From Electrical Measurement Data To Compact Device Models

SAND2019-9190C

Karthik V. Aadithya
Org. 1355

Paul A. Kuberry
Org. 1442

Eric R. Keiter
Org. 1355

Pavel B. Bochev
Org. 1442

Photo

Alan A. Mar
Org. 5445

Ting Mei
Org. 1355

Biliana S. Paskaleva
Org. 1356

Kenneth M. Leeson
Org. 1356

Machine Learning and Deep Learning Conference 2019

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Overview of this talk

Part 1.

Circuit design basics.

Overview of this talk

Part 1.

Circuit design basics.

Part 2.

Compact models: what they are, why they're needed.

Overview of this talk

Part 1.
Circuit design basics.

Part 2.
Compact models: what they are, why they're needed.

Part 3.
Developing compact models is hard.

Overview of this talk

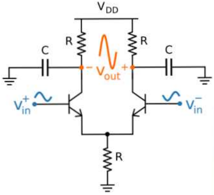
Part 1.
Circuit design basics.

Part 2.
Compact models: what they are, why they're needed.

Part 3.
Developing compact models is hard.

Part 4.
Our solution: use machine learning to create compact models.

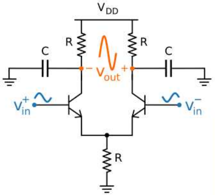
Lego-style circuit design

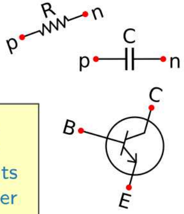


Problem

Designing circuits (*e.g.*, amplifiers) that meet Sandia's stringent specs

Lego-style circuit design



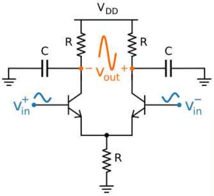


Problem

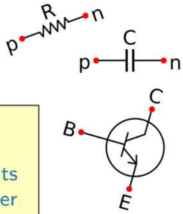
Designing circuits (*e.g.*, amplifiers) that meet Sandia's stringent specs

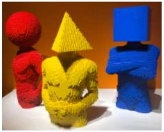
Design philosophy

Build complex circuits by connecting together simpler devices (*e.g.*, diodes, BJTs)



Lego-style circuit design





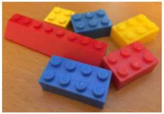
Problem

Designing circuits (*e.g.*, amplifiers) that meet Sandia's stringent specs

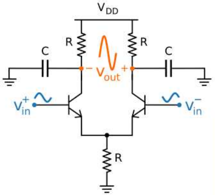
←

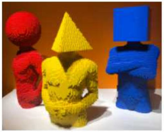
Design philosophy

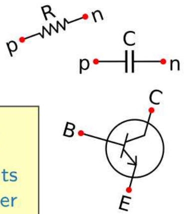
Build complex circuits by connecting together simpler devices (*e.g.*, diodes, BJTs)

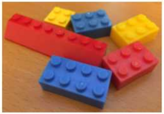


Lego-style circuit design









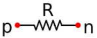
Problem

Designing circuits (*e.g.*, amplifiers) that meet Sandia's stringent specs

Design philosophy

Build complex circuits by connecting together simpler devices (*e.g.*, diodes, BJTs)

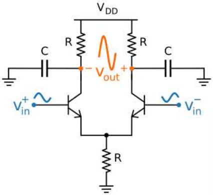
Each device performs a specific function

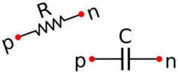


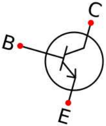
Resistor

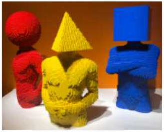
Impediment. Limit current in a circuit branch.

Lego-style circuit design







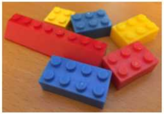


Problem

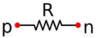
Designing circuits (*e.g.*, amplifiers) that meet Sandia's stringent specs

Design philosophy

Build complex circuits by connecting together simpler devices (*e.g.*, diodes, BJTs)



Each device performs a specific function

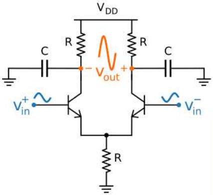


Resistor

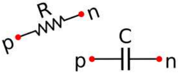
Impediment. Limit current in a circuit branch.

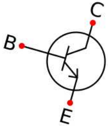
Capacitor

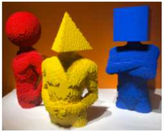
Reservoir. Store charge, convert it into current later.



Lego-style circuit design





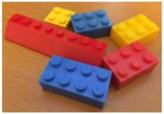


Problem

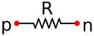
Designing circuits (*e.g.*, amplifiers) that meet Sandia's stringent specs

Design philosophy

Build complex circuits by connecting together simpler devices (*e.g.*, diodes, BJTs)



Each device performs a specific function

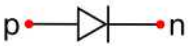


Resistor

Impediment. Limit current in a circuit branch.

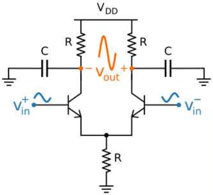
Capacitor

Reservoir. Store charge, convert it into current later.

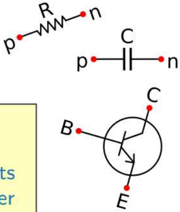


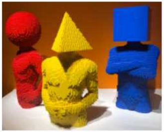
Diode

Gatekeeper. Allow current one way, block it the other way.



Lego-style circuit design



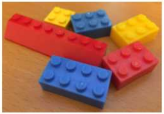


Problem

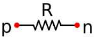
Designing circuits (*e.g.*, amplifiers) that meet Sandia's stringent specs

Design philosophy

Build complex circuits by connecting together simpler devices (*e.g.*, diodes, BJTs)

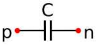


Each device performs a specific function



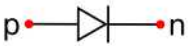
Resistor

Impediment. Limit current in a circuit branch.



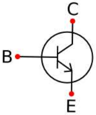
Capacitor

Reservoir. Store charge, convert it into current later.



Diode

Gatekeeper. Allow current one way, block it the other way.



BJT

Switch. When B is +ve, allow current between C and E; disallow when B is -ve.

