Functional Coverage Analysis of Mixed-Signal Verification Based on Correlation Modeling

Jiagang Xu, Flagchip. Suzhou, China (jiagang.xu@flagchip.com.cn)

Sivan Huang, Flagchip. Suzhou, China (huang.song@flagchip.com.cn)

Eric Chen, Flagchip. Suzhou, China (ericchen@flagchip.com.cn)

Agenda

- What is Mixed-signal Verification
- Mixed-signal Verification Challenges
- Proposed Flow for Function Coverage Analysis
- Device Characteristics & Signal Flow Path
- Signal Flow Extraction
- Correlation Analysis
- Mapping Strategy
- Improvements

What is Mixed-Signal Verification

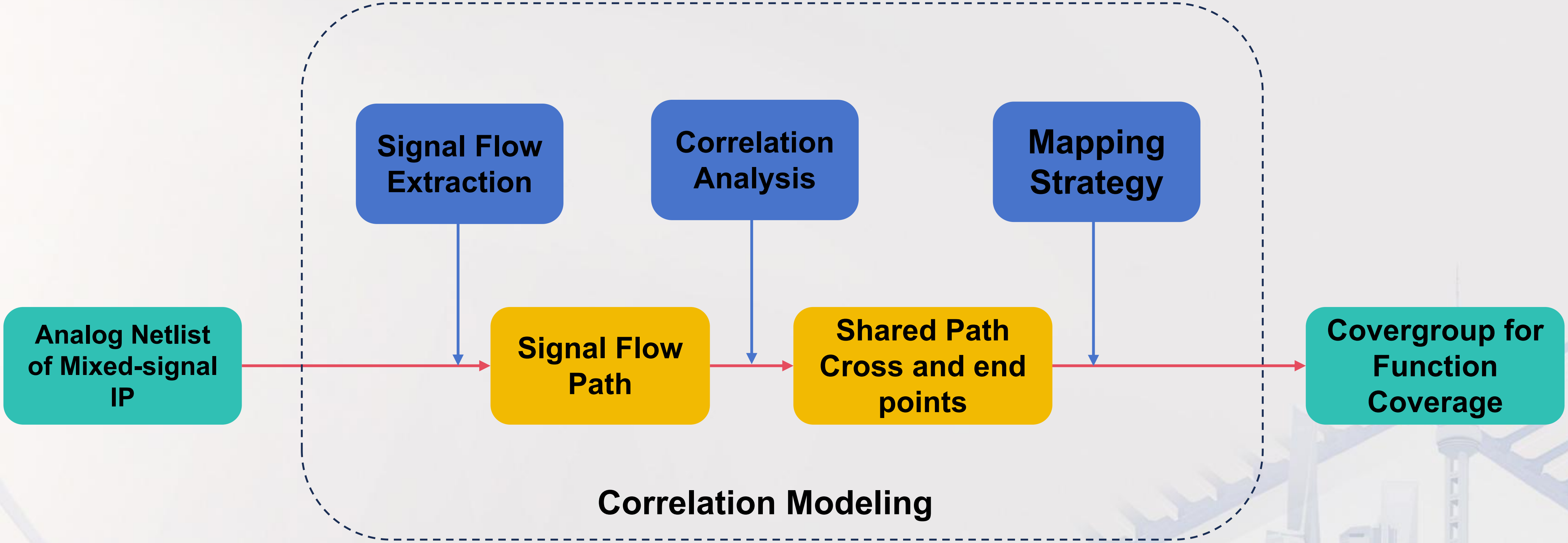
- Mixed-signal designs combine analog and digital signals and functionalities. Analog signals are continuous and can take on any value within a range, while digital signals are discrete and represent information using binary code (0s and 1s)
- Mixed-signal verification involves validating the interactions between analog and digital components, verifying the accuracy of analog signal processing, and ensuring the proper functioning of the overall system.

Stage	Digital Part	Analog Part	Performance	Accuracy
RTL Verification	RTL	BHV(Verilog/Verilog-A/Verilog-AMS/System-Verilog)	good	low
RTL Verification	RTL	Spice Netlist	reduced	high
Gate-Level Verification	Gate-Level Netlist	Spice Netlist	bad	high

