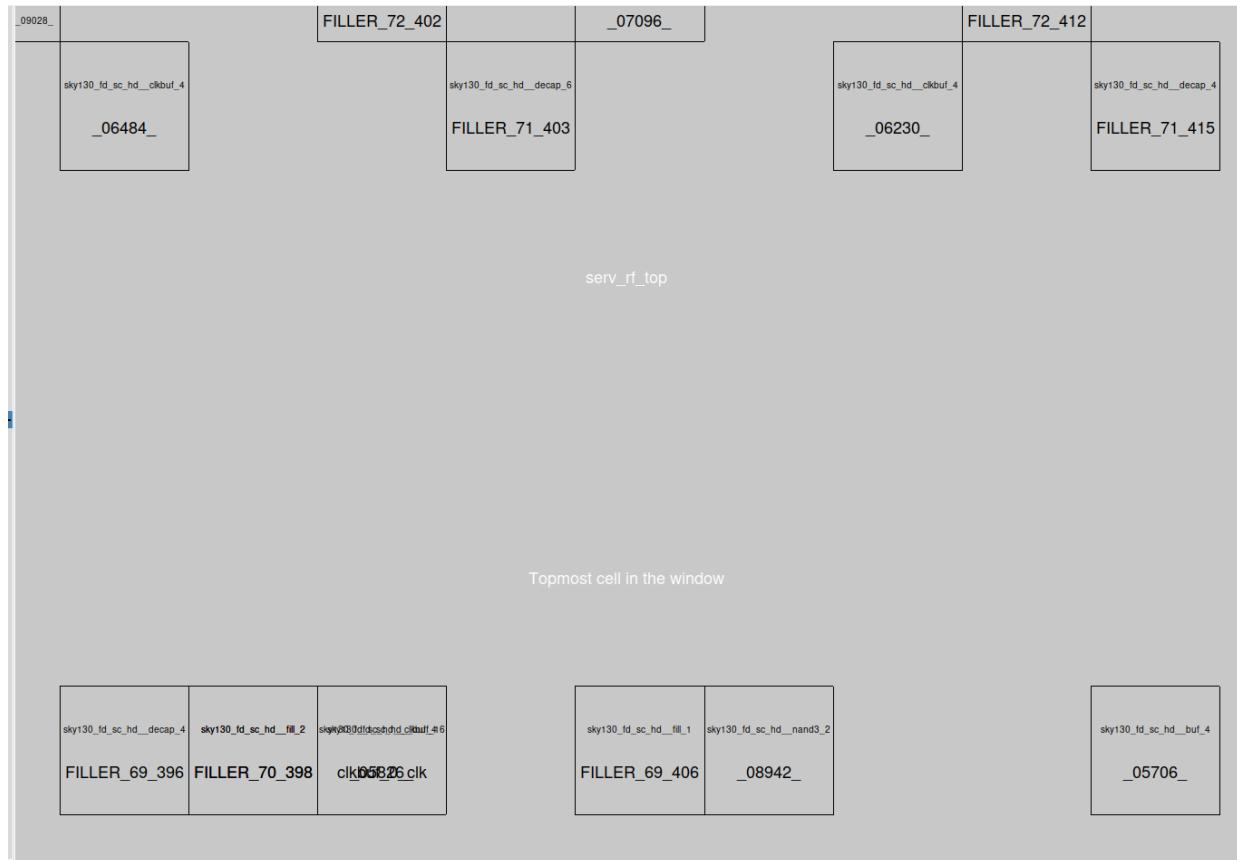
Post Routing

The below command is for viewing layout after routing. All the images after the below command are from what was generated during Routing.

```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane/designs/serv_rf_top/runs/11-08_11-58/results/floorplan$ magic -T /home/badboy07/Desktop/OpenLane/pdks/open_pdks/sky130/libs.tech/magic/sky130.tech def read ../../tmp/merged.lef def read 15-serv_rf_top.def &|
```