Circuit design illustration: Snap Circuits

My cousin

His toy

Circuit design illustration: Snap Circuits

My cousin

Devices in Snap Circuits kit

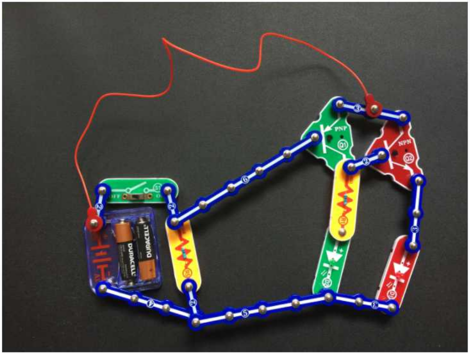
His toy

Circuit design illustration: Snap Circuits

My cousin

Devices in Snap Circuits kit

His toy



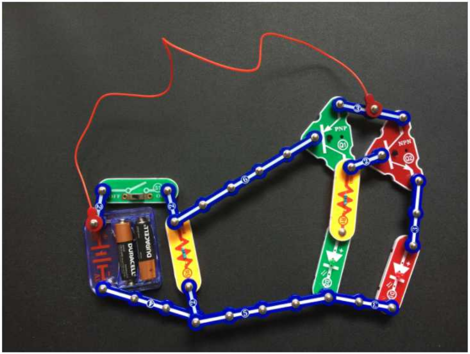
Movie: Circuit 1

Circuit design illustration: Snap Circuits

My cousin

His toy

Devices in Snap Circuits kit



Movie: Circuit 1 → Circuit 2

Modern circuit design = Heavily computer-aided

- Modern circuits are huge, cannot be built by hand

9.6B transistors, 14nm

2B transistors, 20nm

Modern circuit design = Heavily computer-aided

- Modern circuits are huge, cannot be built by hand

9.6B transistors, 14nm

2B transistors, 20nm

- Complex designs
  - Lots of inputs, outputs, and internal states
  - Digital, analog, and mixed-signal functionality
  - Wide range of operating environments

Modern circuit design = Heavily computer-aided

- Modern circuits are huge, cannot be built by hand

9.6B transistors, 14nm

2B transistors, 20nm

- Complex designs
  - Lots of inputs, outputs, and internal states
  - Digital, analog, and mixed-signal functionality
  - Wide range of operating environments

- EDA/CAD software tools used for every step in design cycle

Modern circuit design = Heavily computer-aided

- Modern circuits are huge, cannot be built by hand

9.6B transistors, 14nm

2B transistors, 20nm

- Complex designs
  - Lots of inputs, outputs, and internal states
  - Digital, analog, and mixed-signal functionality
  - Wide range of operating environments

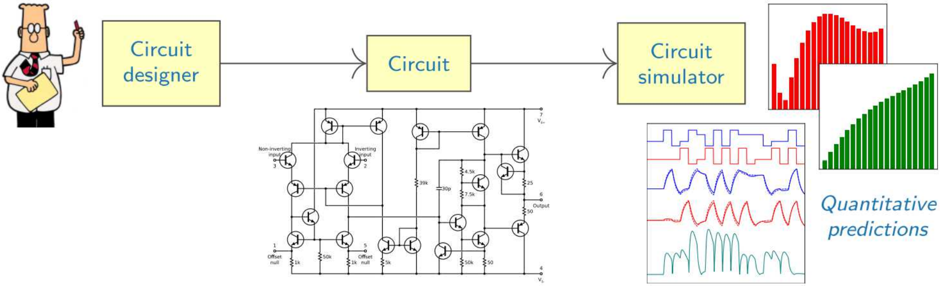
- EDA/CAD software tools used for every step in design cycle

- Extensive circuit simulation to validate, test designs pre-fab

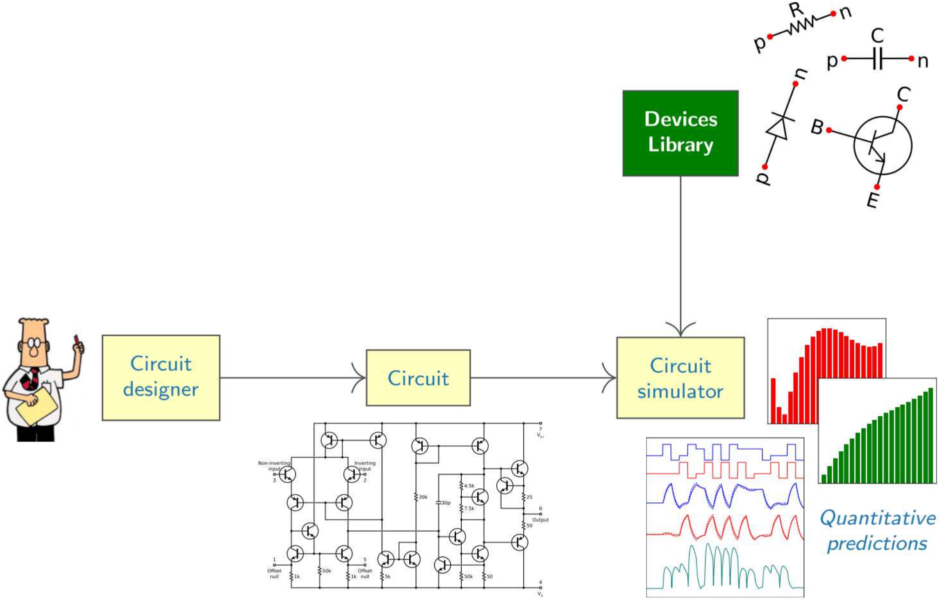
LTspice Xyce OrCAD PSpice

HSPICE SPICE OPUS Spyce MAPP

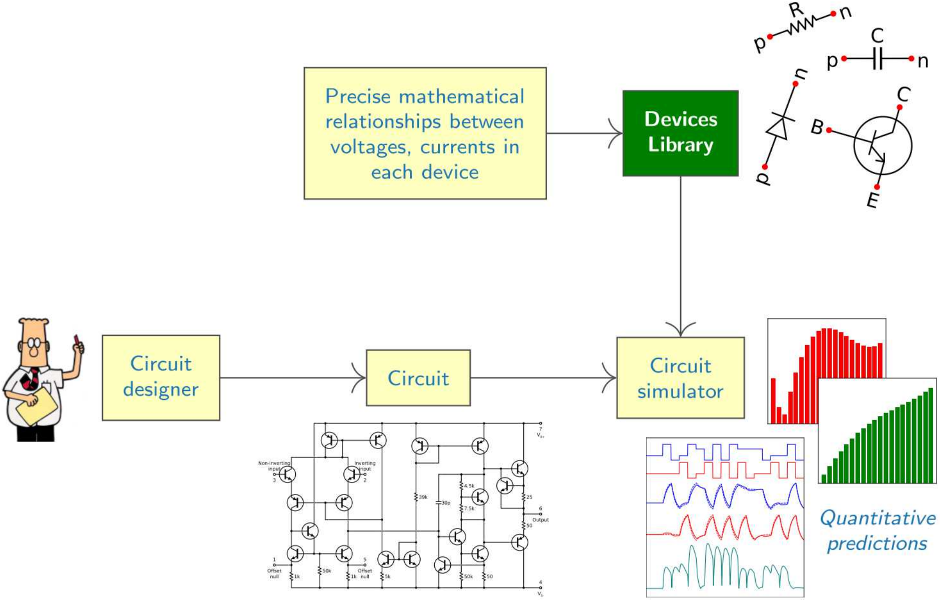
Circuit simulators need to be told how devices behave



