TFE4188 - Lecture 11

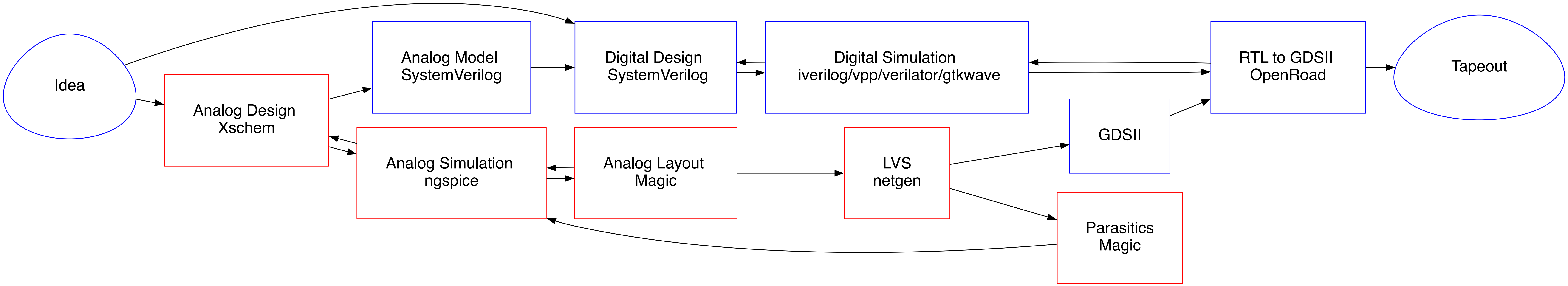
Analog SystemVerilog

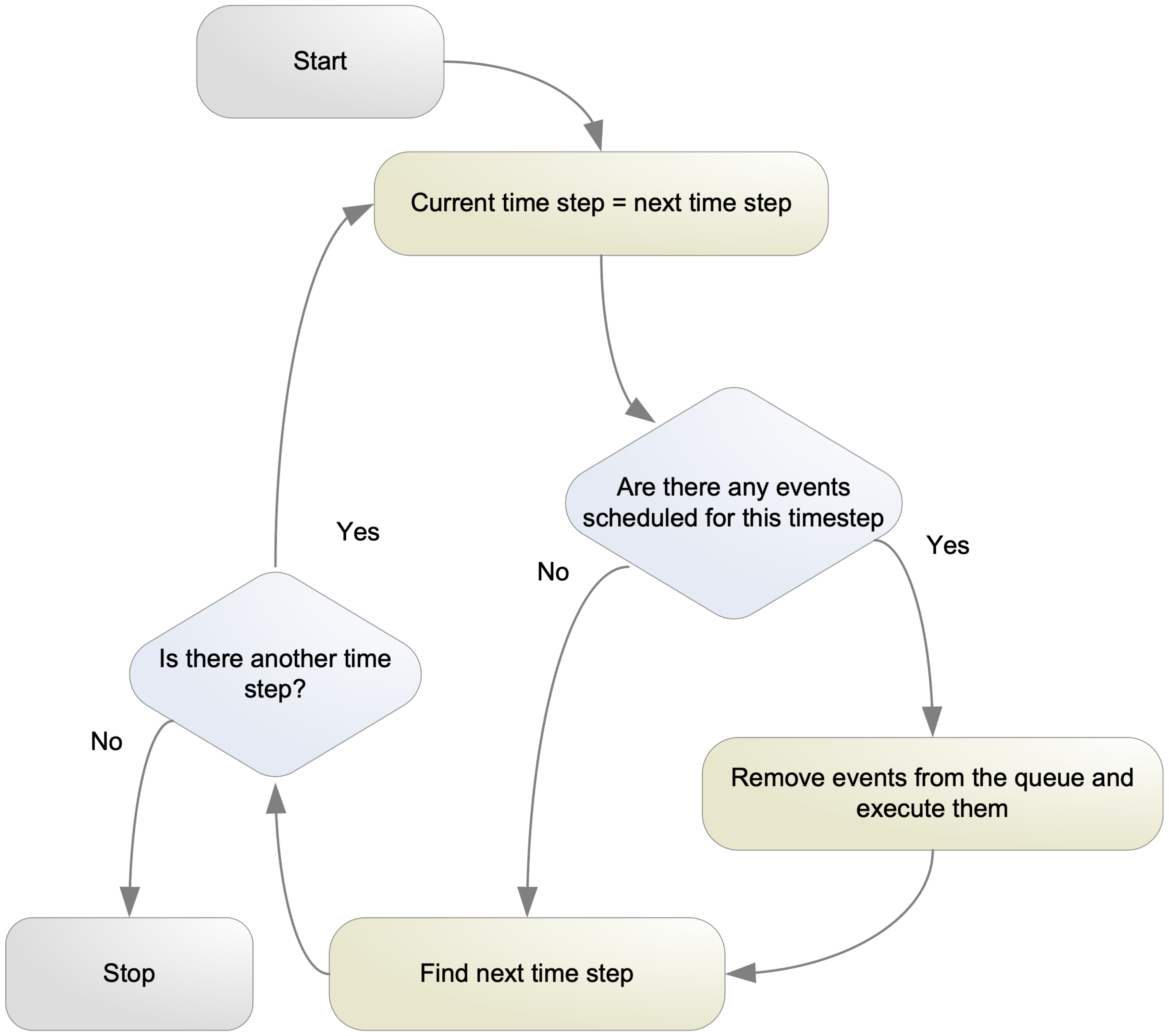
Goal

Explain why we need **Analog SystemVerilog** models

Introduction to Analog SystemVerilog

Why

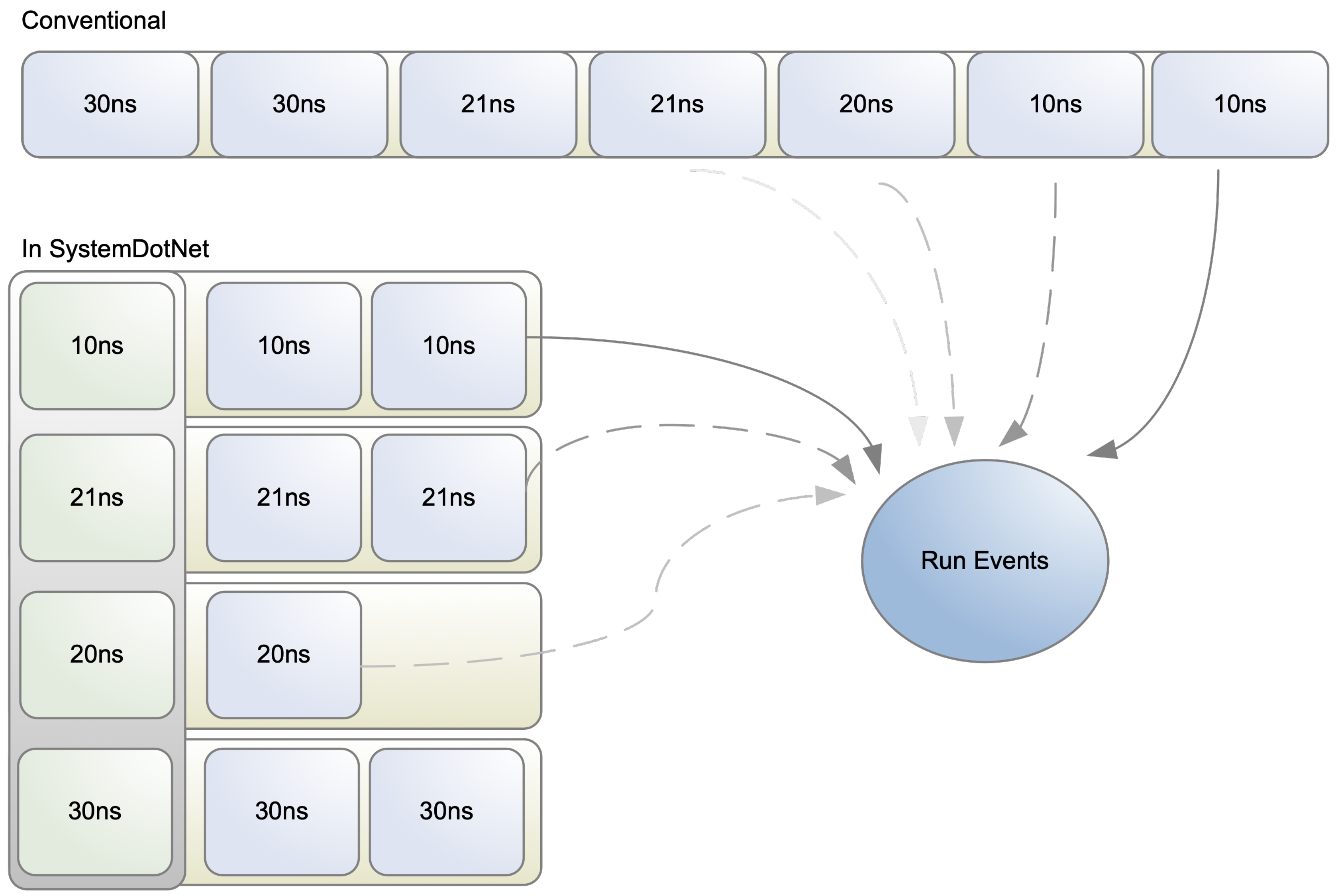

Idea
Analog Design
Xschem
Analog Model
SystemVerilog
Digital Design
SystemVerilog
Digital Simulation
iverilog/vpp/verilator/gtkwave
RTL to GDSII
OpenRoad
Tapeout
Analog Simulation
ngspice
Analog Layout
Magic
LVS
netgen
GDSII
Parasitics
Magic


Start
Current time step = next time step
Are there any events scheduled for this timestep
Yes
No
Remove events from the queue and execute them
Find next time step
Is there another time step?
Yes
No
Stop

Digital simulation

- The order of execution of events at the same timestep do not matter
- The system is causal. Changes in the future do not affect signals in the past

Digital Simulators



Commercial

Cadence Excelium

Siemens Questa

Synopsys VCS

Open Source

iverilog/vpp

Verilator

SystemDotNet

```
module counter(
               output logic [WIDTH-1:0] out,
               input logic              clk,
               input logic              reset
               );

   parameter WIDTH = 8;

   logic [WIDTH-1:0]                    count;
   always_comb begin
      count = out + 1;
   end

   always_ff @(posedge clk or posedge reset) begin
      if (reset)
        out <= 0;
      else
        out <= count;
   end

endmodule // counter
```

