Improving Power Supply Induced Jitter Simulation Accuracy for IBIS Model

Yin Sun, Chulsoon Hwang

EMC Laboratory

Missouri University of Science and Technology

IBIS Summit at
2020 IEEE Virtual Symposium on EMC+SIPI
August 28, 2020

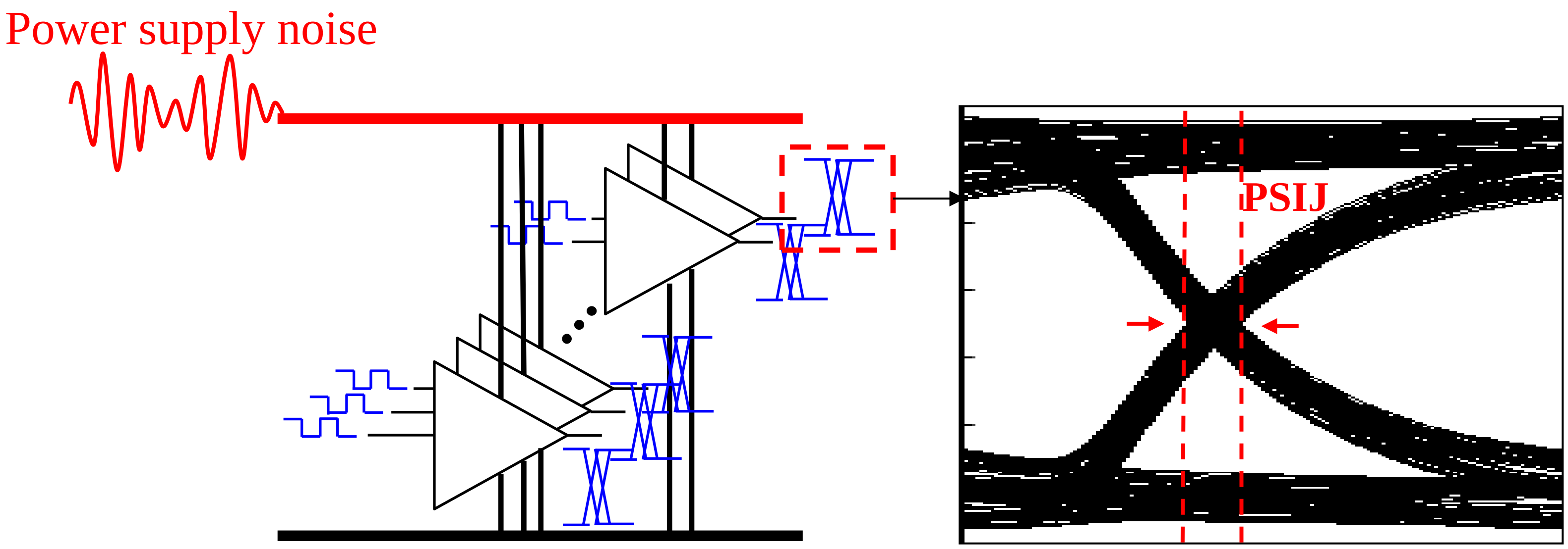
Outline

- Introduction of Power Supply Induced Jitter (PSIJ)
- Limitations of Current Power-Aware IBIS Model
- New Behavior Model Proposal
- Model Validation
- Conclusions

Power Supply Induced Jitter (PSIJ)

Power supply induced jitter

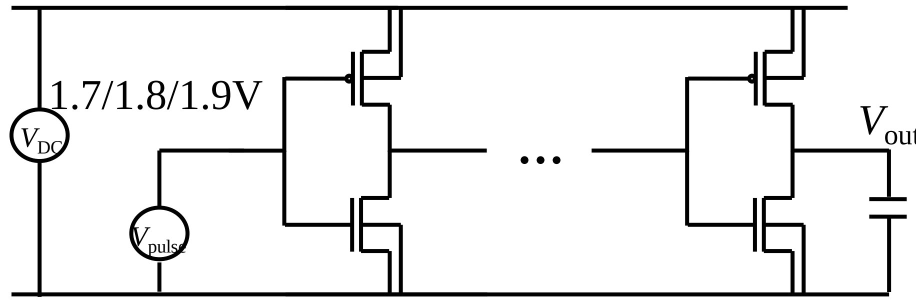
- The time variation in the output transition edges from ideal positions due to the voltage fluctuations on power rail.



Limitations of Current Power-Aware IBIS Model

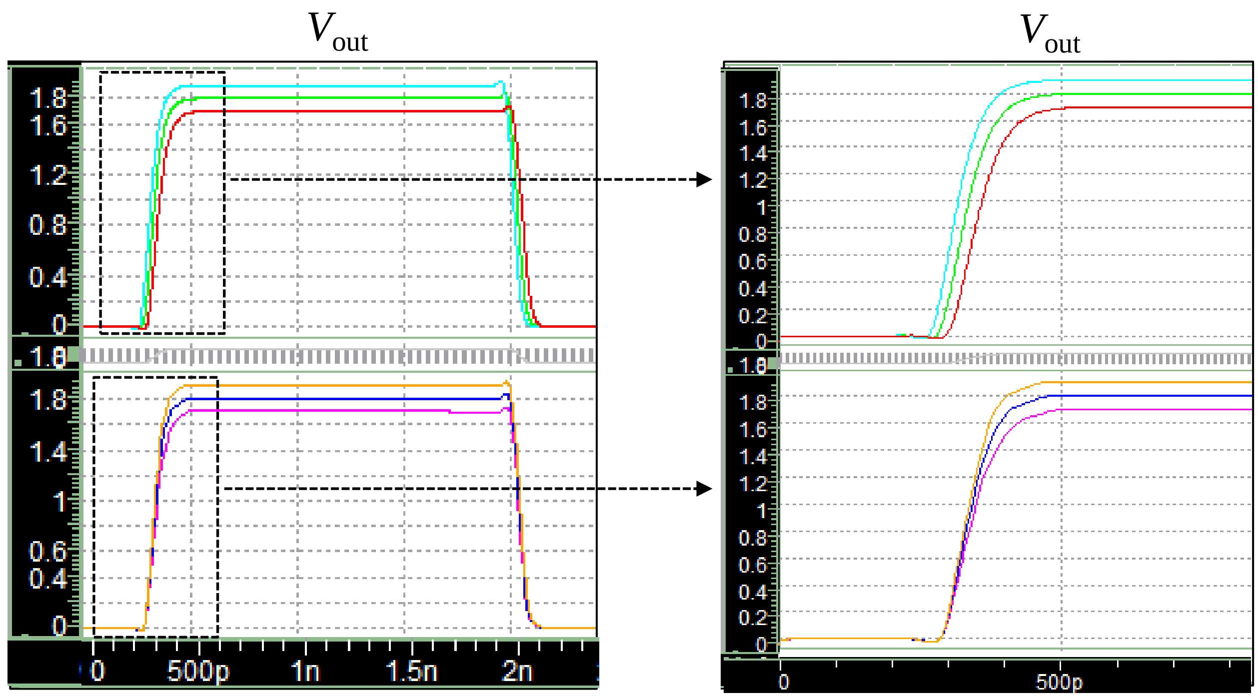
- **Cannot** account for the delay change caused by power noise correctly.

  - Example: an inverter chain output, change power voltage to 1.7/1.8/1.9V, respectively



Spice Results

Power-aware IBIS model Results
(ver5.1,generated with EDA tool)



Limitations of Current Power-Aware IBIS Model

- Power-aware IBIS model considers gate modulation effect, ratio modification on Ku, Kd based on power rail voltage value

Gate Modulation Coefficients

The ST "Gate Modulation" solution is based on the introduction of two coefficients, one for the Pullup and one for the Pulldown stage, which modulate properly the IBIS standard current (I_IBIS-STD) when a bouncing noise occurs on the power and ground nodes

I(Vgs, Vds) = Kssn(Vgs,Vds)*I(Vgs=VDD,Vds)

Effective SPICE current — IBIS standard current

I_effective = Kssn(Vgs,Vds)*I_IBIS-STD

$$K_d(t) I_{pd} \rightarrow K_{sspd}(V_{pd}) K_d(t) I_{pd}$$

$$K_u(t) I_{pu} \rightarrow K_{sspu}(V_{pu}) K_u(t) I_{pu}$$

$$K_{sspd}(V_{pd}) = \frac{V_{pd}}{I_{sspd}(0)}$$

$$K_{sspu}(V_{pu}) = \frac{V_{pu}}{I_{sspu}(0)}$$

Source: "BIRD 98 and ST 'Gate Modulation' Convergence", IBIS Open Forum Teleconference, Jan. 27th, 2007

The ratio modification Ksspd, Ksspu on Ku, Kd is only a function of V_{pd} or V_{pu}, it cannot reflect the effect of power rail voltage noise on switching edge timing change

New Behavior Model Proposal

- Modify Ku(t), Kd(t) as a function of **time averaged** power rail voltage Vcc(t); introduce correction coefficient B and A as a function of **time**

$$K_u(t) = K_{u0}(t) + B_u(t)\cdot\left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0}\right] + A_u(t)\left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0}\right]^2$$

$$K_d(t) = K_{d0}(t) + B_d(t)\cdot\left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0}\right] + A_d(t)\left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0}\right]^2$$

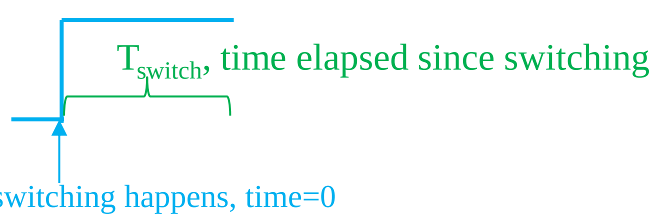
Source:

Ku, Kd under nominal Vcc (typical)

Linear fitting coefficient

Quadratic fitting coefficient

Averaged Vcc(t) after the switching event happens;