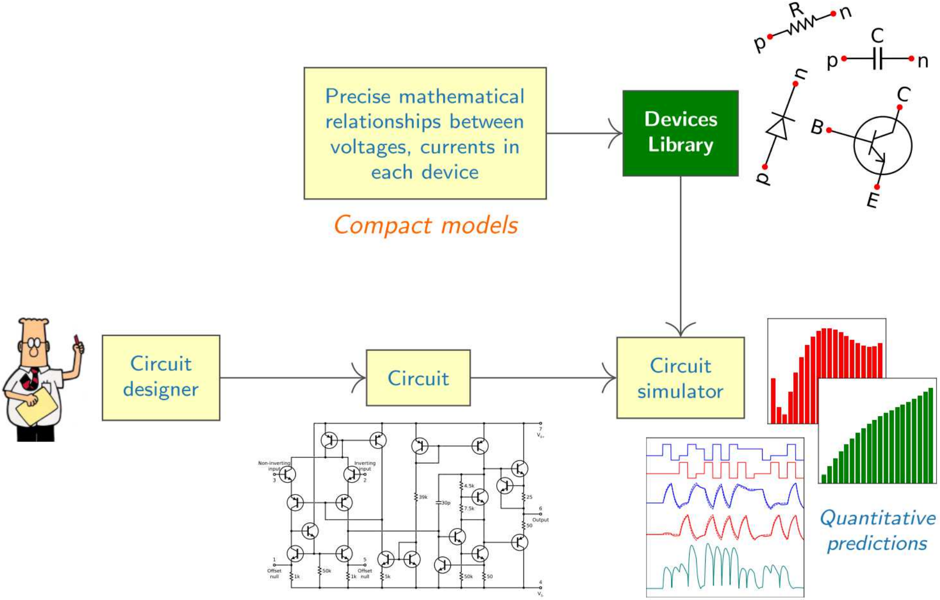
Circuit simulators need to be told how devices behave



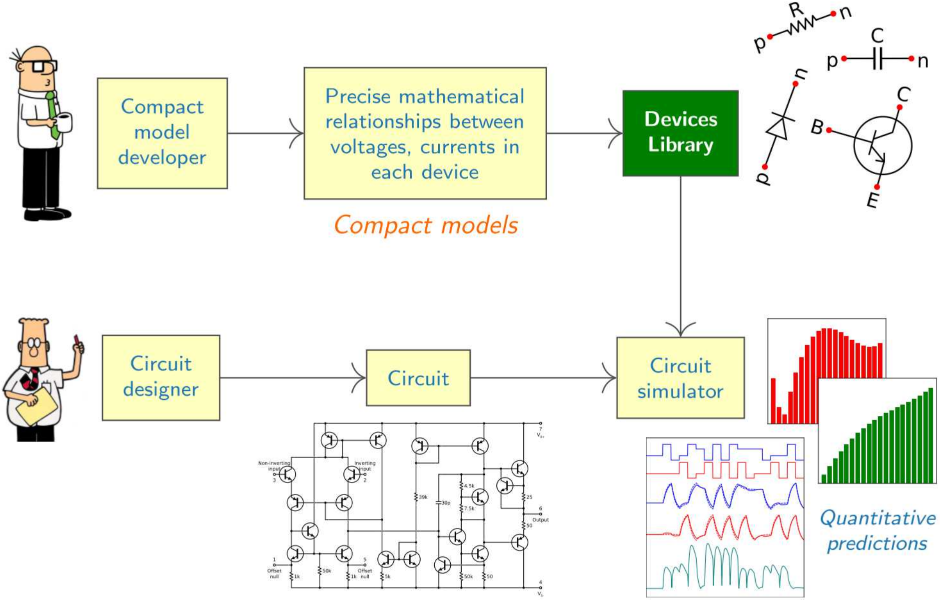
Circuit simulators need to be told how devices behave



Circuit simulators need to be told how devices behave

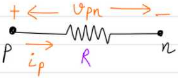


Circuit simulators need to be told how devices behave



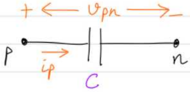
Compact models: some simple examples

Resistor



$$i_p = \frac{v_{pn}}{R}$$

Capacitor

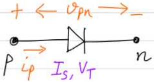


$$i_p = \frac{d}{dt} C v_{pn}$$

General Form

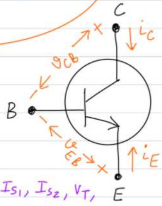
$$\frac{d}{dt}\vec{q}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right) + \vec{f}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right) = \vec{0}$$

Diode



$$i_p = I_s\left(e^{v_{pn}/V_T} - 1\right)$$

BJT



$I_{s_1}, I_{s_2}, V_T,$
α_F, α_R

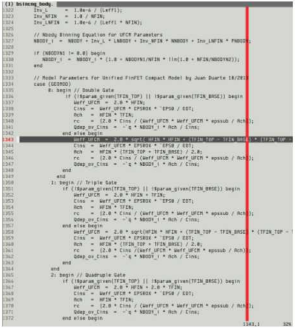
$$i_1 = I_{s_1}\left(e^{-v_{EB}/V_T} - 1\right)$$

