BROWN
School of Engineering

DIGITAL ELECTRONICS SYSTEM DESIGN

FALL 2019

PROF. IRIS BAHAR

OCTOBER 2, 2019

LECTURE 9: MORE VERILOG & CMOS TRANSIENT BEHAVIOR

MORE TUTORIALS FOR VERILOG

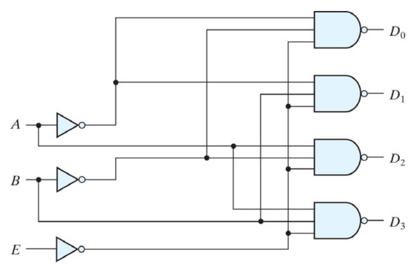
- On the course website you can find some useful links to additional Verilog examples
  - Example of a 3 bit counter
  - Blocking vs. non-blocking assignments within *always* blocks
  - State machine design for a Coke dispenser

DATASHEETS FOR PARTS

- All parts provided for you in our kits come with datasheets
  - Pin layout in package
  - Schematic design
  - Specify operating conditions
  - Provide description of how to operate chip correctly to get desired output
- The datasheets can be downloaded from the course webpage (FullDataSheets-ENGN1630.zip)
  - Please feel free to refer to these datasheets to help answer some questions you have for the lab assignments.

2:4 DECODER WITH ENABLE

- Complemented enable input
- 0-hot output encoding



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

2:4 DECODER: STRUCTURAL

```
// Gate-level (structural) description of 2-to-4 decoder
module decoder_2x4_gates(D, A, B, enable_);
  output      [3:0] D;
  input       A, B, enable_;
  wire     A_not, B_not, enable_not;

  not G1(A_not, A);
  not G2(B_not, B);
  not G3(enable_not, enable_);
  nand G4(D[0], A_not, B_not, enable_not);
  nand G5(D[1], A_not, B, enable_not);
  nand G6(D[2], A, B_not, enable_not);
  nand G7(D[3], A, B, enable_not);
endmodule
```

multi-bit output

2:4 DECODER: BEHAVIORAL (DATAFLOW)

```
// Behavioral (dataflow) description of 2-to-4 decoder
module decoder_2x4_df(D, A, B, enable_);
  output      [3:0] D;
  input       A, B, enable_;

  assign D[0] = ~((~A) & (~B) & (~enable_));
  assign D[1] = ~((~A) & B & (~enable_));
  assign D[2] = ~(A & (~B) & (~enable_));
  assign D[3] = ~(A & B & (~enable_));

endmodule
```

left hand side must be a wire (or output)

2:4 DECODER: BEHAVIORAL 2 (DATAFLOW)

```
// Behavioral description of 2-to-4 decoder
//  inputs A and B replaced with I[1:0]
module decoder_2x4_beh2(D, I, enable_);
  output      [3:0] D;
  input       [1:0] I;
  input       enable_;

  wire     [3:0] Di;

  assign Di = (I == 2'b11) ? 4'b0111:
          (I == 2'b10) ? 4'b1011:
          (I == 2'b01) ? 4'b1101: 4'b1110;

  assign D = (enable_ == 1'b0) ? Di : 4'b1111;

endmodule
```

conditional continuous assignment statement

VERILOG: CONTINUOUS ASSIGNMENTS

- Drive values onto a net
- Left hand side must be a **`wire`**, (or output)
- Continuous assignments are always active
- The assignment expression is evaluated as soon as one of the right-hand side operands changes
- Operands on the right-hand side can be registers or wires

VERILOG: PROCEDURAL ASSIGNMENTS

- Updates values of **`reg`** variables
- Value placed on a variable remains unchanged until another procedural assignment
- Not to be confused with continuous assignment (!) where the left-hand side is continuously driven

VERILOG: INITIAL BLOCKS

- An **`initial`** block:
  - starts at time 0
  - executes exactly once during a simulation
  - executes independently of other blocks
  - **`initial`** is not synthesizable so should not be used except for testbenches or algorithm development
  - Everything within the block concurrently active
  - When all processes are done, the block expires

VERILOG: INITIAL BLOCKS

```
module stimulus  // testbench
reg a, b, m;

initial
  m = 1'b0;   // single statement;  does not need begin/end

initial
begin
  #5 a = 1'b1;
  #25 b = 1'b0;
end

initial
  #50 $finish;

endmodule
```

VERILOG: ALWAYS

- An **`always`** block:
  - starts at time 0
  - executes statements continuously in a looping fashion, sequentially
  - executes independently of other blocks
- Keyword always @(sensitivity list)
  - Sensitivity list includes variables on right side of assignment statements inside block

VERILOG: ALWAYS

```
module clock_gen   // testbench
reg clock;

initial
  clock = 1'b0;   // initialize clock at time zero

// Toggle clock every half-cycle (time period = 20)
always
  #10 clock = ~clock;

initial
  #1000 $finish;

endmodule
```