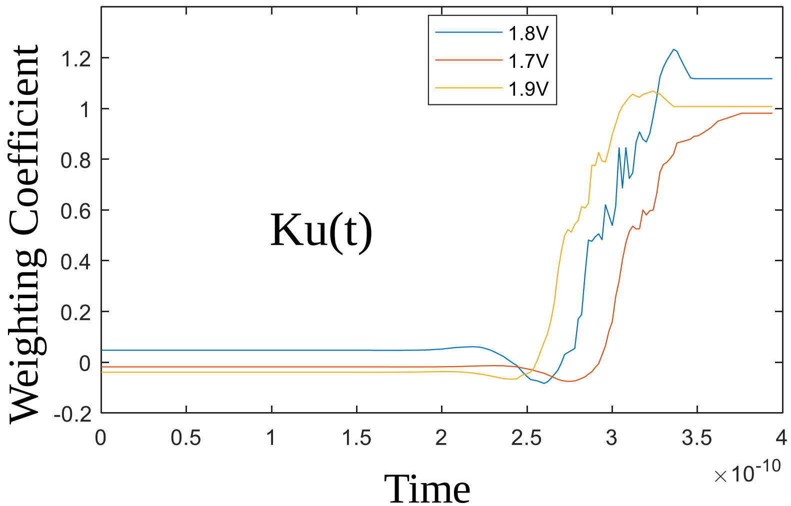


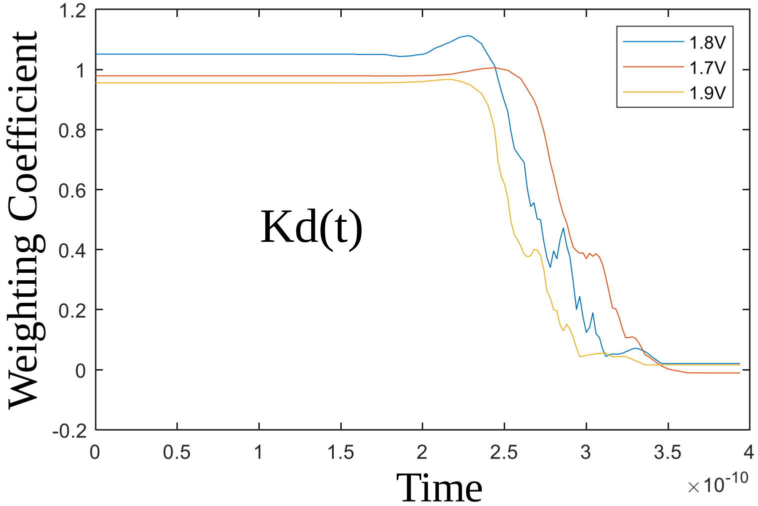
Previous method on modification of Ku, Kd does not consider the time averaged effect;

Source: Behavioral modeling of jitter due to power supply noise for input/output buffers (US Patent 9842177B1)

New Behavior Model Proposal

- How the modified Ku(t), Kd(t) account for Vcc(t) caused delay change





$$K_u(t) = K_{u0}(t) + B_u(t) \cdot \left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0} \right] + A_u(t) \left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0} \right]^2$$

$$K_d(t) = K_{d0}(t) + B_d(t) \cdot \left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0} \right] + A_d(t) \left[\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}} - V_{cc0} \right]^2$$

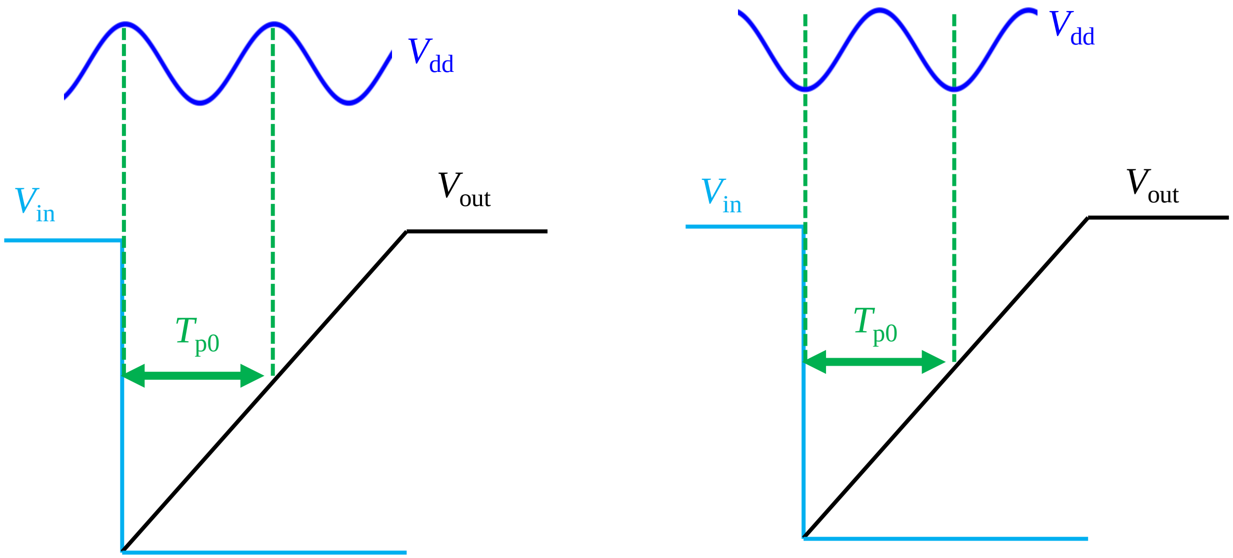
1. At each time point, use Ku, Kd under three cases => B(t), A(t);

2. B(t), A(t) can account for the delay change due to Vcc(t) noise;

3. The total effect of Vcc(t) during the time range of propagation delay is considered by the time-averaged Vcc(t)

New Behavior Model Proposal

- Why consider **time averaged** power rail voltage effect



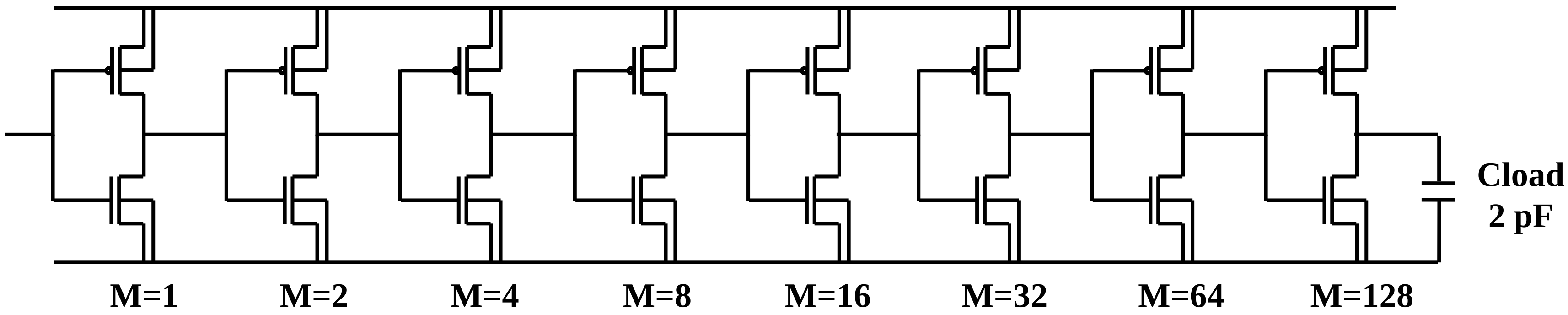
Propagation delay will be the same for the two cases

1. The Vcc noise can take effect during the propagation delay time range;

2. The influence is accumulated, just consider instantaneous voltage value is not accurate.

Model Validation

- Tested driver

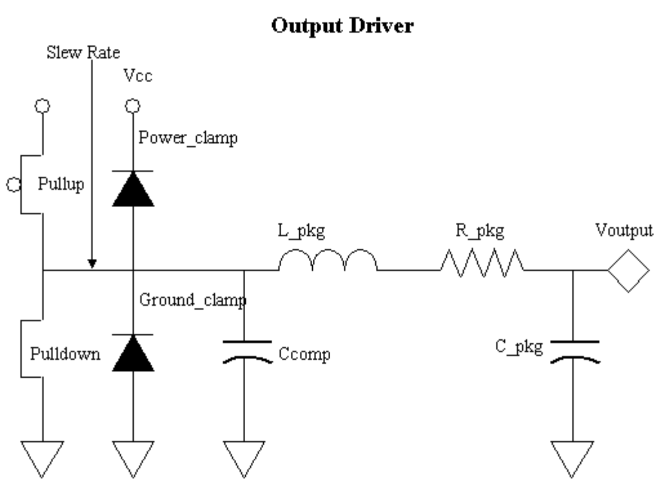


< All NMOS>
nch_tn
W = 1 um
L = 180 nm

< All PMOS>
pch_tn
W = 2 um
L = 180 nm

180nm technology, nominal voltage 1.8V

- Corresponding IBIS model (output)