$$i_2 = I_{s_2}\left(e^{-v_{CB}/V_T} - 1\right)$$

$$i_E = \alpha_R i_2 - i_1$$

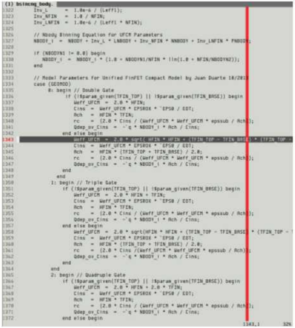
$$i_C = \alpha_F i_1 - i_2$$

“Real” compact models: the BSIM family



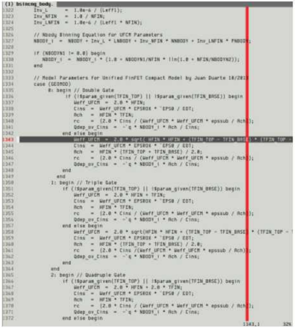
5,444 lines of code

“Real” compact models: the BSIM family



5,444 lines of code

“Real” compact models: the BSIM family



```
(1) bsimcmg_body.
Inv_L     =  1.0e-6 / (Leff1);
Inv_NFIN  =  1.0 / NFIN;
Inv_LNFIN =  1.0e-6 / (Leff1 * NFIN);

// Nbody Binning Equation for UFCM Parameters
NBODY_i  =  NBODY + Inv_L * LNBODY + Inv_NFIN * NNBODY + Inv_LNFIN * PNBODY;

if (NBODYN1 != 0.0) begin
   NBODY_i  =  NBODY_i * (1.0 + NBODYN1/NFIN * lln(1.0 + NFIN/NBODYN2));
end

// Model Parameters for Unified FinFET Compact Model by Juan Duarte 10/2013
case (GEOMOD)
     0: begin // Double Gate
           if (!$param_given(TFIN_TOP) || !$param_given(TFIN_BASE)) begin
              Weff_UFCM  =  2.0 * HFIN;
              Cins  =  Weff_UFCM * EPSROX * `EPS0 / EOT;
              Ach   =  HFIN * TFIN;
              rc    =  (2.0 * Cins / (Weff_UFCM * Weff_UFCM * epssub / Ach));
              Qdep_ov_Cins  =  -`q * NBODY_i * Ach / Cins;
          end else begin
              Weff_UFCM  =  2.0 * sqrt( HFIN * HFIN + (TFIN_TOP - TFIN_BASE) * (TFIN_TOP -
              Cins  =  Weff_UFCM * EPSROX * `EPS0 / EOT;
              Ach   =  HFIN * (TFIN_TOP + TFIN_BASE) / 2.0;
              rc    =  (2.0 * Cins / (Weff_UFCM * Weff_UFCM * epssub / Ach));
              Qdep_ov_Cins  =  -`q * NBODY_i * Ach / Cins;
          end
        end
      1: begin // Triple Gate
          if (!$param_given(TFIN_TOP) || !$param_given(TFIN_BASE)) begin
              Weff_UFCM  =  2.0 * HFIN + TFIN;
              Cins  =  Weff_UFCM * EPSROX * `EPS0 / EOT;
              Ach   =  HFIN * TFIN;
              rc    =  (2.0 * Cins / (Weff_UFCM * Weff_UFCM * epssub / Ach));
              Qdep_ov_Cins  =  -`q * NBODY_i * Ach / Cins;
          end else begin
              Weff_UFCM  =  2.0 * sqrt(HFIN * HFIN + (TFIN_TOP - TFIN_BASE) * (TFIN_TOP -
              Cins  =  Weff_UFCM * EPSROX * `EPS0 / EOT;
              Ach   =  HFIN * (TFIN_TOP + TFIN_BASE) / 2.0;
              rc    =  (2.0 * Cins /(Weff_UFCM * Weff_UFCM * epssub / Ach));
              Qdep_ov_Cins  =  -`q * NBODY_i * Ach / Cins;
          end
      end
      2: begin // Quadruple Gate
          if (!$param_given(TFIN_TOP) || !$param_given(TFIN_BASE)) begin
              Weff_UFCM  =  2.0 * HFIN + 2.0 * TFIN;
              Cins  =  Weff_UFCM * EPSROX * `EPS0 / EOT;
              Ach   =  HFIN * TFIN;
              rc    =  (2.0 * Cins / (Weff_UFCM * Weff_UFCM * epssub / Ach));
              Qdep_ov_Cins  =  -`q * NBODY_i * Ach / Cins;
          end else begin
                                                          1343,1        32%
```

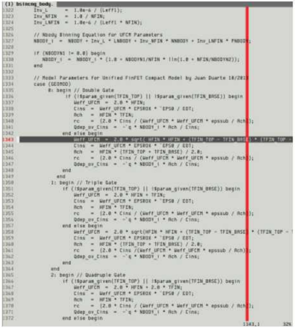
5,444 lines of code

3.1.6 NFIN scaling equations

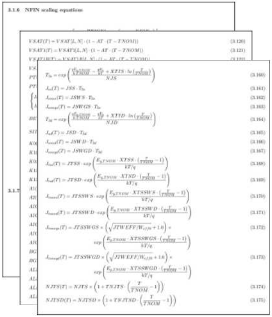
$$VSAT(T) = VSAT[L,N] \cdot (1 - AT \cdot (T - TNOM)) \quad (3.120)$$
$$VSAT1(T) = VSAT1[L,N] \cdot (1 - AT \cdot (T - TNOM)) \quad (3.121)$$
$$VSAT1R(T) = VSAT1R[L,N] \cdot (1 - AT \cdot (T - TNOM)) \quad (3.122)$$
$$VSATCV(T) = VSATCV[L] \cdot (1 - ATCV \cdot (T - TNOM)) \quad (3.123)$$
$$PTWG(T) = PTWG[L] \cdot (1 - PTWGT \cdot (T - TNOM)) \quad (3.124)$$
$$PTWGR(T) = PTWGR[L] \cdot (1 - PTWGT \cdot (T - TNOM)) \quad (3.125)$$
$$\begin{cases} MEXP(T) = MEXP[L] \cdot (1 + TMEXP \cdot (T - TNOM)) & \text{if } ASYMMOD = 0 \\ MEXPR(T) = MEXPR[L] \cdot (1 + TMEXPR \cdot (T - TNOM)) & \text{if } ASYMMOD = 1 \end{cases} \quad (3.126)$$
$$BETA0(T) = BETA0_i \cdot \left(\frac{T}{TNOM}\right)^{UTE} \quad (3.127)$$
$$SII0(T) = SII0_i \left(1 + TII\left(\frac{T}{TNOM} - 1\right)\right) \quad (3.128)$$
$$K0(T) = K0_i + K01_i \cdot (T - TNOM) \quad (3.129)$$
$$K1(T) = K1_i + K11_i \cdot (T - TNOM) \quad (3.130)$$
$$K0SI(T) = K0SI_i + K0SI1_i \cdot (T - TNOM) \quad (3.131)$$
$$K1SI(T) = K1SI_i + K1SI1_i \cdot (T - TNOM) \quad (3.132)$$
$$K1SAT(T) = K1SAT_i + K1SAT1_i \cdot (T - TNOM) \quad (3.133)$$
$$A1(T) = A1_i + A11_i \cdot (T - TNOM) \quad (3.134)$$
$$A2(T) = A2_i + A21_i \cdot (T - TNOM) \quad (3.135)$$
$$AIGBINV(T) = AIGBINV_i + AIGBINV1_i \cdot (T - TNOM) \quad (3.136)$$
$$AIGBACC(T) = AIGBACC_i + AIGBACC1_i \cdot (T - TNOM) \quad (3.137)$$
$$AIGC(T) = AIGC_i + AIGC1_i \cdot (T - TNOM) \quad (3.138)$$
$$AIGS(T) = AIGS_i + AIGS1_i \cdot (T - TNOM) \quad (3.139)$$
$$AIGD(T) = AIGD_i + AIGD1_i \cdot (T - TNOM) \quad (3.140)$$
$$BGIDL(T) = BGIDL_i \cdot (1 + TGIDL \cdot (T - TNOM)) \quad (3.141)$$
$$BGISL(T) = BGISL_i \cdot (1 + TGIDL \cdot (T - TNOM)) \quad (3.142)$$
$$ALPHA0(T) = ALPHA0_i + ALPHA01_i \cdot (T - TNOM) \quad (3.143)$$
$$ALPHA1(T) = ALPHA1_i + ALPHA11_i \cdot (T - TNOM) \quad (3.144)$$
$$ALPHAII0(T) = ALPHAII0_i + ALPHAII01_i \cdot (T - TNOM) \quad (3.145)$$
$$ALPHAII1(T) = ALPHAII1_i + ALPHAII11_i \cdot (T - TNOM) \quad (3.146)$$