Counter

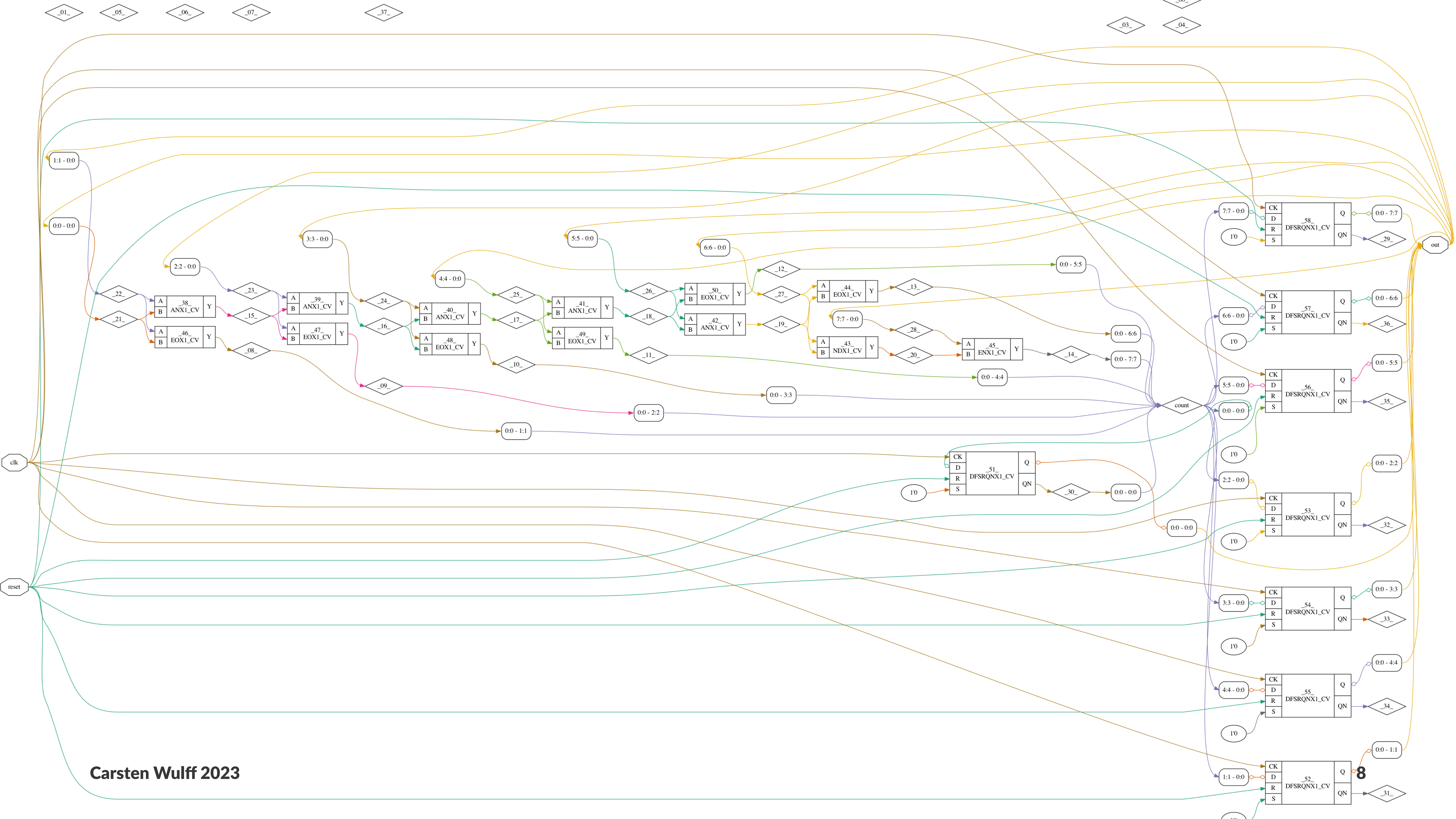
Assume `clk, reset, out = 0`

Assume event with `clk = 1`

0: Set `out = count` in next event (1)

1: Set `count = out + 1` using logic (may consume multiple events)

X: no further events



Carsten Wulff 2023

Transient analog simulation

$$\begin{pmatrix} y_1^{(n)} \\ y_2^{(n)} \\ \vdots \\ y_m^{(n)} \end{pmatrix} = \begin{pmatrix} f_1\left(x, \mathbf{y}, \mathbf{y}', \mathbf{y}'', \ldots, \mathbf{y}^{(n-1)}\right) \\ f_2\left(x, \mathbf{y}, \mathbf{y}', \mathbf{y}'', \ldots, \mathbf{y}^{(n-1)}\right) \\ \vdots \\ f_m\left(x, \mathbf{y}, \mathbf{y}', \mathbf{y}'', \ldots, \mathbf{y}^{(n-1)}\right) \end{pmatrix}$$

Parse spice netlist, and setup partial/ordinary differential equations for node matrix

Model non-linear current/voltage behavior between all nodes

Use numerical methods to compute time evolution

- Euler
- Runge-Kutta
- Crank-Nicolson
- Gear

How spice works

Simulation Program with Integrated Circuit Emphasis (SPICE)