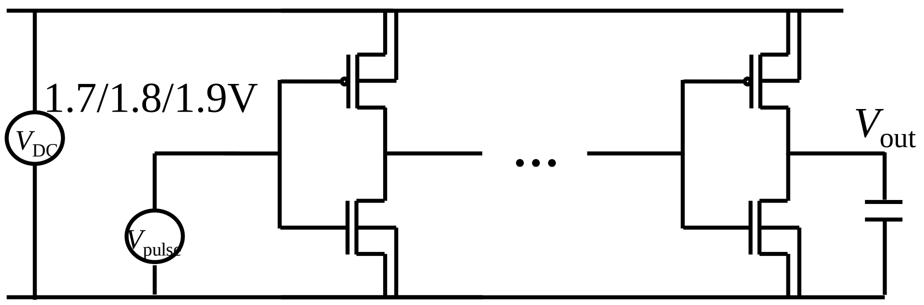


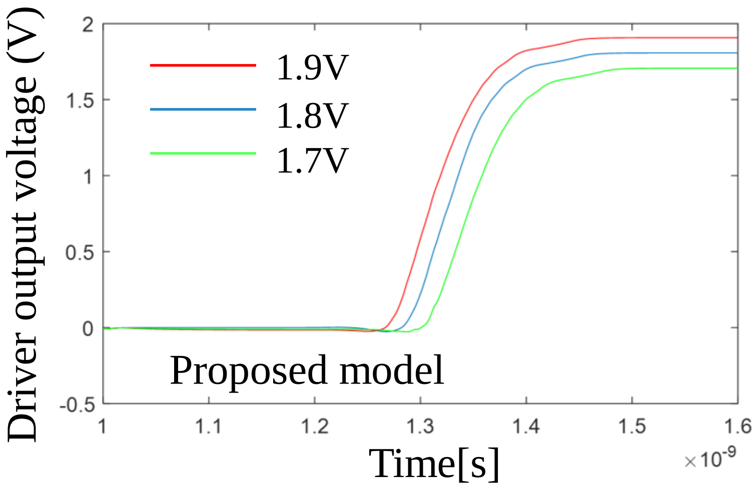
In this case, there is no power_clamp and ground_clamp;
C_comp is extracted as 0.46pF;

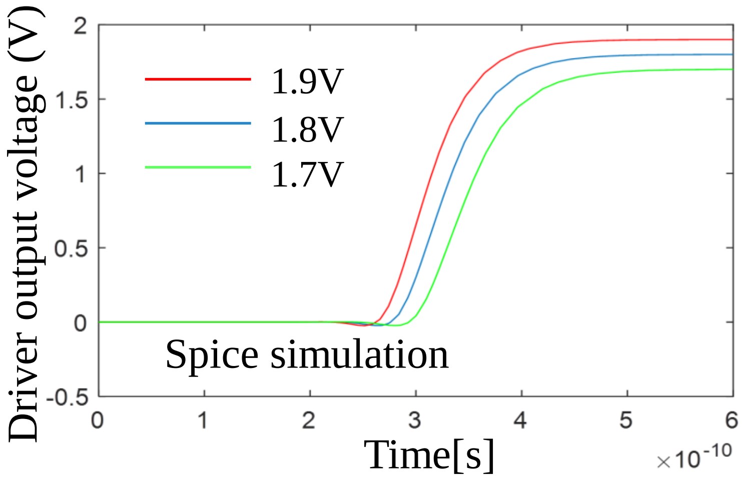
The Ku, Kd is implemented with the new method.

Model Validation

1. Vcc 1.7/1.8/1.9V respectively

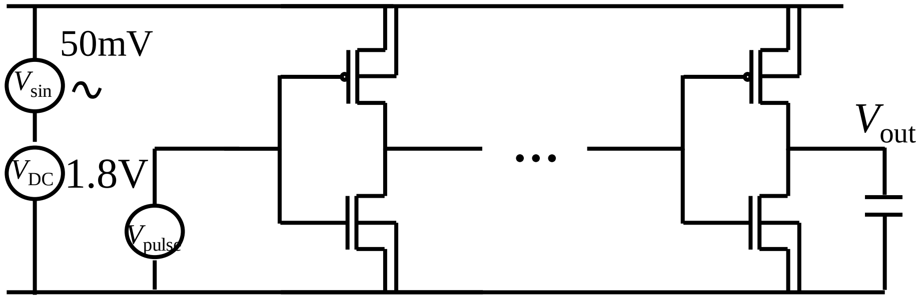






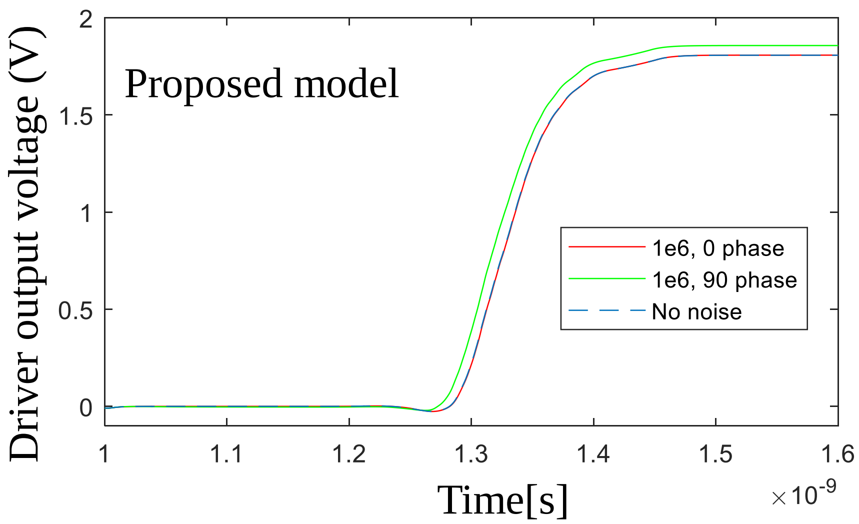
Model Validation

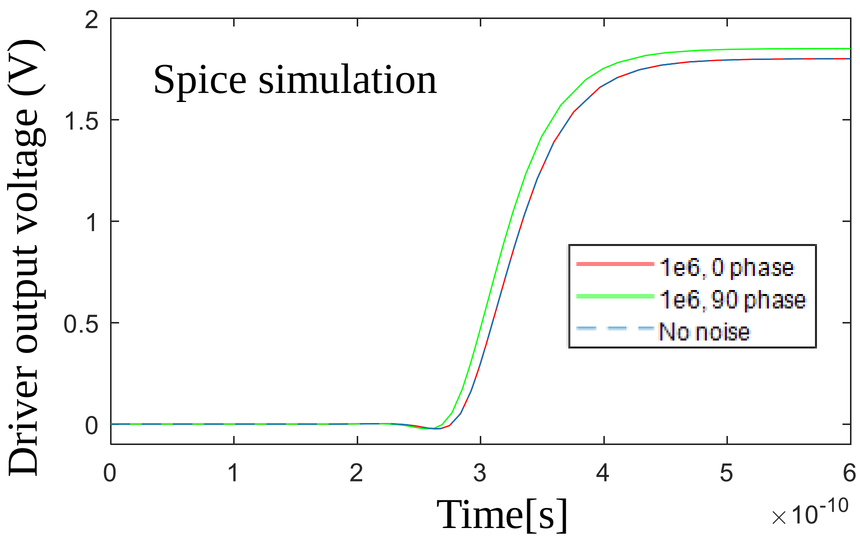
2. Vcc have very low frequency noise



Vcc=1.8V+0.05*sin(2*pi*1e6)

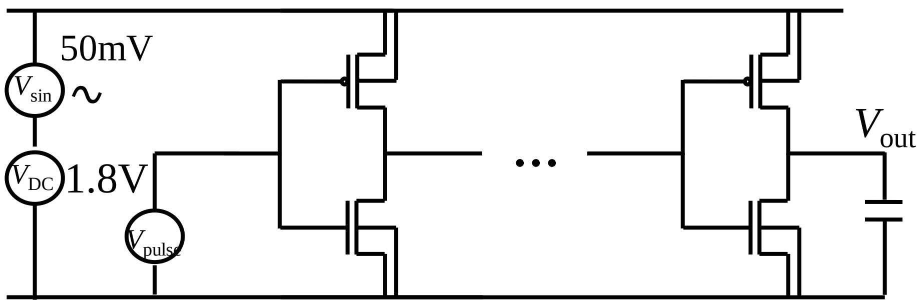
Vcc=1.8V+0.05*sin(2*pi*1e6+pi/2)





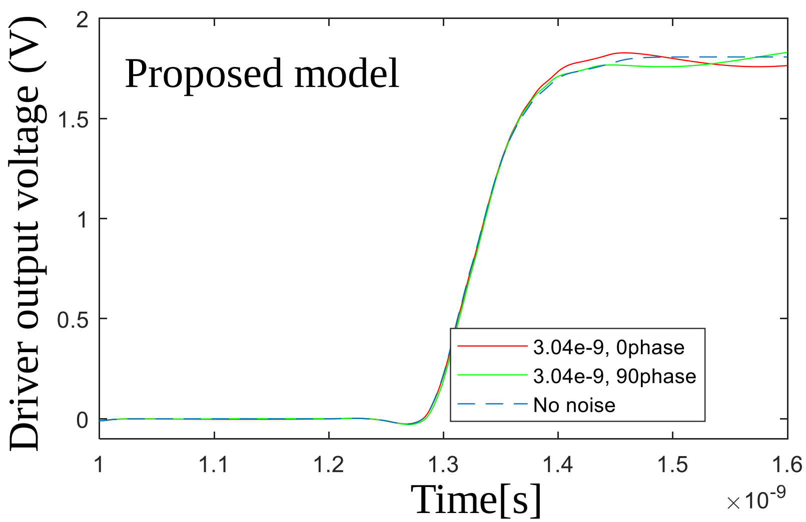
Model Validation

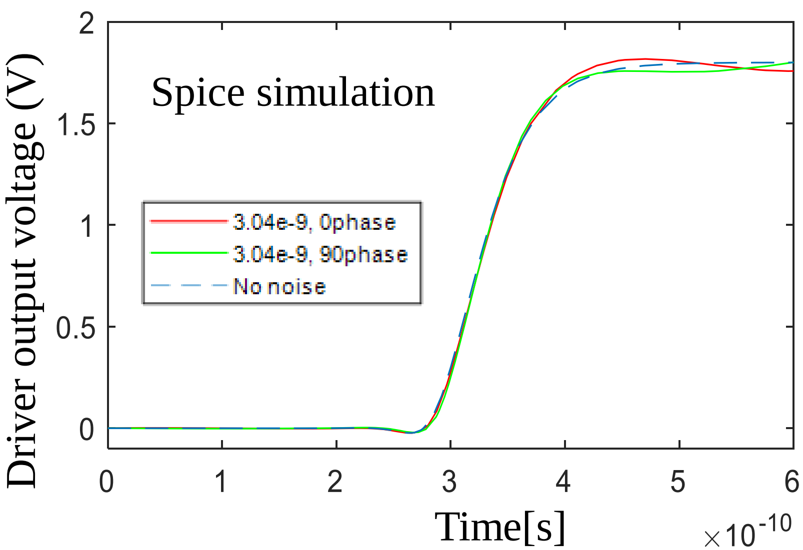
3. Vcc have noise with frequency corresponds to propagation delay (329ps)