3.1.7

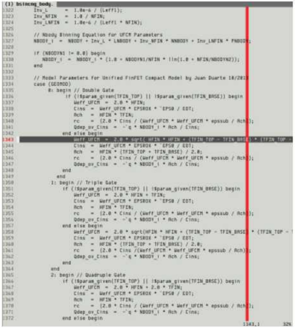
“Real” compact models: the BSIM family



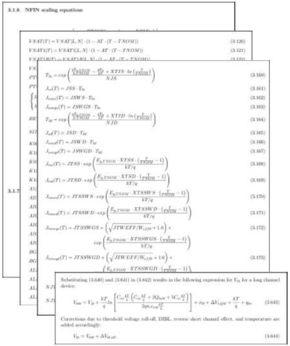
5,444 lines of code



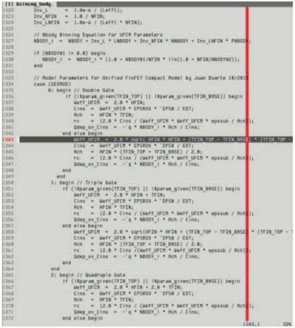
“Real” compact models: the BSIM family



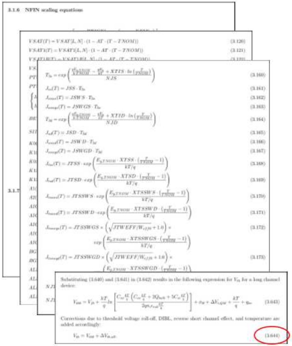
5,444 lines of code



"Real" compact models: the BSIM family

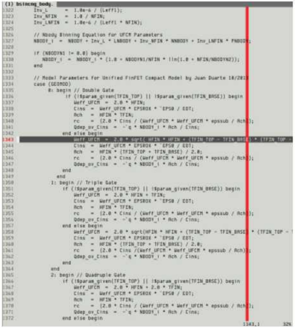


5,444 lines of code

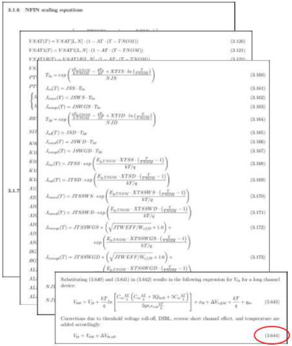


Pages and pages of equations (644 in all)

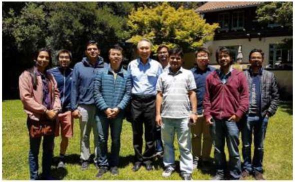
“Real” compact models: the BSIM family



5,444 lines of code

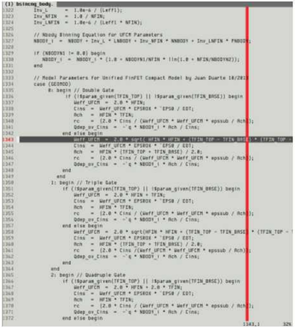


Pages and pages of equations (644 in all)

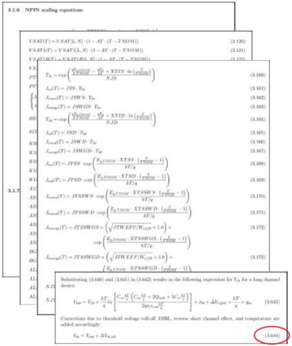


Several PhD students over 20+ years; this is just the current group.

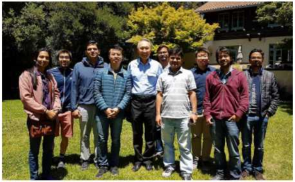
"Real" compact models: the BSIM family



5,444 lines of code



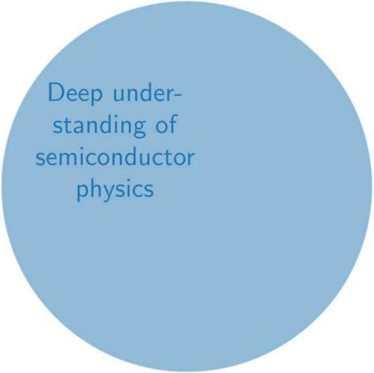
Pages and pages of equations (644 in all)



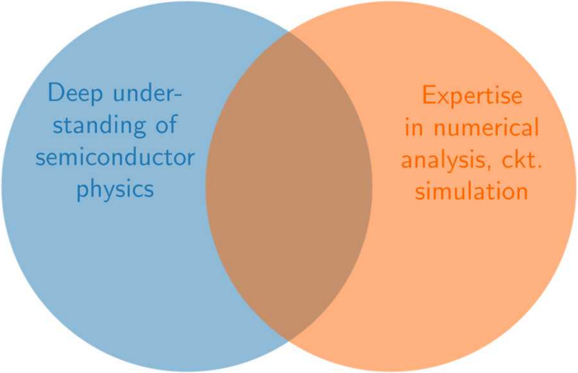
Several PhD students over 20+ years; this is just the current group.

Developing a compact model: NOT an easy undertaking!

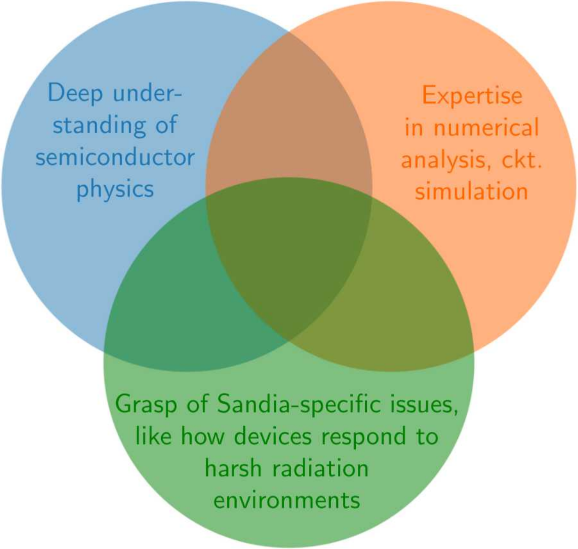
Compact model development: what it takes



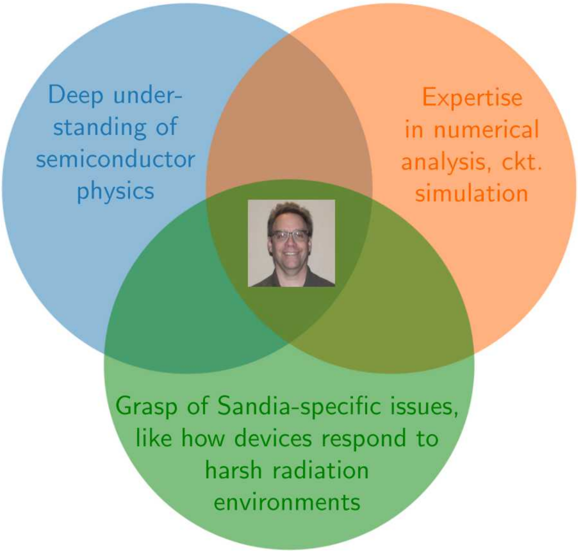
Compact model development: what it takes



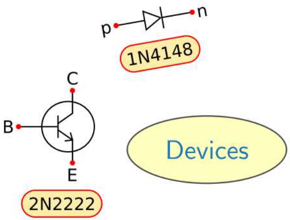
Compact model development: what it takes



Compact model development: what it takes

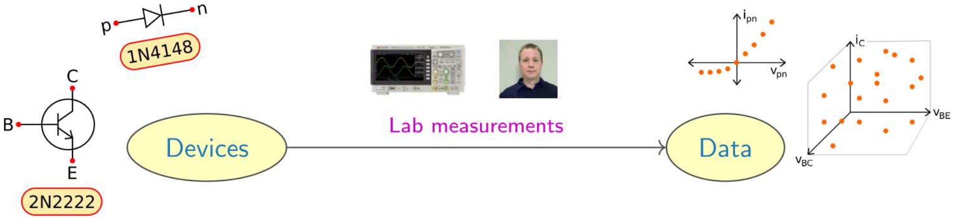


Our approach: Data-driven compact models (1/2)