VERILOG: ALWAYS

```
module pass_input(clock, in, out);
input   clock, in;
output  reg out;

// Toggle clock every half-cycle
always @(clock)
begin
   out = in;
end

endmodule
```

BLOCKING AND NON-BLOCKING ASSIGNMENTS

- Blocking assignments (X=A)
  - executed in the order they appear in a procedural block
  - completes the assignment before continuing on to next statement
- Non-blocking assignments (X<=A)
  - allow simultaneous scheduling
  - completes in zero time and doesn't change the value of the target until "end" is reached

BLOCKING AND NON-BLOCKING ASSIGNMENTS

- Example: swap

```
always @(posedge CLK)
    begin
        temp = B;
        B = A;
        A = temp;
    end
```

```
always @(posedge CLK)
    begin
        A <= B;
        B <= A;
    end
```

2:4 DECODER: BEHAVIORAL 3

```
module decoder_2x4_beh3(D, I, enable_);
  output      [3:0] D;
  input       [1:0] I;
  input        enable_;
  reg     [3:0] Di;

  always @(I)
  begin
   case (I)
     2'b11: Di<=4'b0111;
     2'b10: Di<=4'b1011;
     2'b01: Di<=4'b1101;
     default: Di<=4'b1110;
   endcase
  end
  assign D = (enable_ == 1'b0) ? Di : 4'b1111;
endmodule
```

assigned in a procedural block

sensitivity list

procedural block

non-blocking assignments

2:4 DECODER: TESTBENCH

```
`timescale 1 ns / 100 ps
module decoder_2x4_testbench;
  wire      [3:0] D;       // for instance outputs
  reg       enable_, [1:0] I;  // for instance inputs

  decoder_2x4_beh3 M1(D, I, enable_); // instance to be tested
  initial
  begin
    enable_ = 1'b1;        // initialize inputs
    I = 2'b0;
    #4 enable_ = 1'b0;
    #10 $finish;       // end simulation
  end

  always
  begin
    #1 I = I + 1'b1;
  end
endmodule
```

NMOS I-V SUMMARY

- *Shockley* 1st order transistor models

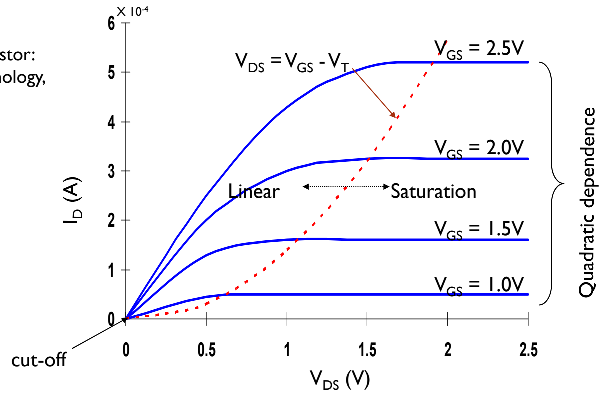
$$I_{ds} = \begin{cases} 0 & V_{gs} < V_{th} \quad \text{cutoff} \\ \beta\left(V_{gs} - V_{th} - {}^{V_{ds}}/_{2}\right)V_{ds} & V_{gs} < V_{dsat} \quad \text{linear} \\ \frac{\beta}{2}\left(V_{gs} - V_{th}\right)^2 & V_{ds} > V_{dsat} \quad \text{saturation} \end{cases}$$

$$\beta = \mu C_{ox}\frac{W}{L}$$

where C_{ox} is the capacitance per unit area of SiO_2

LONG CHANNEL I-V PLOT (NMOS)

NMOS transistor:
0.25um technology,
L_d = 10um,
W/L = 1.5,
V_{DD} = 2.5V,
V_T = 0.4V



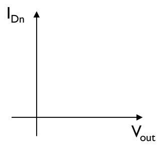
VOLTAGE TRANSFER CHARACTERISTICS

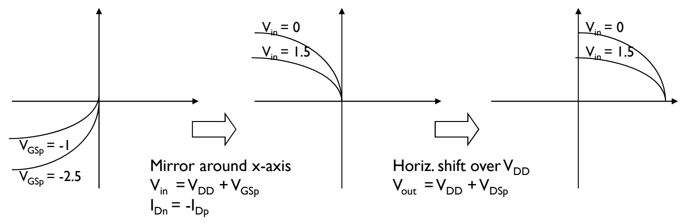
- What happens when input voltage is not "at rail"
- Vin < Vdd, or Vin > Gnd?
- If the transistor is ON, then voltage at output will change, but will not go to rail.