Vcc=1.8V+0.05*sin(2*pi*3.04e9)

Vcc=1.8V+0.05*sin(2*pi*3.04e9+pi/2)





Implementation of New Behavior Model Proposal

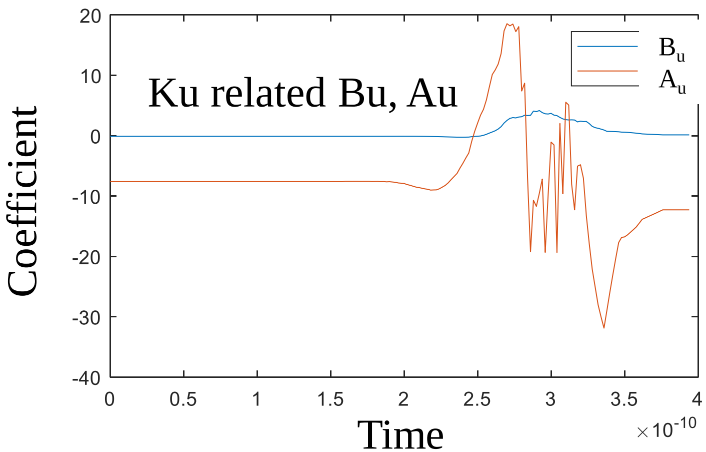
- Extraction of Bu(t), Au(t), Bd(t) and Ad(t) from Ku(t), Kd(t) under different Vcc
  - 1.Extraction of Ku (t) and Kd (t) for three voltage cases
  - 2. Bu(t), Au(t), Bd(t), Ad(t) extractions

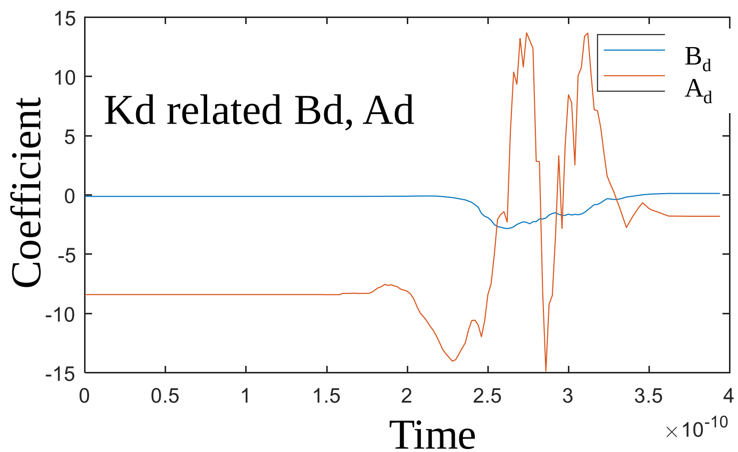
$K_u(V_{cc\_max},t) = K_u(V_{cc0},t) + B_u(t)*(V_{cc\_max}-V_{cc0}) + A_u(t)*(V_{cc\_max}-V_{cc0})^2$
$K_u(V_{cc\_min},t) = K_u(V_{cc0},t) + B_u(t)*(V_{cc\_min}-V_{cc0}) + A_u(t)*(V_{cc\_min}-V_{cc0})^2$

➡ 2 equations, 2 unknowns algorithm => Bu(t), Au(t)

V_{cc_max} 1.9V;
V_{cc_min} 1.7V;
V_{cc0} 1.8V

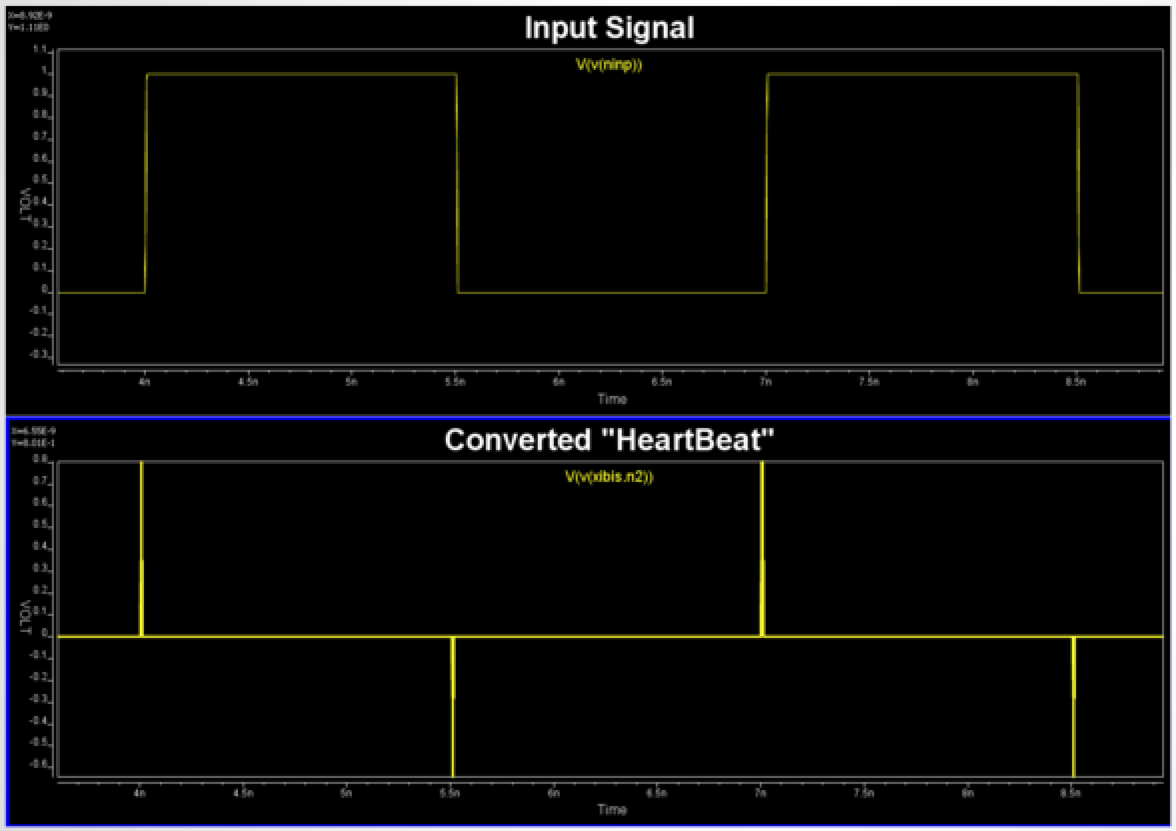
Bd(t), Ad(t) can be obtained similarly





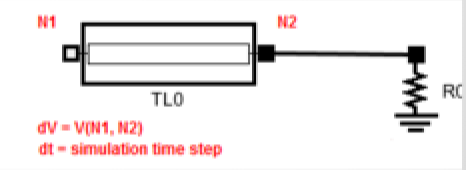
Implementation of New Behavior Model Proposal

- Implementation in Ngspice (Modify based on current ibis2spice algorithm)
    1. Ku, Kd, Bu, Au, Bd, Ad calculated offline from rising/falling waveforms
    2. From input switching edge dv/dt, judging rising or falling



Source:
http://www.spisim.com/blog/ibis2spice_p1/
http://www.spisim.com/blog/ibis2spice_p2/

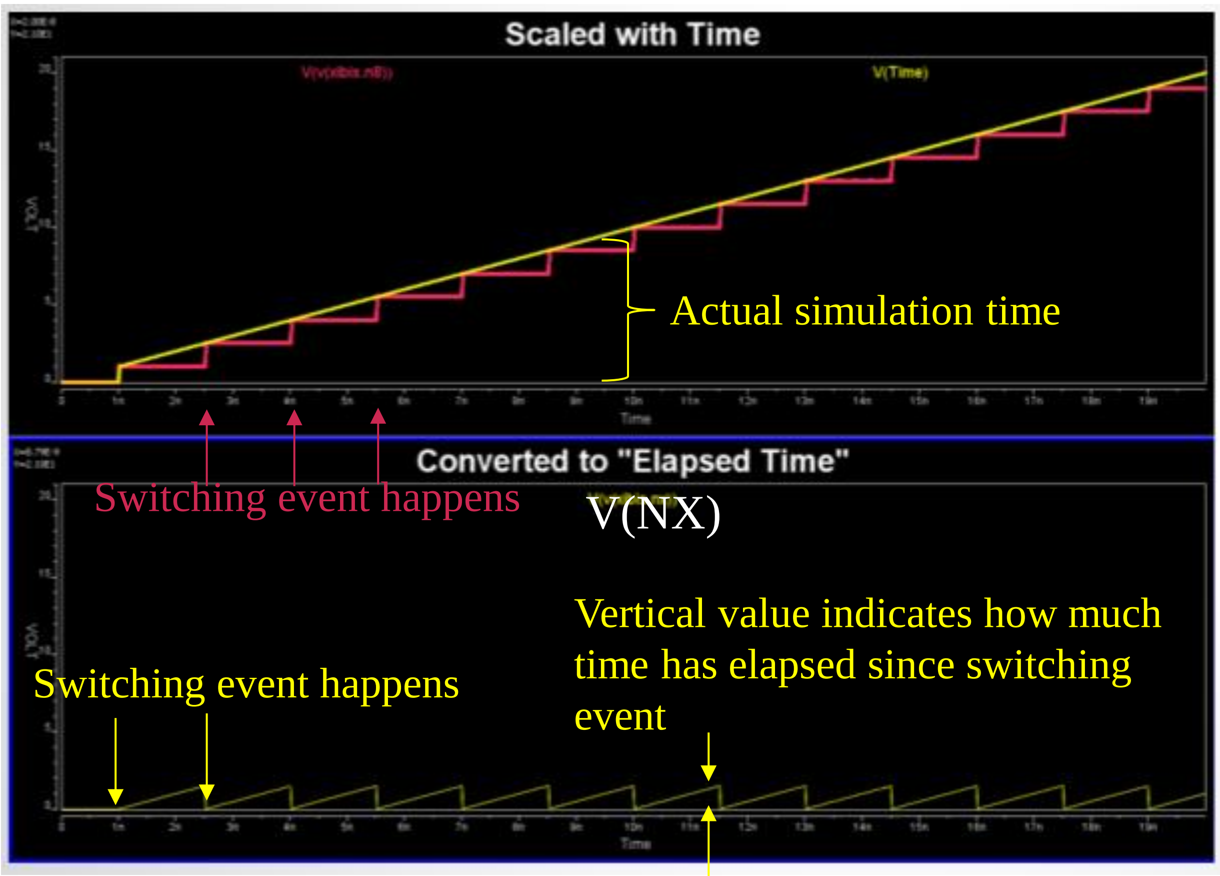
Use a transmission line to realize the differentiation