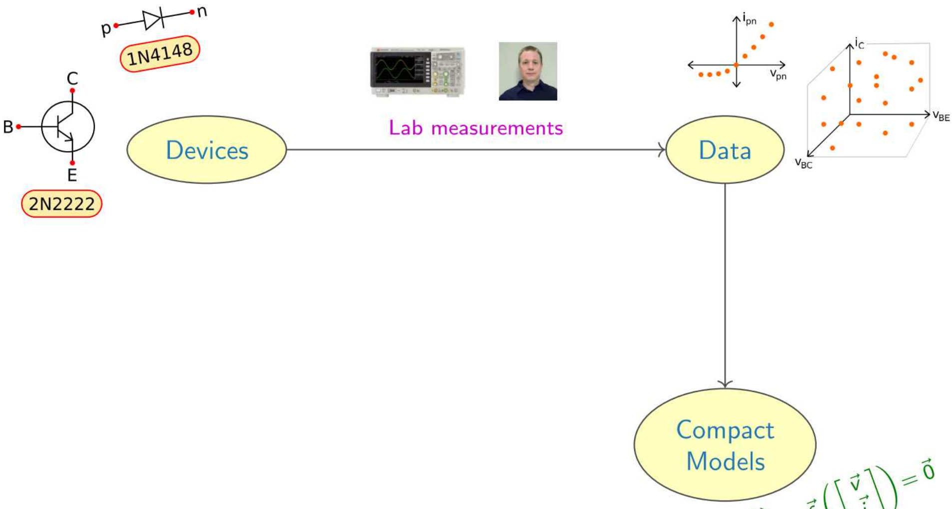


Our approach: Data-driven compact models (1/2)

Our approach: Data-driven compact models (1/2)

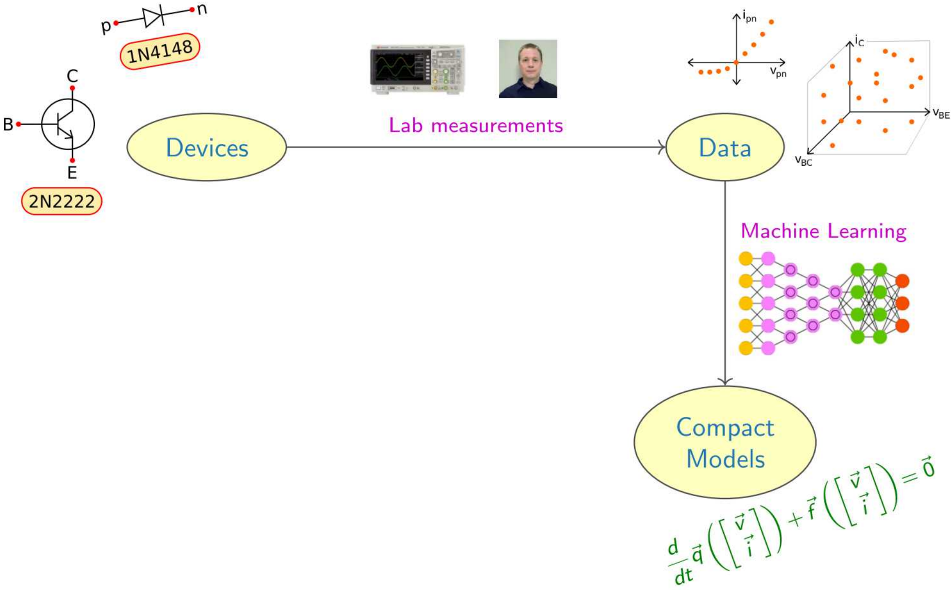


Our approach: Data-driven compact models (1/2)

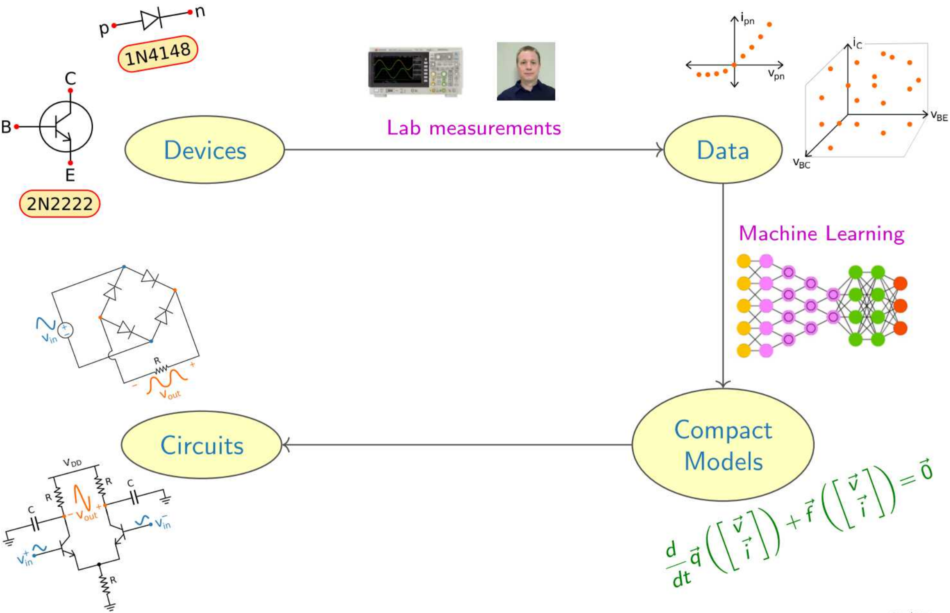


$$\frac{d}{dt}\vec{q}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right)+\vec{f}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right)=\vec{0}$$

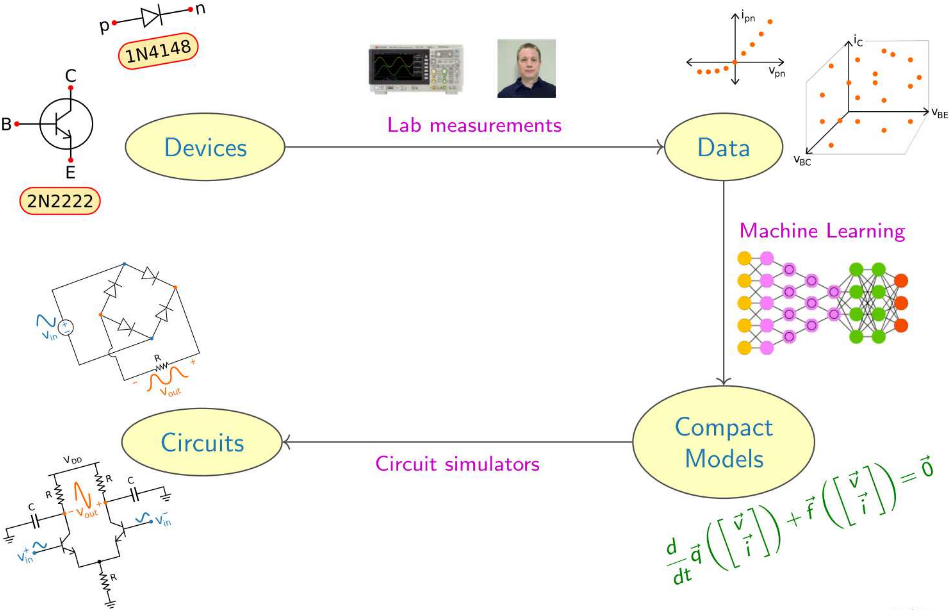
Our approach: Data-driven compact models (1/2)