Mixed-signal Verification Challenges

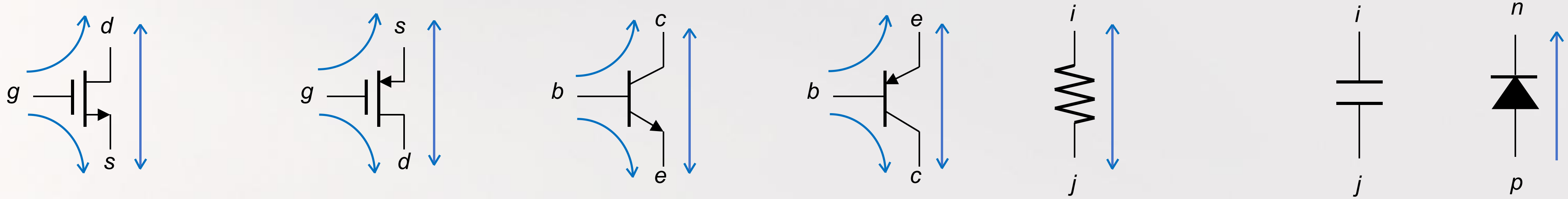
- **Complexity:** Mix of analog and digital components
- **Convergence:** Hard to ensure robust performance
- **Simulation time:** Longer simulations than pure digital designs
- **Lack of standardized methodologies:**
  - verification methodology
  - System-level modeling and verification
  - Analog fault injection
  - coverage
- **Debugging:** Complex due to interactions
- **Testbench development:** Comprehensive and effective testbenches needed

Proposed Flow for Function Coverage Analysis

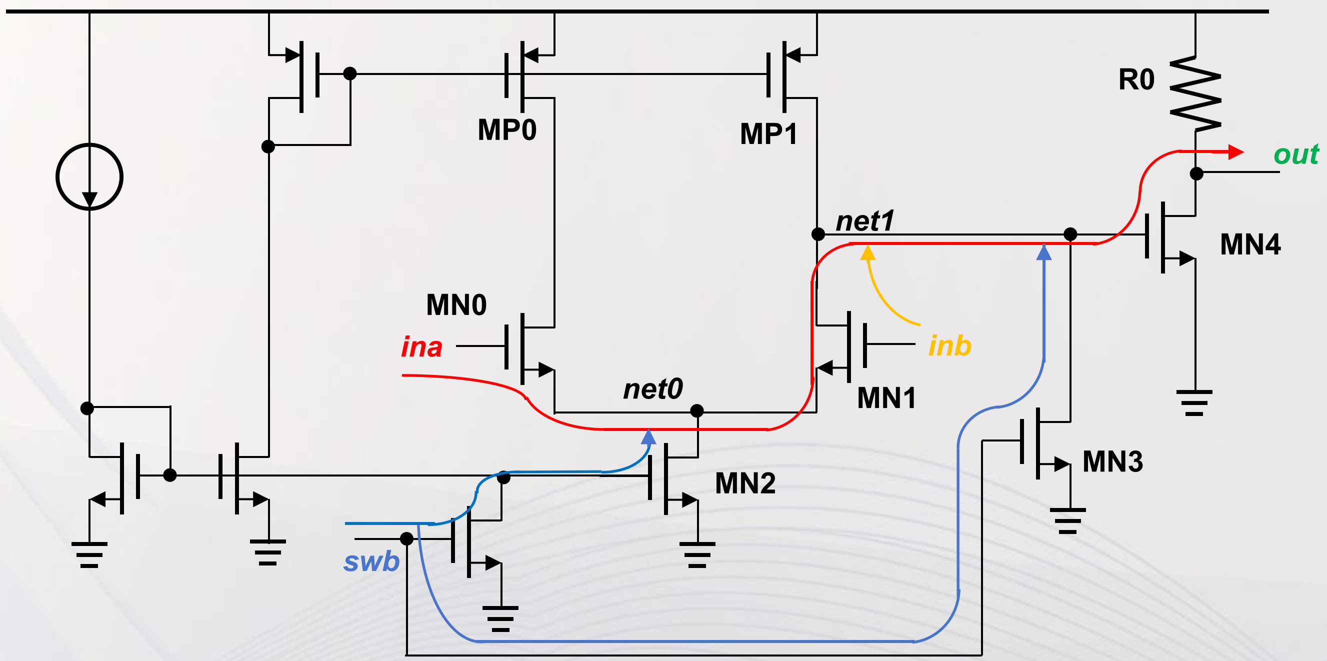


Device Characteristics & Signal Flow Path

- Define allowed signal flow for most commonly used devices in analog design



- Signal flow Path (Take two-stage amplifier as example)