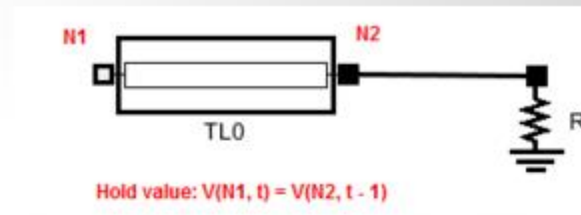
Implementation of New Behavior Model Proposal

- Implementation in Ngspice (Modify based on current ibis2spice algorithm)

  3. Record elapsed time since every switching event



Source:
http://www.spisim.com/blog/ibis2spice_p1/
http://www.spisim.com/blog/ibis2spice_p2/



The level hold (latch) realized with an ideal transmission line

t - value hold

Implementation of New Behavior Model Proposal

- Implementation in Ngspice (Modify based on current ibis2spice algorithm)

4. Implement the time averaged Vcc (**<u>Improved algorithm in this work, a practical implementation in open-source Ngspice</u>**)

```
* INPUT CONTROL
BN NINX 0 V= ((V(NINP) > 0.5) & (V(NENB) > 0.5))? 1.0 : 0.0

* CONTROL LOGIC
BI NI 0 V=(V(NINX) - 0.5)
B2 N2 0 V=V(NI, N9) * 8
B3 N3 0 V=abs(V(N2))
B4 N4 0 V=(V(N3) > 0.5)? 1 : -1
B5 N5 0 V=V(N4) > 0? TIME * 1E9: 0
B6 N6 0 V=V(N4) > 0? V(N5) : V(N8)
B7 NX 0 V=(V(N6) >= 1.0)? TIME * 1E9 - V(N8) : 0.0
B8 NT1 0 V=(V(NX) > 0.01)? (V(NVCC)*0.001-1.8*0.001+V(NTD)) : 0.0
B9 NT 0 V=(V(NX) > 0.01)? V(NT1)/V(NX) : 0.0

* DELAY ELEMENT: Td value must match time-step
T1 N6 0 N8 0 Z0=50 Td=1p
T2 NI 0 N9 0 Z0=50 Td=1p
T3 NT1 0 NTD 0 Z0=50 Td=1p
R1 N8 0 50
R2 N9 0 50
R3 NTD 0 50
```

Vcc

Vcc0

V(NT1) store the summation of Vcc voltage since start of switching

Realized by:
Vcc-Vcc0+V(NTD)

V(NX) time elapsed since the switching

V(NT) is the time averaged Vcc

$$\frac{\int_0^{T_{switch}} V_{cc}(t)}{T_{switch}}$$

Implementation of Improved Behavior Model Proposal

- Implementation in Ngspice (Modify based on current ibis2spice algorithm)

  5. Implement the modified Ku, Kd as B source (**<u>Improved algorithm in this work, a practical implementation in open-source Ngspice</u>**)

V(1,2) is V(NX), time elapsed since switching event;
V(3,4) is the Ku or Kd value
V(5) is the time averaged Vcc

Original Ku implementation: Ku0(t)

```
* KU COEF RISE
.SUBCKT driver_TYP_KU_R 3 4 1 2
B1 3 4 V =
+ (V(1,2) < 0.000000E0)? 0.000000E0:
+ (V(1,2) < 3.622352E-3)? 1.287944E1 * V(1,2) + 0.000000E0:
+ (V(1,2) < 7.244704E-3)? -7.295161E-5 * V(1,2) + 4.665411E-2:
```

Modified Ku implementation

```
* KU COEF RISE
.SUBCKT driver_TYP_KU_R 3 4 1 2 5
B1 3 4 V =
+ (V(1,2) < 0.000000E0)? 0.000000E0:
+ (V(1,2) < 0.0036223520000000)? 12.8794400000000007 * V(1,2) + 0.0000000000000000
+ (V(1,2) < 0.0072447040000000)? -0.0000729516100000 * V(1,2) + 0.0466541100000000
```

Ku0(t)

```
+ (0.0000251111959497 * V(1,2) + -0.1034584500000000) * V(5) + (0.0039680452540881 * V(1,2) + -7.5443674999999999) * V(5) * V(5) :
+ (0.0002022823036612 * V(1,2) + -0.1034590917761164) * V(5) + (0.0091645843101826 * V(1,2) + -7.5443863236936428) * V(5) * V(5) :
```

Bu(t) Au(t)

5.1 Bxxxx: Nonlinear dependent source (ASRC)

B source in Ngspice to store tabulated data

5.1.1 Syntax and usage

General form:

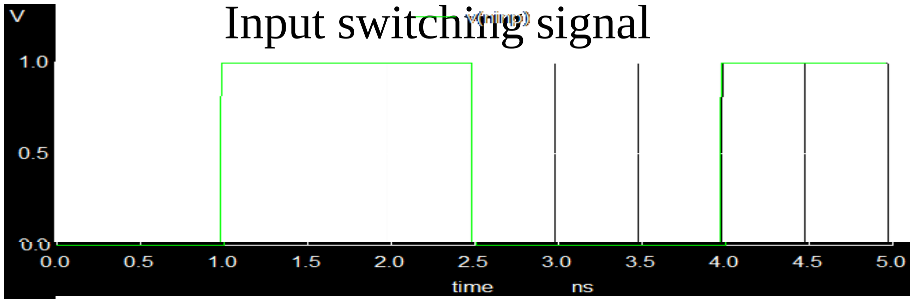
```
BXXXXXXX n+ n- <i=expr> <v=expr> <tc1=value> <tc2=value>
+ <temp=value> <dtemp=value>
```

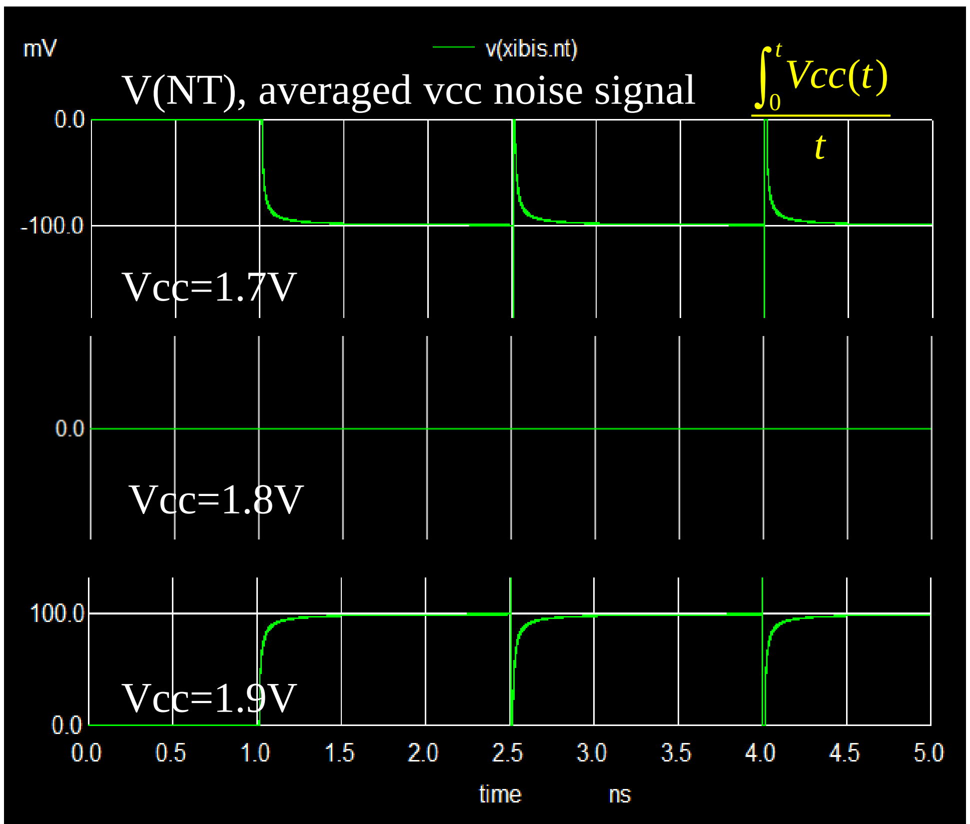
Examples:

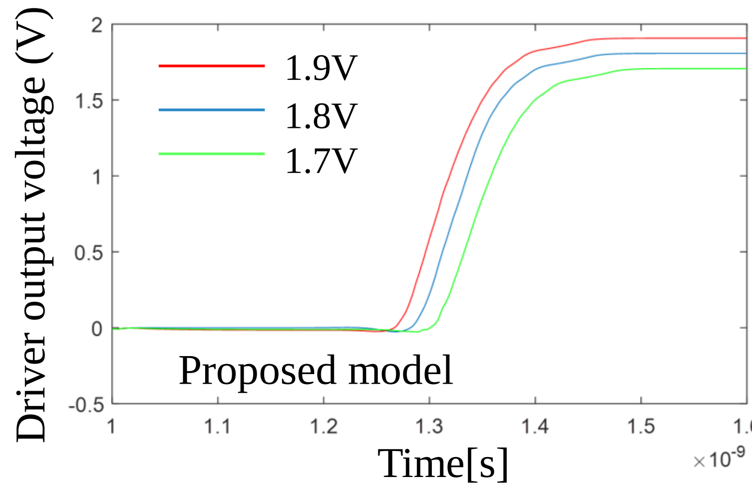
```
B1 0 1 I=cos(v(1))+sin(v(2))
B2 0 1 V=ln(cos(log(v(1,2)^2)))-v(3)^4+v(2)^v(1)
B3 3 4 I=17
B4 3 4 V=exp(pi^i(vdd))
B5 2 0 V = V(1) < {Vlow} ? {Vlow} : V(1) > {Vhigh} ? {Vhigh} : V(1)
```

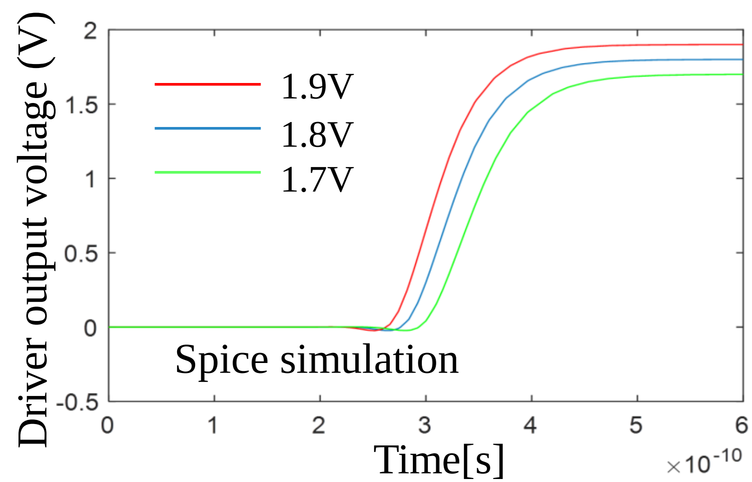
Simulation Results of Implemented Time-Averaged Vcc[V(NT)]

1. Vcc 1.7/1.8/1.9V respectively





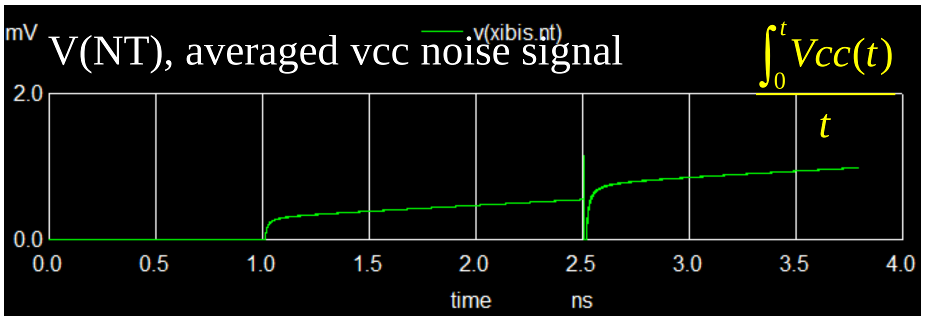


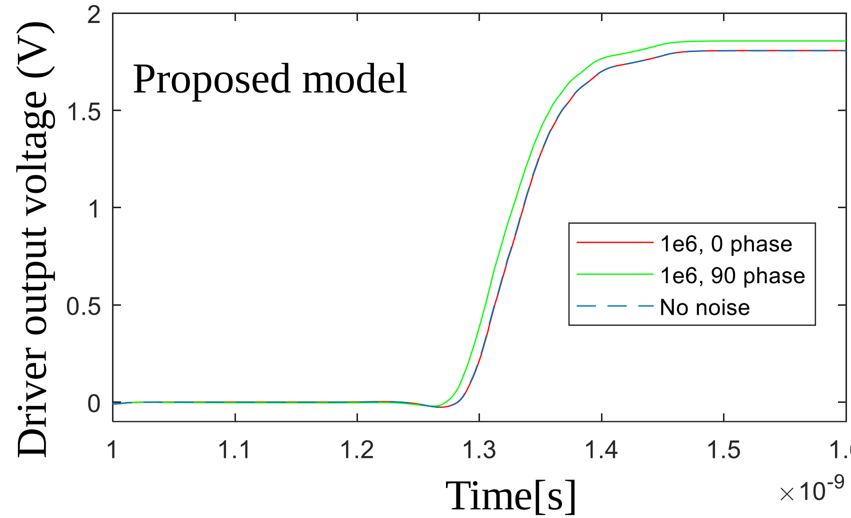


Simulation Results of Implemented Time-Averaged Vcc[V(NT)]

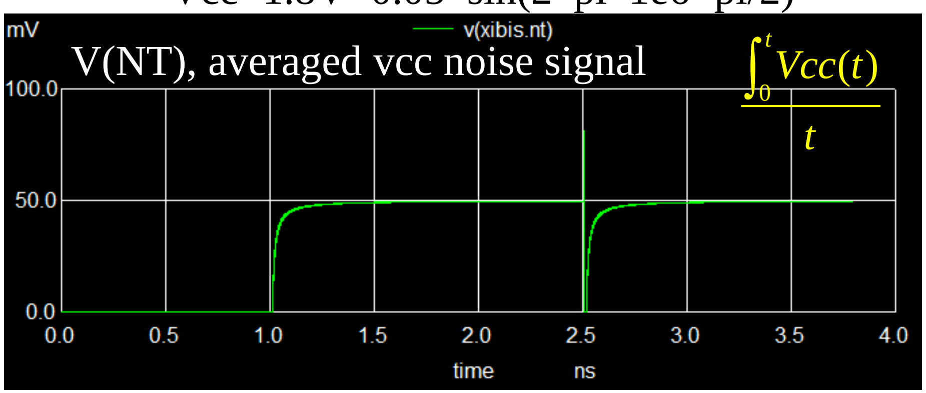
2. Vcc have very low frequency noise

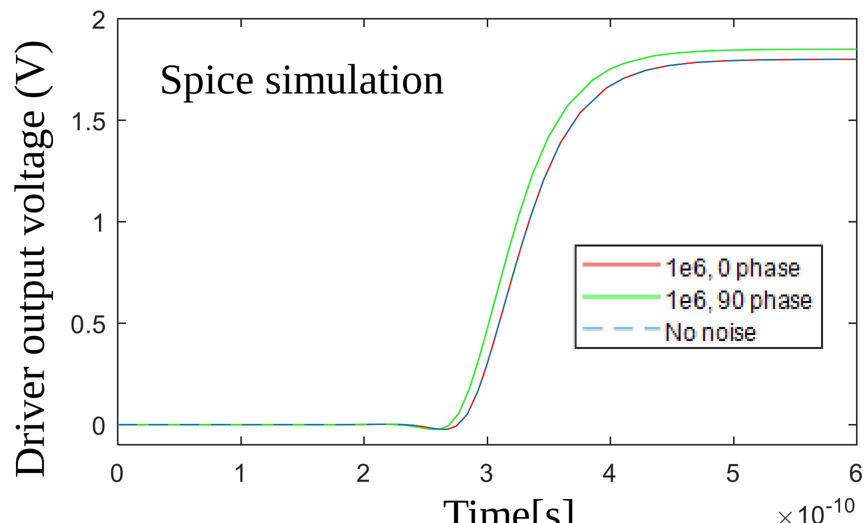
Vcc=1.8V+0.05*sin(2*pi*1e6)





Vcc=1.8V+0.05*sin(2*pi*1e6+pi/2)

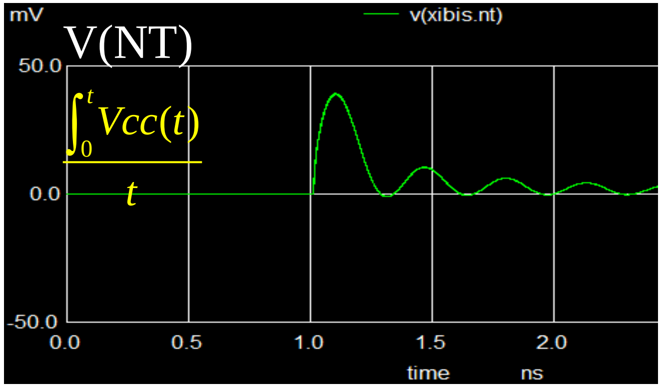


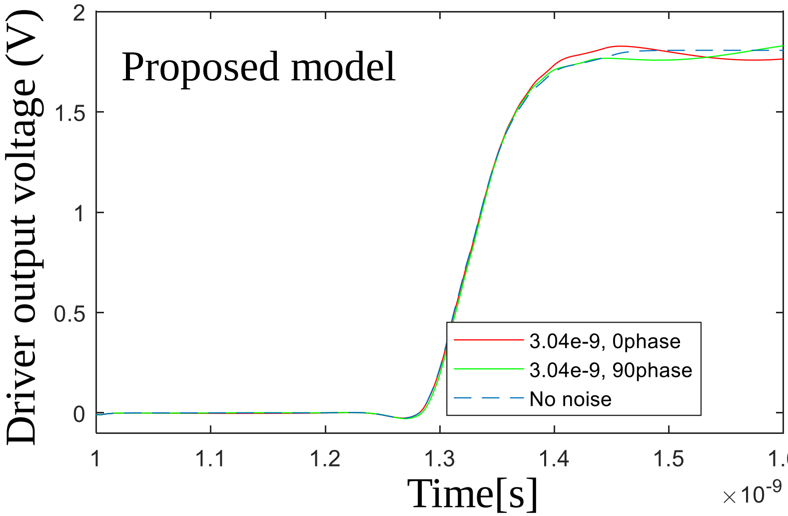


Simulation Results of Implemented Time-Averaged Vcc[V(NT)]