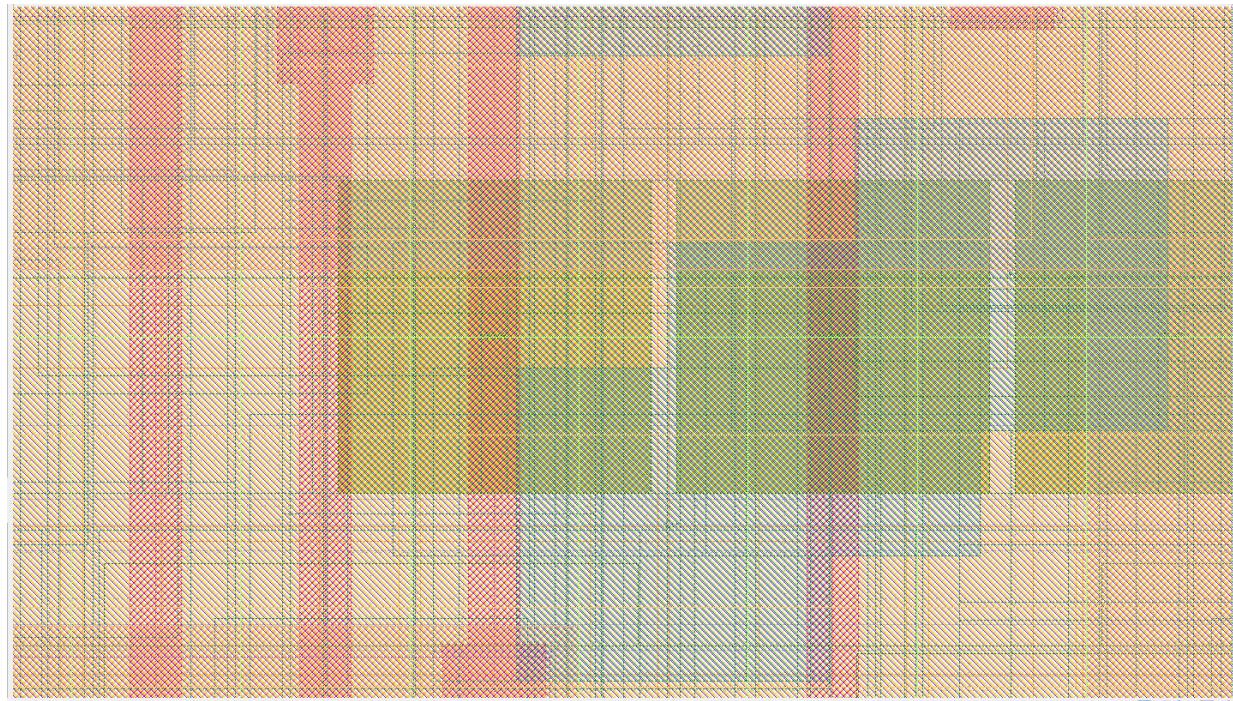