Our approach: Data-driven compact models (1/2)



Our approach: Data-driven compact models (1/2)



Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning		
Compact Model Development		
Circuit Simulation		

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
Compact Model Development		
Circuit Simulation		

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre  
Compact Model Development		
Circuit Simulation		

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre  
	Multi-dimensional splines	STEAM Berkeley UNIVERSITY OF CALIFORNIA
Compact Model Development		
Circuit Simulation		

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre  
	Multi-dimensional splines	STEAM Berkeley UNIVERSITY OF CALIFORNIA
Compact Model Development	Differential-algebraic equation systems	
Circuit Simulation		

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre  
	Multi-dimensional splines	STEAM Berkeley
Compact Model Development	Differential-algebraic equation systems	ModSpec Berkeley
Circuit Simulation		

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre  
	Multi-dimensional splines	STEAM Berkeley UNIVERSITY OF CALIFORNIA
Compact Model Development	Differential-algebraic equation systems	ModSpec Berkeley UNIVERSITY OF CALIFORNIA
		Spyce 
Circuit Simulation		

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre  
	Multi-dimensional splines	STEAM Berkeley UNIVERSITY OF CALIFORNIA
Compact Model Development	Differential-algebraic equation systems	ModSpec Berkeley UNIVERSITY OF CALIFORNIA
		Spyce 
Circuit Simulation	Newton-Raphson, limiting, linear multi-step methods like forward and backward Euler, small-signal circuit analysis, *etc.*	

Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre 
	Multi-dimensional splines	STEAM Berkeley UNIVERSITY OF CALIFORNIA
Compact Model Development	Differential-algebraic equation systems	ModSpec Berkeley UNIVERSITY OF CALIFORNIA
		Spyce 
Circuit Simulation	Newton-Raphson, limiting, linear multi-step methods like forward and backward Euler, small-signal circuit analysis, *etc.*	Xyce 

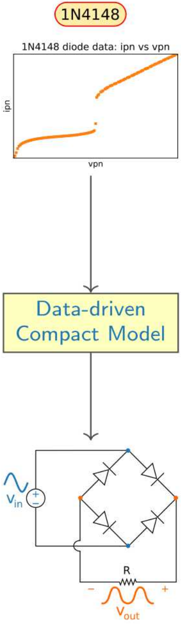
Our approach: Data-driven compact models (2/2)

Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre  
	Multi-dimensional splines	STEAM Berkeley UNIVERSITY OF CALIFORNIA
Compact Model Development	Differential-algebraic equation systems	ModSpec Berkeley UNIVERSITY OF CALIFORNIA
		Spyce 
Circuit Simulation	Newton-Raphson, limiting, linear multi-step methods like forward and backward Euler, small-signal circuit analysis, *etc.*	Xyce 
		MAPP Berkeley UNIVERSITY OF CALIFORNIA

Our approach: Data-driven compact models (2/2)

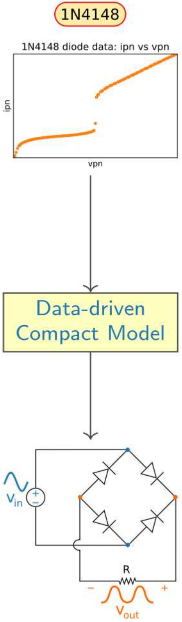
Task	Techniques we use	Software we use
Machine Learning	Backpropagation, deep learning, neural networks	TensorFlow
	GMLS	Compadre 
	Multi-dimensional splines	STEAM Berkeley UNIVERSITY OF CALIFORNIA
Compact Model Development	Differential-algebraic equation systems	ModSpec Berkeley UNIVERSITY OF CALIFORNIA
		Spyce 
Circuit Simulation	Newton-Raphson, limiting, linear multi-step methods like forward and backward Euler, small-signal circuit analysis, *etc.*	Xyce 
		MAPP Berkeley UNIVERSITY OF CALIFORNIA
		Spyce 

Results (1/2)

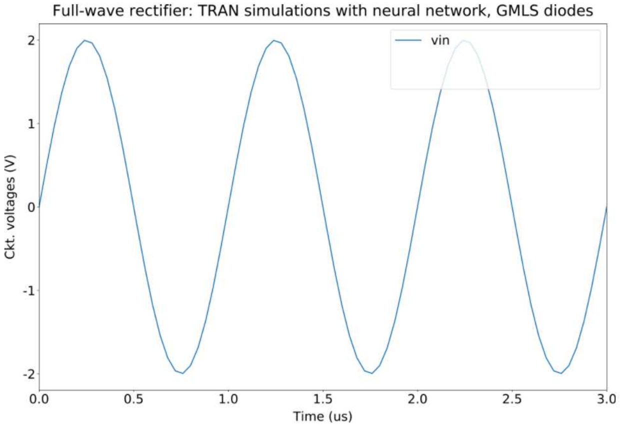


- We designed a full-wave rectifier circuit using data-driven compact models for a 1N4148 diode.

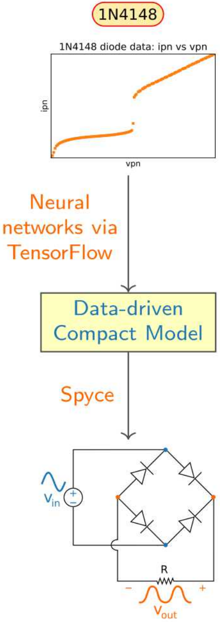
Results (1/2)



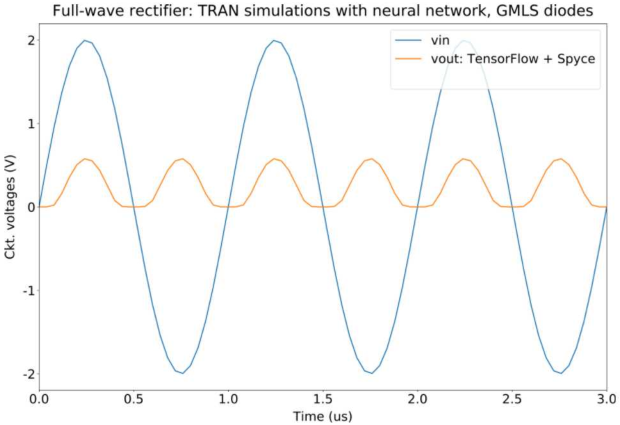
- We designed a full-wave rectifier circuit using data-driven compact models for a 1N4148 diode.