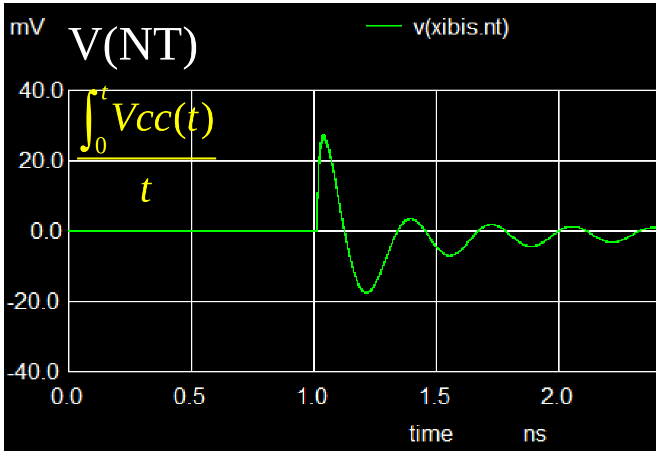
3. Vcc have noise with frequency corresponds to propagation delay (329ps)

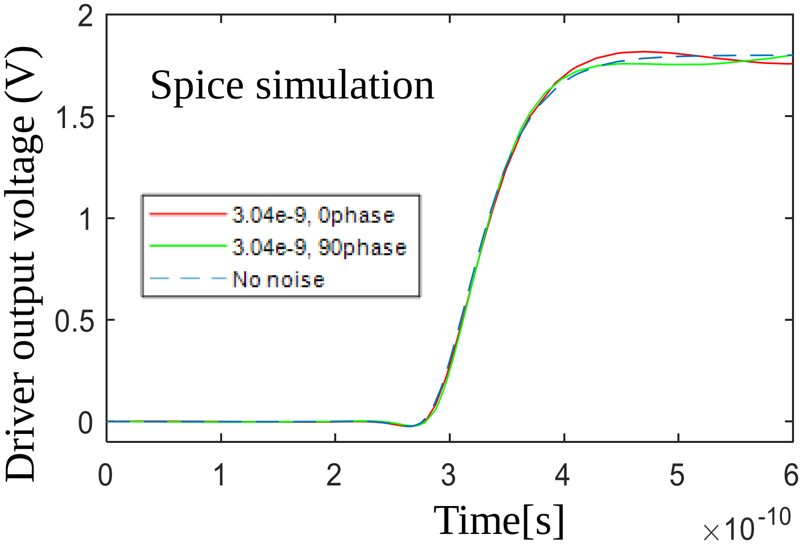
Vcc=1.8V+0.05*sin(2*pi*3.04e9)





Vcc=1.8V+0.05*sin(2*pi*3.04e9+pi/2)





Conclusions

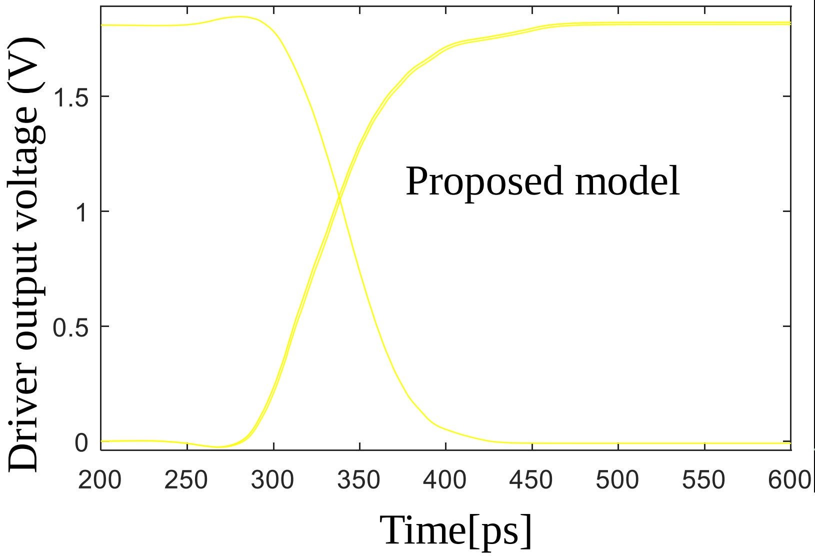
- The ability to correctly account for the power supply induced jitter has been improved
- This work has extended the current IBIS model to include the delay change effect caused by the power rail noise
  - The Ku, Kd are modified as a function of Vcc
  - The time averaged effect of Vcc has been considered
  - A plausible algorithm has been provided and implemented in Ngspice
  - This method is suitable for small power noise situation

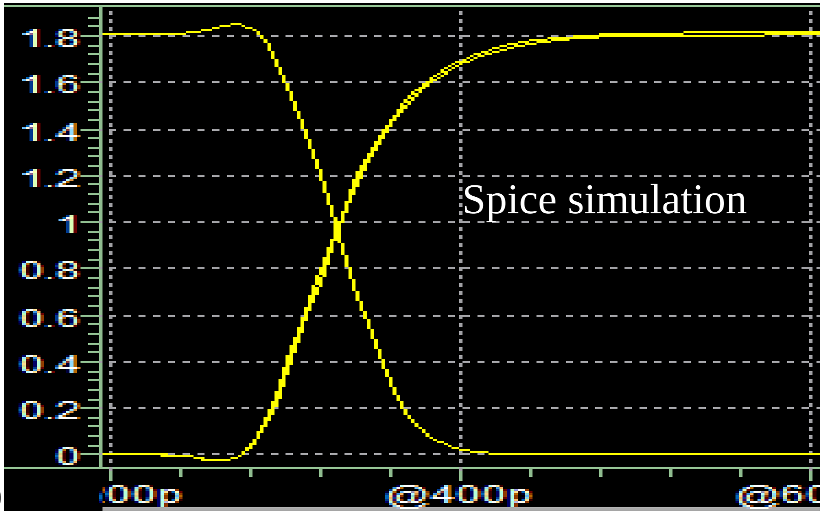
Back-Up, Partial Eye Diagram

For the current implementation, the Ngspice can only run for a short period of time (possibly due to the not perfect implementation of the algorithm as spice sub-circuit);

Usually, 2 rising edges and 1 falling edge can be obtained with the improved IBIS model

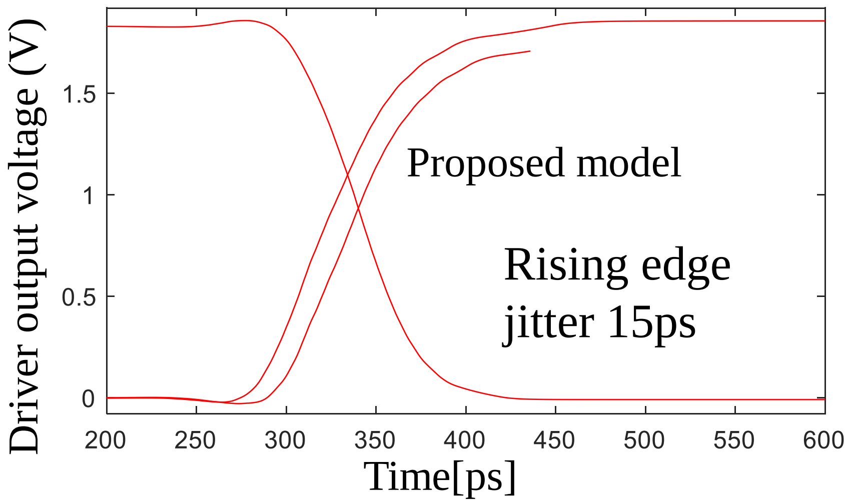
1. Vcc=1.8V+0.05*sin(2*pi*10e6)