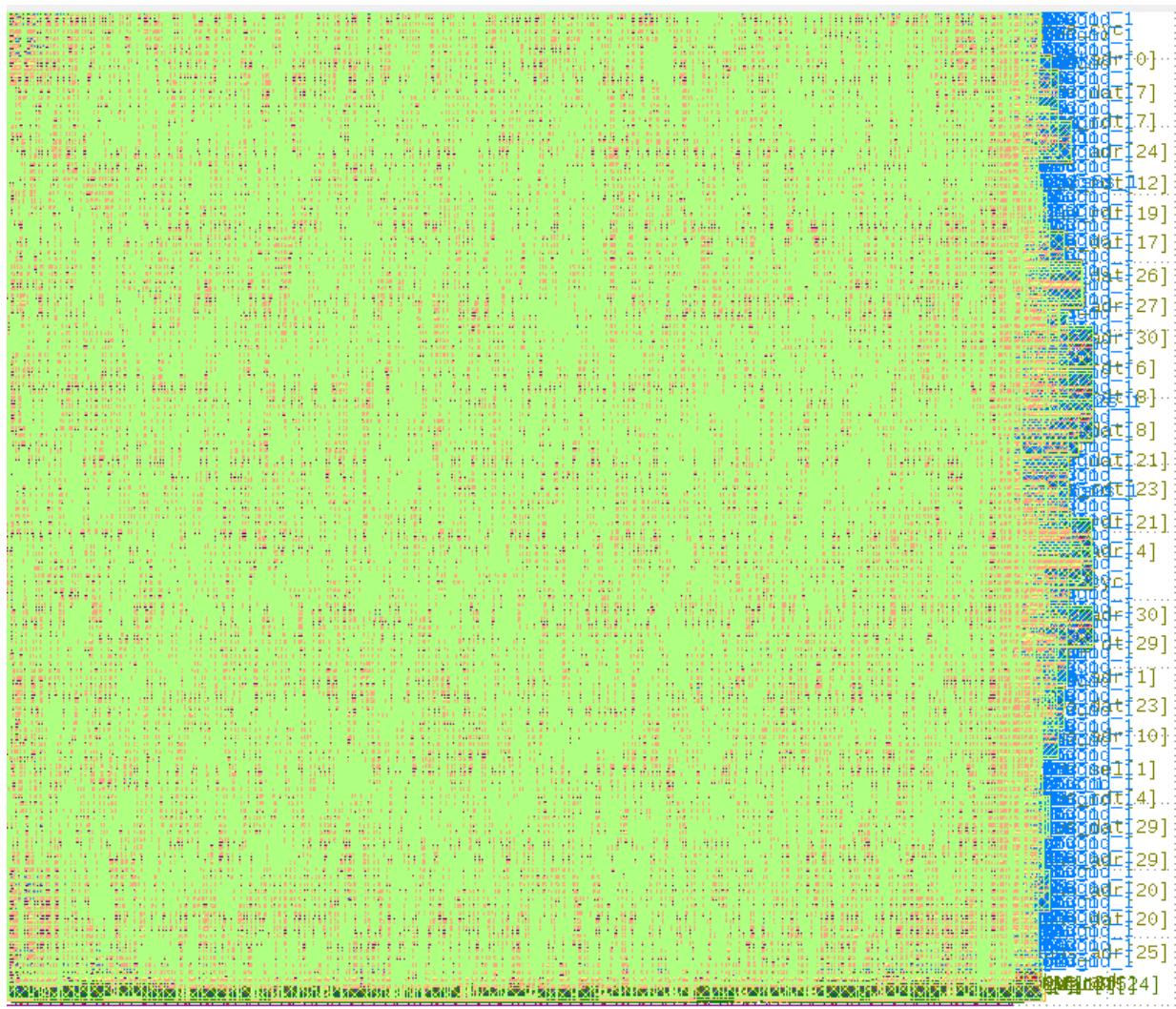