After discarding unreachable/unallowed path:

ina->net0->net1->out

swb->net0->net1->out

inb->net1->out

Signal Flow Extraction - Algorithm

- **input:** analog netlist, **output:** signal flow path

Device CZ

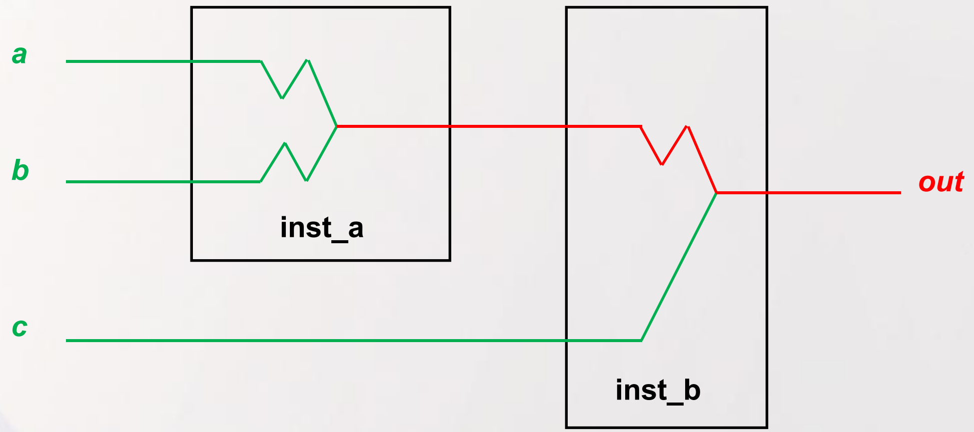
Algorithm 1: generate signal flow for inputs and output

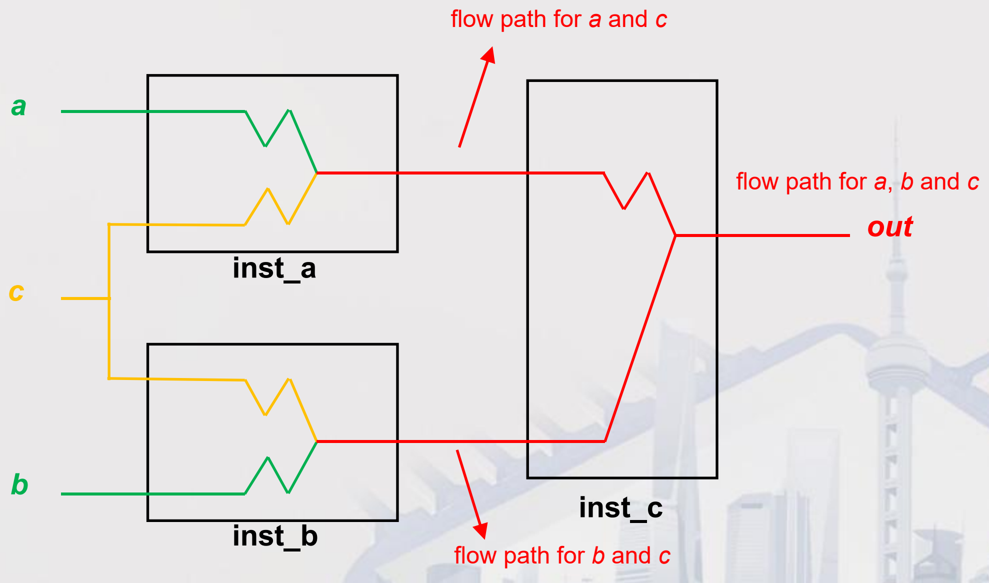
```
function GenFlowLayer(nets, top_net)
    nets.top_net.expanded <- True
    libcell_net <- find libcell net that connected to top_net
    for each net in libcell_net do
        Connected_nets <- all adjacent nets of libcell_net
        Nets.connected_nets.checked = True

        allowed_net <- FlowControl(libcell_net)
        for each n in allowed_net do
            if nets.n.expanded == False, and n is not power net then
                nets = GenFlowLayer(nets, n)
            end if
        end for
    end for
    return nets
end function
```

```
function FlowControl(libcell_net)
    if libcell_net belongs to MOSFET than
        if libcell='drain' then
            allowed_net = 'source'
        endif
        if libcell='gate' then
            allowed_net = 'source', 'drain'
        endif
        if libcell='source' then
            allowed_net = 'drain'
    end if
    if libcell_net belongs to BJT than
        …
    end if
    …
    return allowed_net
end function
```