Published in 1973 by Nagel and Pederson

SPICE (Simulation Program with Integrated Circuit Emphasis)

https://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/ERL-m-382.pdf

Commercial

Cadence Spectre

Siemens Eldo

Synopsys HSPICE

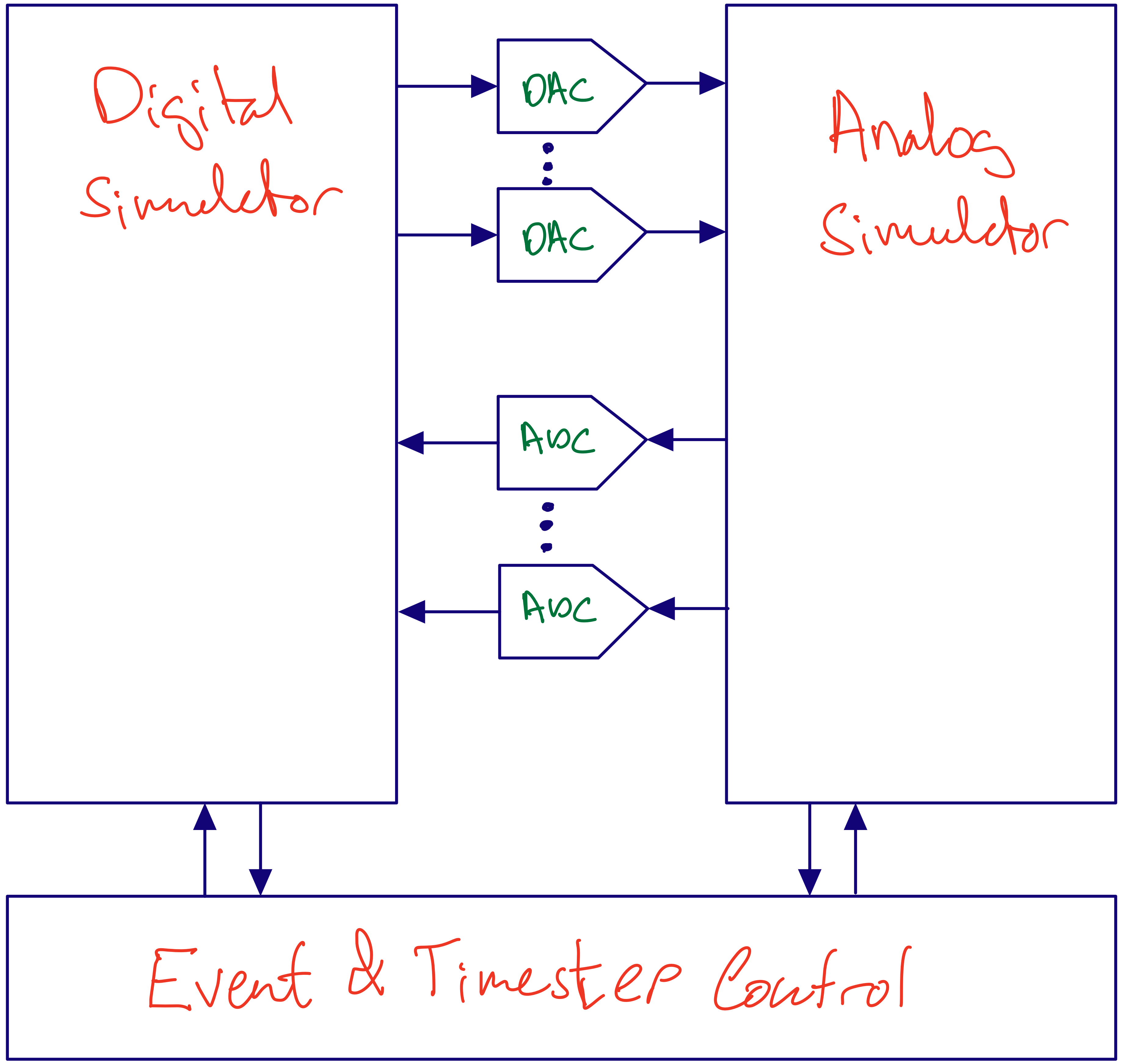
Free