TRANSFORMING PMOS I-V LINES

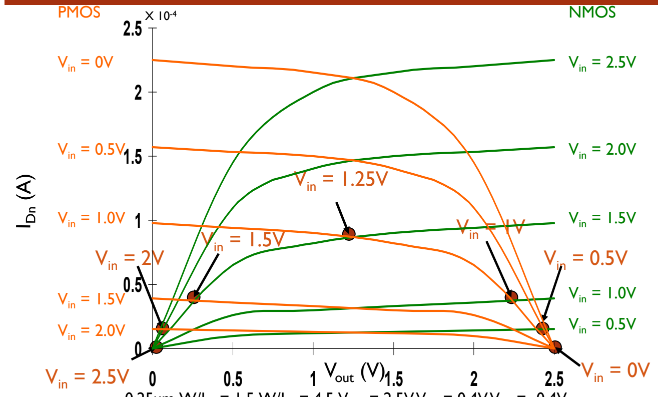
- Want common coordinate set V_{in}, V_{out}, and I_{Dn}

$$I_{DSp} = -I_{DSn}$$
$$V_{GSn} = V_{in} \quad ; V_{GSp} = V_{in} - V_{DD}$$
$$V_{DSn} = V_{out} \quad ; V_{DSp} = V_{out} - V_{DD}$$



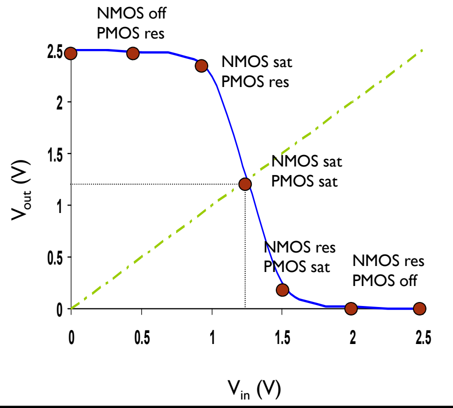


CMOS INVERTER I-V CURVES



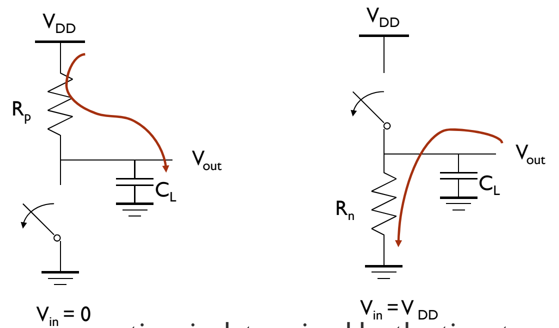
0.25um, W/L_n = 1.5, W/L_p = 4.5, V_{DD} = 2.5V, V_{Tn} = 0.4V, V_{Tp} = -0.4V

CMOS INVERTER VTC



Taking those intersection points from the load curves, we obtain the voltage-transfer characteristic

CMOS INVERTER:
SWITCH MODEL OF DYNAMIC BEHAVIOR



- Gate response time is determined by the time to charge C_L through R_p (discharge C_L through R_n)

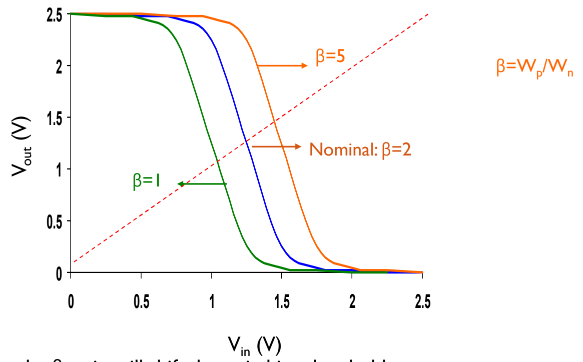
SWITCHING THRESHOLD

- Define V_M to be the point where $V_{in} = V_{out}$ (both PMOS and NMOS in saturation since $V_{DS} = V_{GS}$)
- If $V_M = V_{DD}/2$, then this implies *symmetric rise/fall* behavior for the CMOS gate
- Recall at saturation, $I_D=(k'/2)(W/L)(V_{GS}-V_{th})^2$,
  - where $k'_n = \mu_n C_{ox} = \mu_n \varepsilon_{ox}/t_{ox}$
- Setting $I_{Dp} = -I_{Dn}$ $$\frac{k_n'}{2}\frac{W_n}{L_n}(V_M - V_{Tn})^2 = \frac{k_p'}{2}\frac{W_p}{L_p}(-V_M - V_{Tp})^2$$
- Assuming $V_{thN} = -V_{thP}$ $$\frac{W_p/L_p}{W_n/L_n} = \frac{k_n'}{k_p'} = \frac{\mu_n}{\mu_p}$$

RELATIVE TRANSISTOR SIZING

- When designing static CMOS circuits, balance the driving strengths of the transistors by making the PMOS section wider than the NMOS section to
  - maximize the noise margins and
  - obtain symmetrical characteristics

IMPACT OF UNMATCHED DRIVE STRENGTHS



- Skewing the β ratio will shift the switching threshold