Correlation Analysis - Shared path prototype

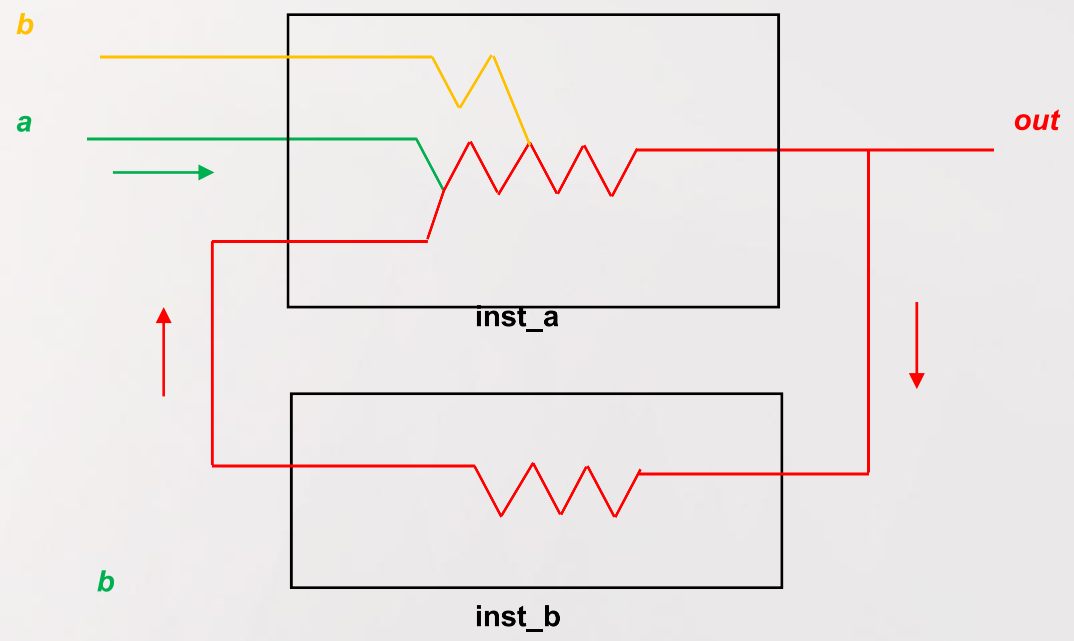
- Combination logics realized in analog circuits
- Gating control in sequential mode, such as scan safe, clock gating, on-off control





Correlation Analysis - Shared path prototype

- Shared path with circle involved, like LDO, PLL



Correlation Analysis – Cross/end points

- Find cross points (>=1) and end points for each shared path
- Cross points represent feature richness and help characterize functions

Correlation Analysis - Algorithm

- **input:** signal flow path, **output:** cross/end points

```
Algorithm 2: generate shared flow path
// nets contains all flow path information for all inputs and ouputs
// port_list is a list of inputs and outputs that need shared flow path to be analyzed.

function GenSharedFlow(nets, port_list)
    for each net in nets do
        if net.expanded is True for all port in port_list do
            append net to SharedNets
        endif
    end for

    while SharedNets is not empty do
        path <- first net in sharedNets
        pop path from sharedNets
        match_exist = True
        while match_exist do
            match_exist = false
            for each net in sharedNets do:
                if net is connected to path.connected_nets do
                    append net to path.connected_nets
                    pop net from sharedNets
                    match_exist <- true
                end if
            end for
        end while
        append path to paths
    end while
    return paths
end function
```

```
Algorithm 3: find cross points for each shared path

// path is element in paths
function FindCross(path, port_list)
    for each net in path.connected_nets do
        hiers <- all hierarchical names of net
        for each hier in hiers do
            if hier is the drain or source of MOSFET, or collector or emitter of BJT
            do
            append the source hier of MOSGET or base hier of BJT to pre_nets
        end for

        pre_net_not_shared <- True
        for each pre_net in pre_nets:
            if pre_net is not power net  and pre_net in path.connected_nets do
                prenet_not_shared = False
                break
            end if
        end for

        expanded_in_port <- False
        for each pre_net in pre_nets:
            for each port in port_list:
                if nets.pre_net.expanded is true for port do
                expanded_in_port <- True
                break
            end for
        end for

        if pre_net_not_shared is True and expanded_in_port is True do
            append net to path.connected_nets.cross
        end if
    end for
    return path
end function
```