Aimspice

Analog Devices LTspice

Open Source

ngspice


Digital Simulator
DAC
DAC
ADC
ADC
Analog Simulator
Event & Timestep Control

Analog SystemVerilog

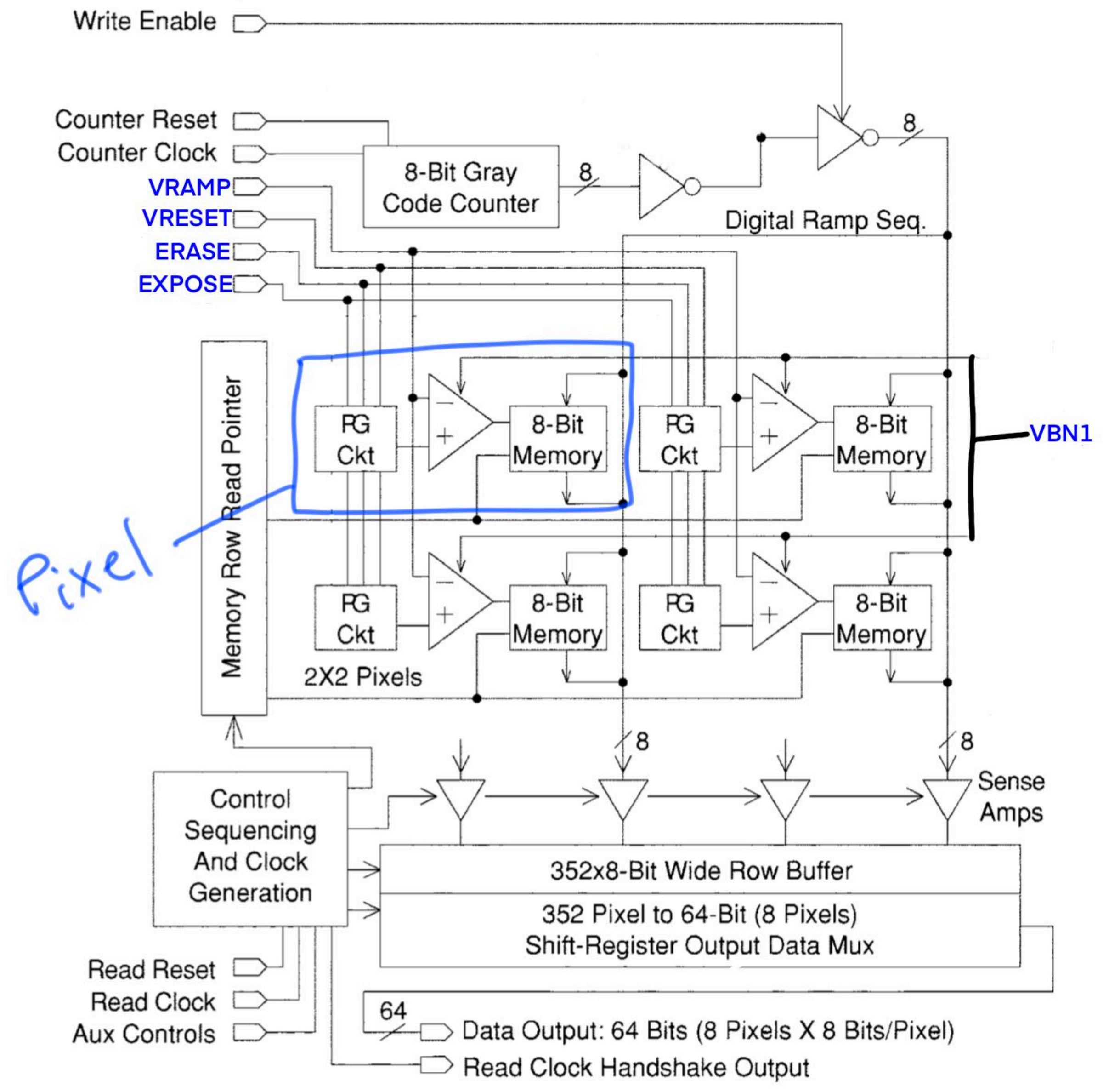
TFE4152 2021 - Project

Be inspired by the ISSCC paper, and design a similar system.