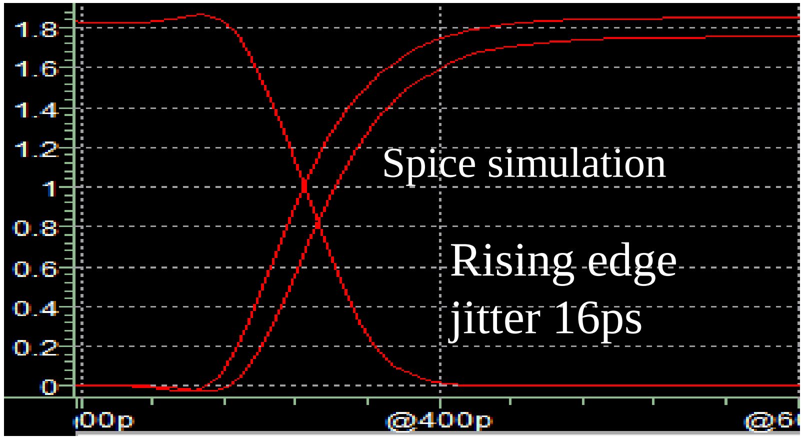




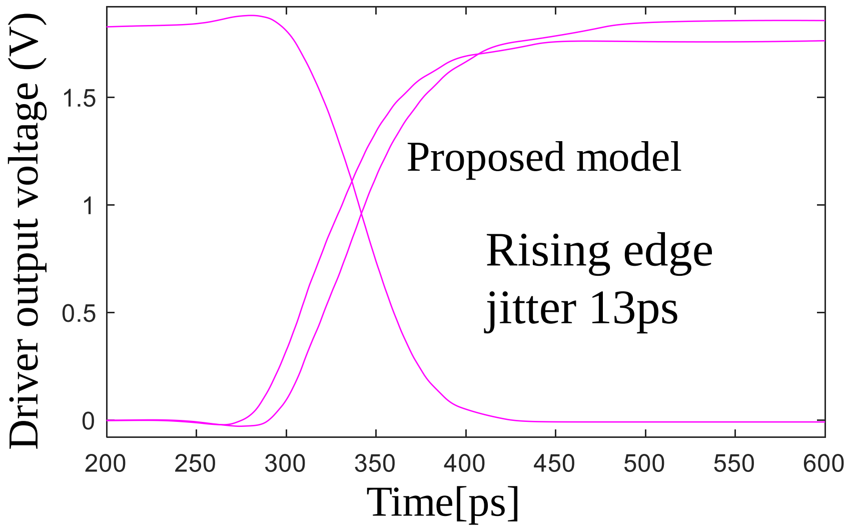
Back-Up, Partial Eye Diagram

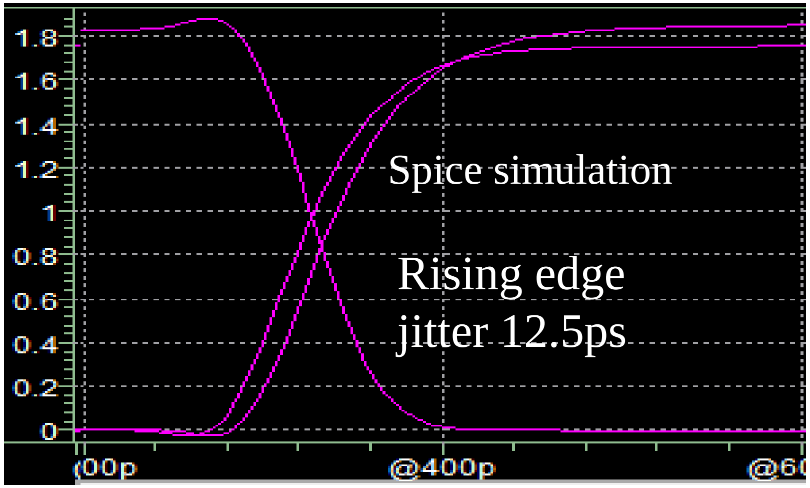
2. Vcc=1.8V+0.05*sin(2*pi*150e6)





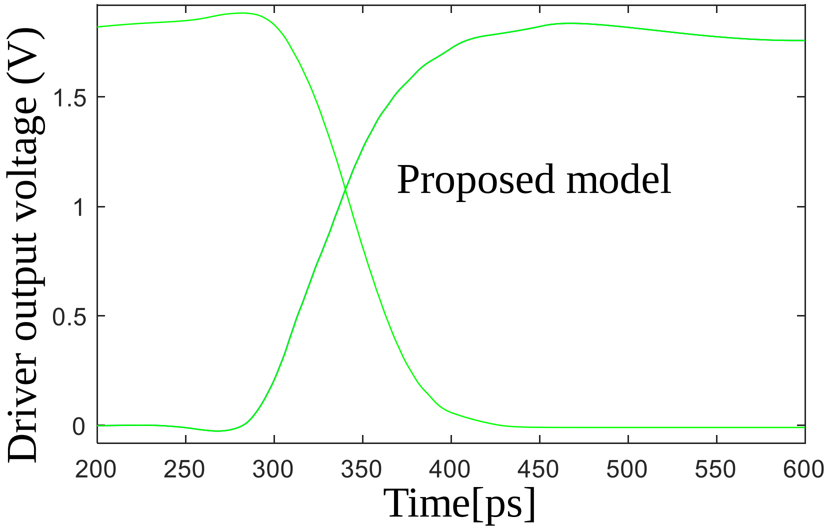
3. Vcc=1.8V+0.05*sin(2*pi*1150e6)

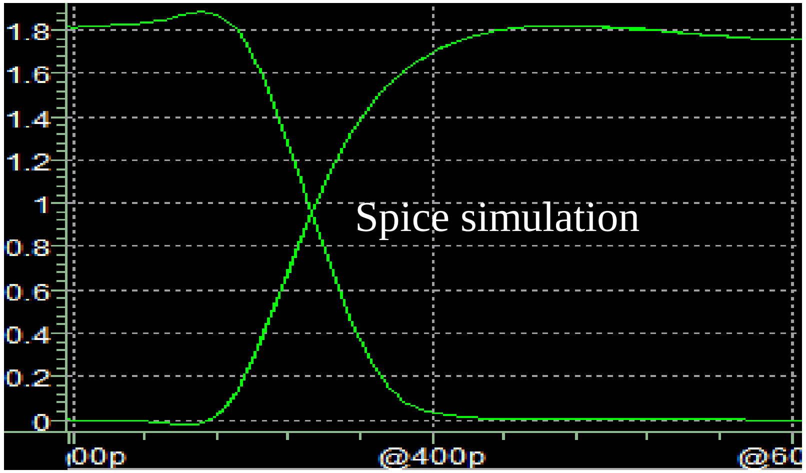




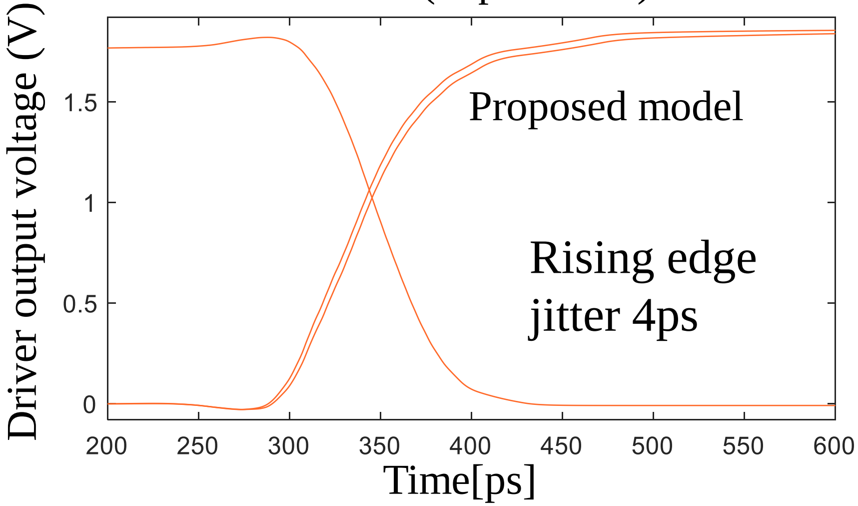
Back-Up, Partial Eye Diagram

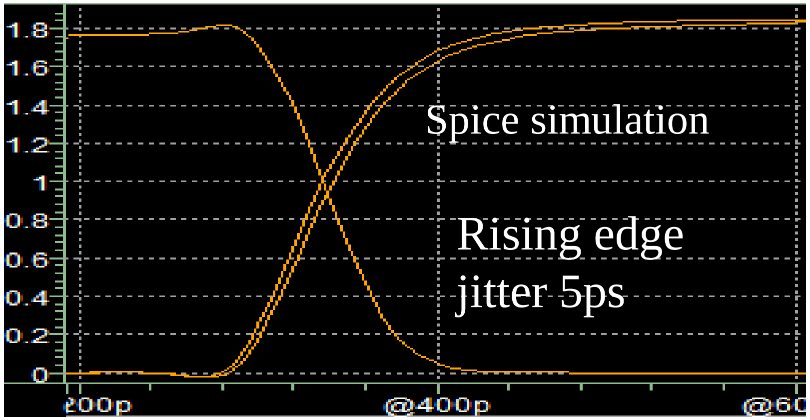
4. Vcc=1.8V+0.05*sin(2*pi*3e9)





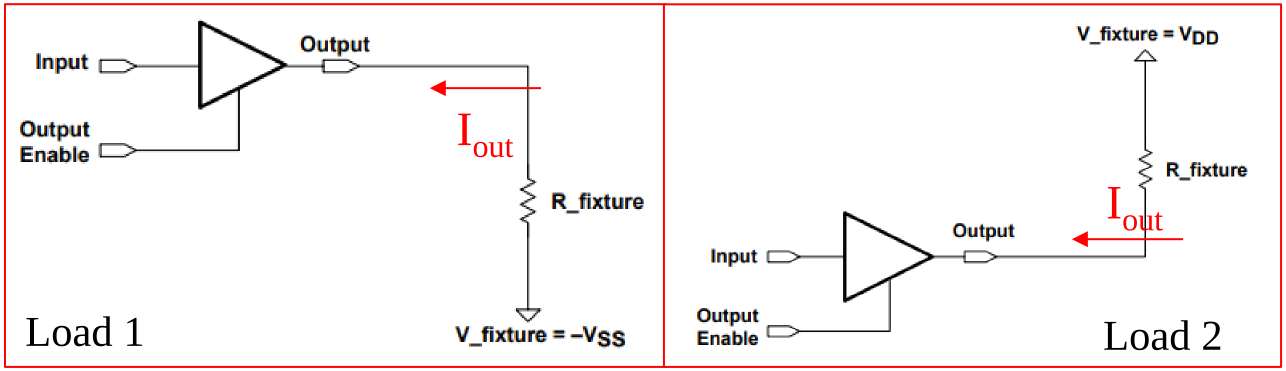
5. Vcc=1.8V+0.05*sin(2*pi*700e6)





Back-Up, Ku, Kd Extraction

- Extraction of Bu(t), Au(t), Bd(t) and Ad(t) from Ku(t), Kd(t) under different Vcc
  - Ku(t), Kd(t) extractions



Driver output current

$I_{out}=K_u*I_u+K_d*I_d$

From pull up/down I-V table

$I_u(V)$, $I_d(V)$;

$K_u(t)*I_u(V_1)+K_d(t)*I_d(V_1)=I_{out}(V_1)$

$K_u(t)*I_u(V_2)+K_d(t)*I_d(V_2)=I_{out}(V_2)$

⬇

2 equations, 2 unknowns

algorithm => Ku(t), Kd(t)

For Vcc=1.8/1.7/1.9V (typ/min/max)