Final GDSII layout

The below command is for opening the GDS file created.

```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane/designs/serv_rf_top/runs/11-08_11-58/results/klayout$ klayout serv_rf_top.gds
```

The below layout was generated by KlAYOUT tool

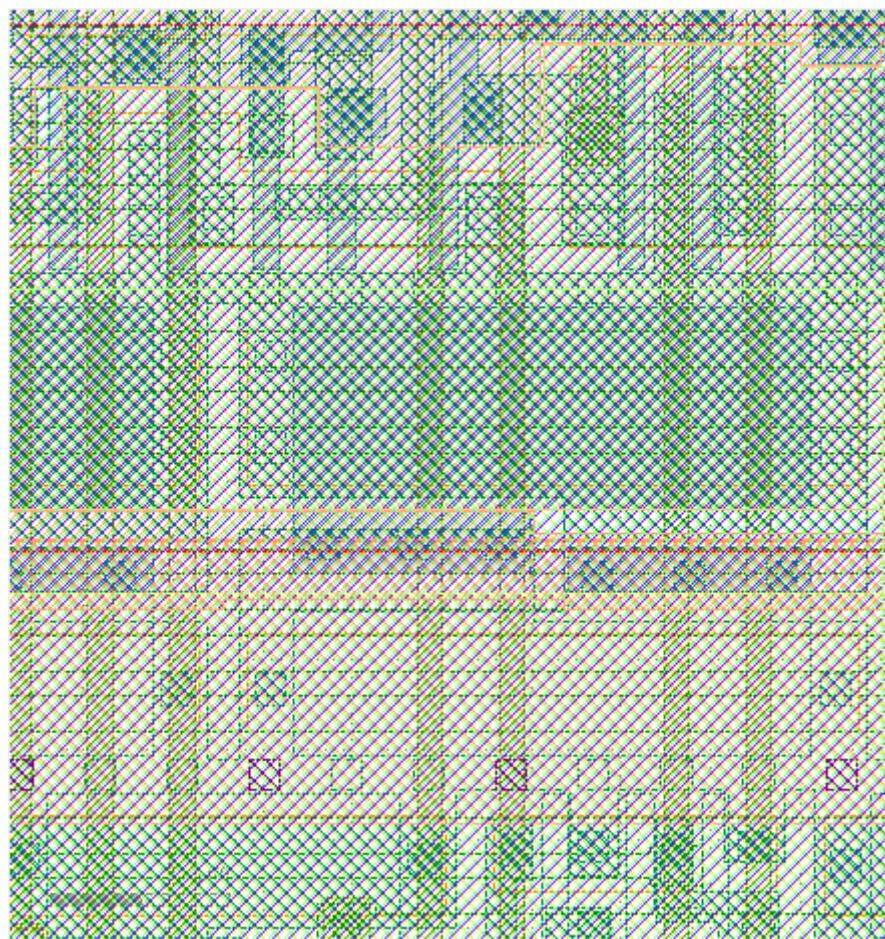


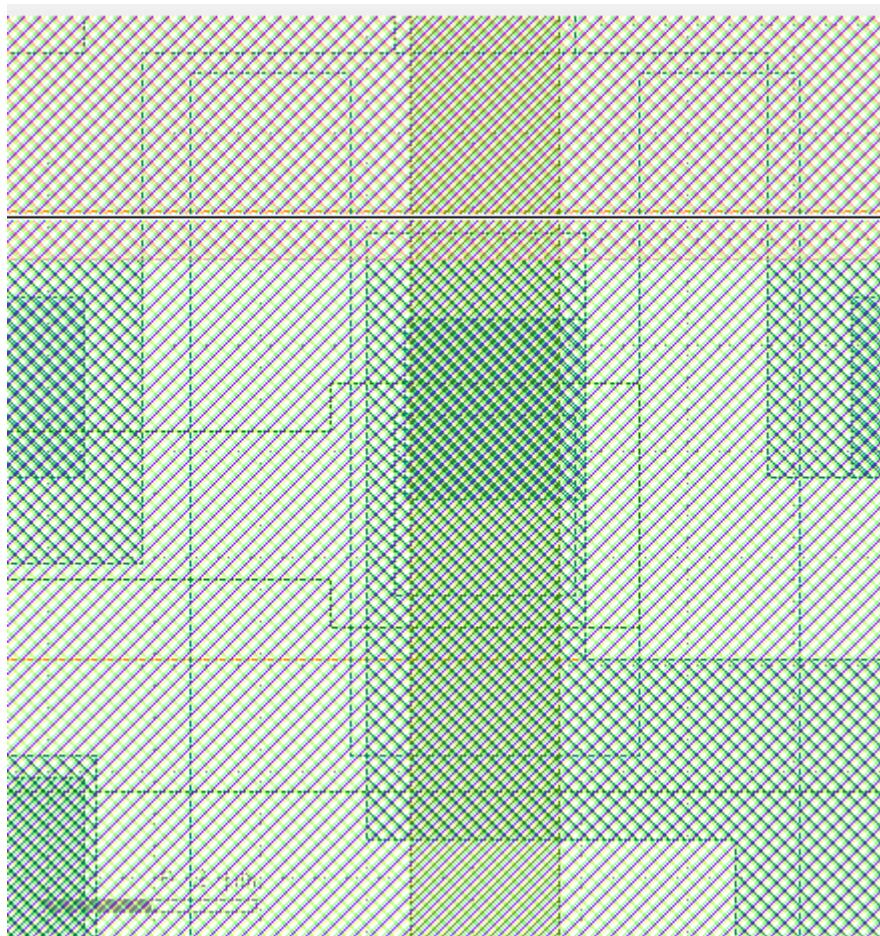


The below command is for opening the GDS file created.

```
(base) badboy07@badboy07-VirtualBox:~/Desktop/OpenLane/designs/serv_rf_top/runs/11-08_11-58/results$ klayout serv_rf_top.xor.gds
```

This is a layout with XOR design using Klayout tool





Inferences

Thus we have designed and synthesized and perform physical design on SERV core design using Yosys and OpenLane flow. We have generated reports for area, power, delay and timing analysis and utilization statistics after each stage and we have also viewed each stage layout by Magic tool and GDSII layout by Klayout tool.