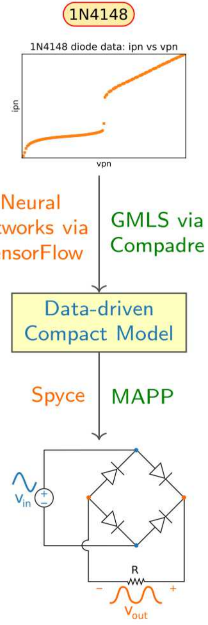
Results (1/2)



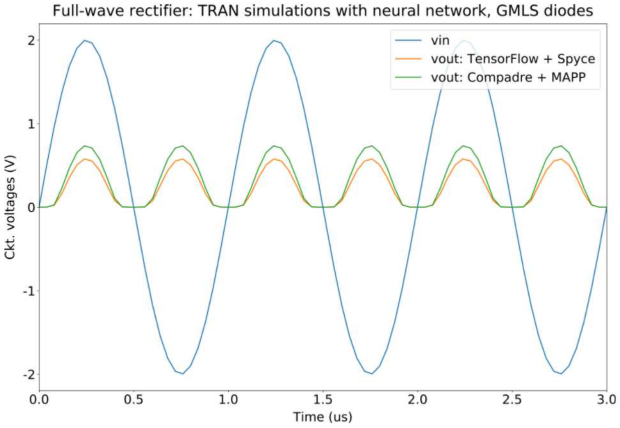
- We designed a full-wave rectifier circuit using data-driven compact models for a 1N4148 diode.



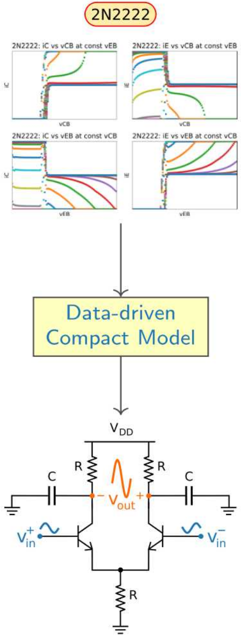
Results (1/2)



- We designed a full-wave rectifier circuit using data-driven compact models for a 1N4148 diode.

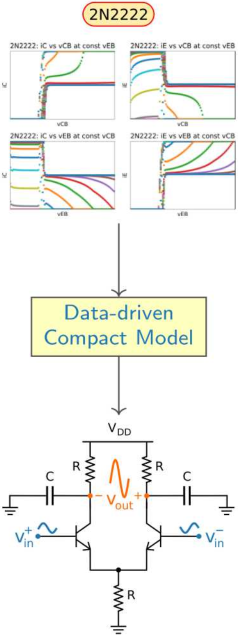


Results (2/2)

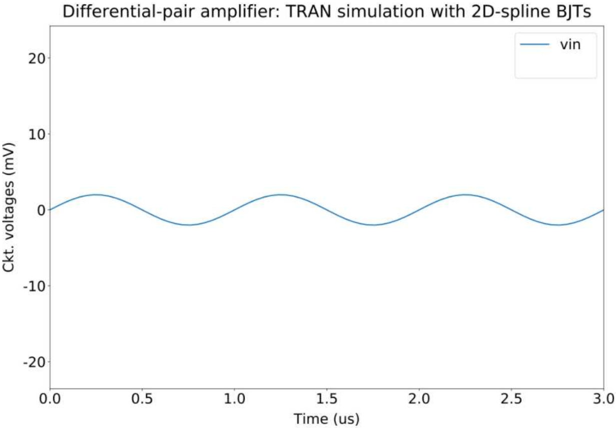


- We designed a differential-pair amplifier using a data-driven compact model for a 2N2222 BJT.

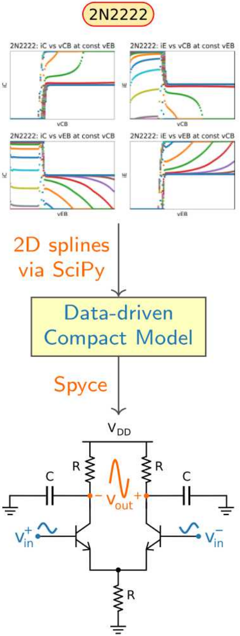
Results (2/2)



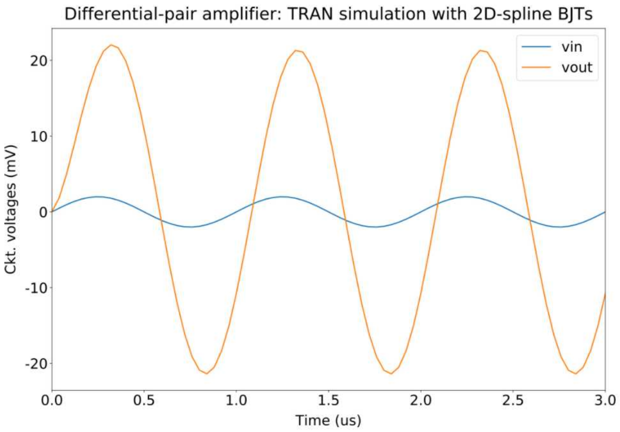
- We designed a differential-pair amplifier using a data-driven compact model for a 2N2222 BJT.



Results (2/2)



- We designed a differential-pair amplifier using a data-driven compact model for a 2N2222 BJT.



Challenges

- Learning $\vec{q}$ is harder than learning $\vec{f}$

$$\text{Compact model: } \frac{d}{dt}\,\vec{q}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right) + \vec{f}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right) = \vec{0}$$

  - We have techniques to learn $\vec{f}$ from DC data
  - We are working on how to learn $\vec{q}$ from small-signal impedance data

Challenges

- Learning $\vec{q}$ is harder than learning $\vec{f}$

$$\text{Compact model: } \frac{d}{dt}\,\vec{q}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right) + \vec{f}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right) = \vec{0}$$

  - We have techniques to learn $\vec{f}$ from DC data
  - We are working on how to learn $\vec{q}$ from small-signal impedance data

- Newton-Raphson often fails with data-driven compact models
  - Idea 1. Construct models with better monotonicity
  - Idea 2. Limit oscillations with Predictor/Corrector Newton-Raphson
  - Idea 3. Extrapolation heuristics for GMLS

Challenges

- Learning $\vec{q}$ is harder than learning $\vec{f}$

  Compact model: $$\frac{d}{dt}\,\vec{q}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right)+\vec{f}\left(\begin{bmatrix}\vec{v}\\ \vec{i}\end{bmatrix}\right)=\vec{0}$$

  - We have techniques to learn $\vec{f}$ from DC data
  - We are working on how to learn $\vec{q}$ from small-signal impedance data

- Newton-Raphson often fails with data-driven compact models
  - Idea 1. Construct models with better monotonicity
  - Idea 2. Limit oscillations with Predictor/Corrector Newton-Raphson
  - Idea 3. Extrapolation heuristics for GMLS

- The compact model form above may not be general enough
  - Cannot handle "dead delays", transmission lines
  - We may need to generalize it further

Questions