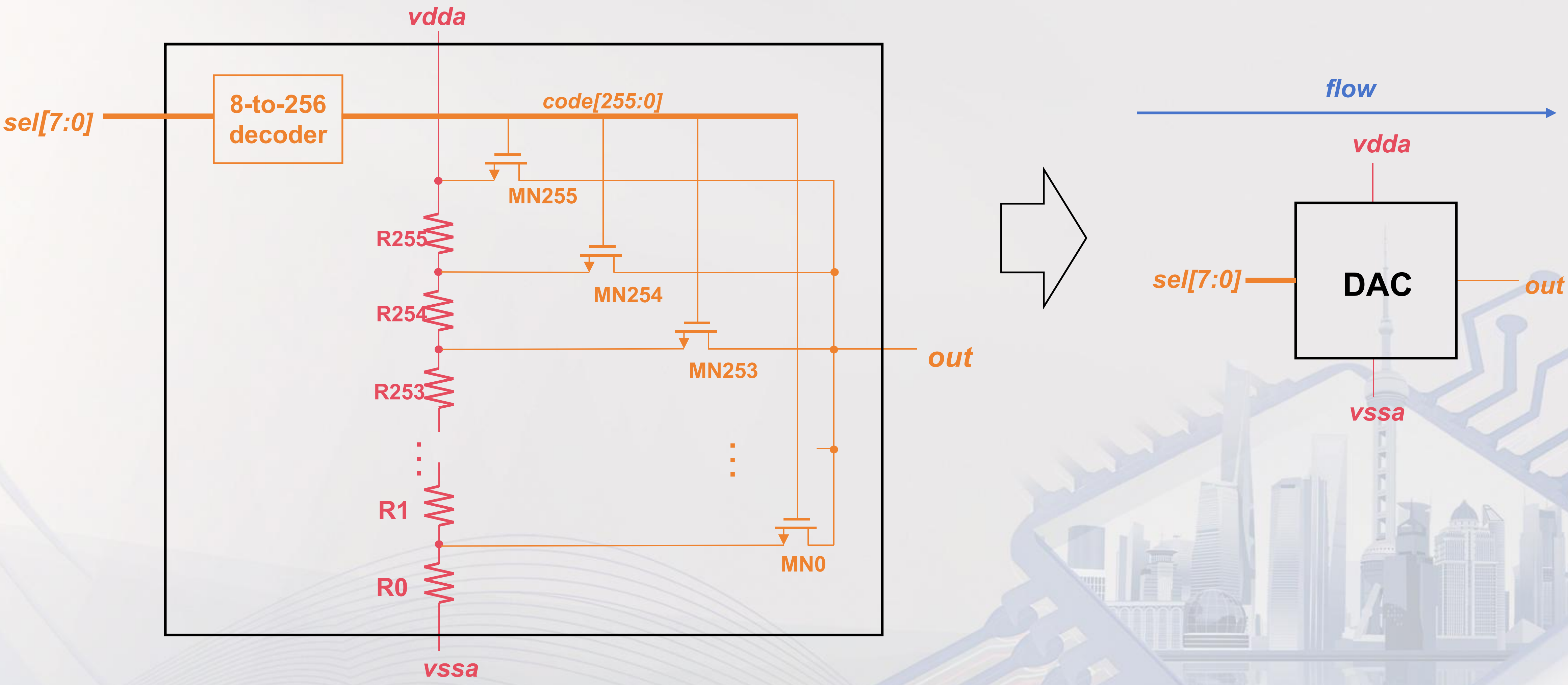
Mapping Strategy

- Example of Mixed-signal design of comparator

No. of combs	Interface names	Shared paths	Cross points	End points	Cover groups
2	acmp_en_lv, dac_en_lv	1	CMP.net0361, net020, net070	acmp_out_5v	acmp_en_lv,dac_en_lv=2’b00, 2’b01, 2’b10, 2’b11, power consumptions within spec.
2	offset_ctrl, input_shielding	0	NA	NA	NA
…	…	…	…	…	…
3	ina_5v_0, acmp_en_lv, fct_scan_mode	1	AMUX.SCAN,net0105, AMUX.SCAN.I46.net012	acmp_out_5v	When fct_scan_mode=1, all combs of ina_5v_0 and acmp_en_lv leads to acmp_out_5v=0.
3	acmp_en_lv, acmp_ho_pwr_lv, acmp_hystctr_lv	1	CMP.*,	acmp_out_5v	Cover all combs of acmp_en_lv, acmp_ho_pwr_lv, acmp_hystctr_lv.
…	…	…	…		…
4	dac_en, vref_sel, vref_int, vref_ext	1	DAC.vrefh, vrefh_ext_5v, vrefh_int_5v	acmp_out_5v, dac_out_5v	Cover all combs of dac_en, vref_sel, vref_int, vref_ext, expect right output of acmp_out_5v and dac_out_5v.
…	…	…	…	…	…

Improvements

- User-defined library cell to keep shared path clean and informative



Improvements

- Redefine nets for two-terminal device to shrink flow path.
  - ✓ For example, resisters used for current limitation, ESD protection, miller compensation does not contribute cross points.

- SKILL script to help highlight shared path back in Virtuoso schematic editor to improve efficiency.

Thank You!