Design analog circuits in SPICE

Design digital circuits in SystemVerilog

A 10 000 Frames/s CMOS Digital Pixel Sensor



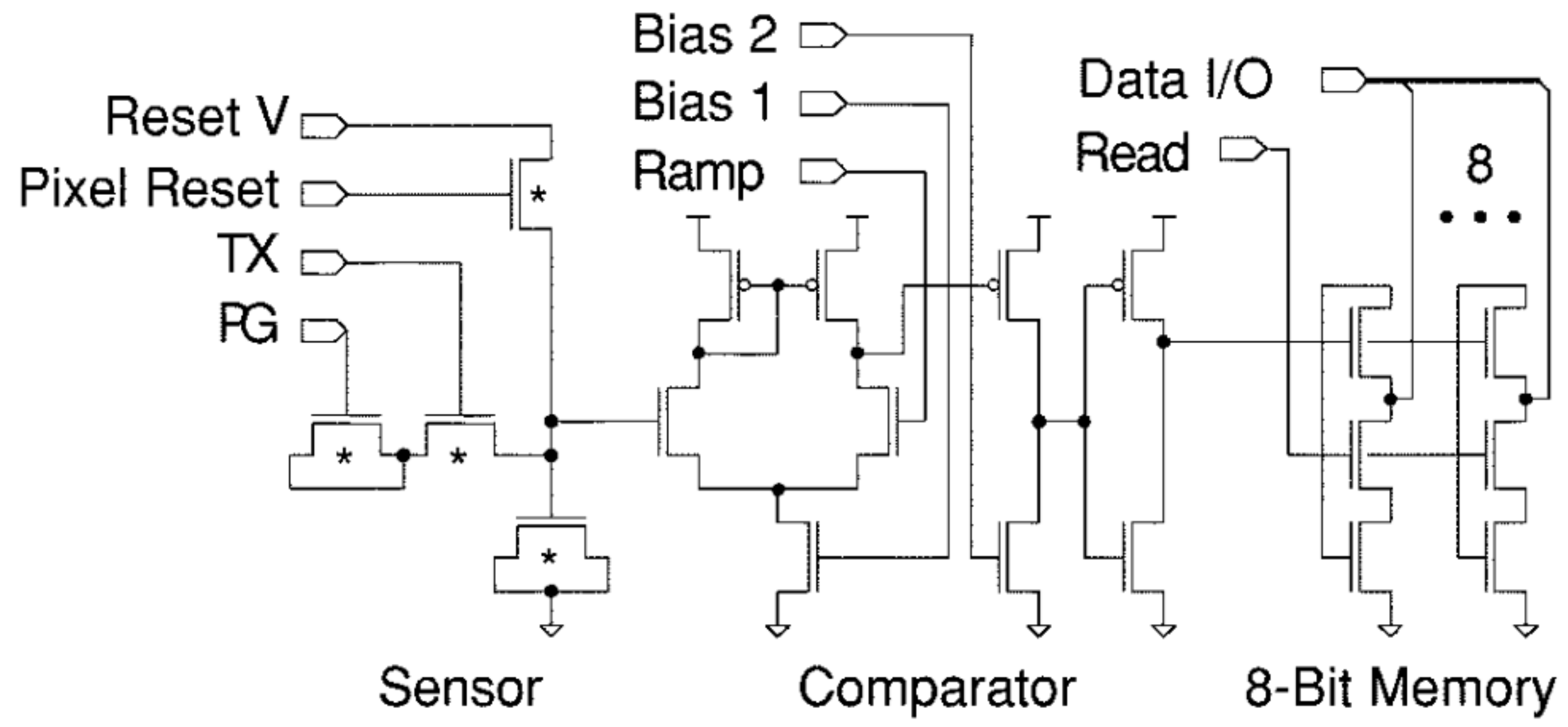
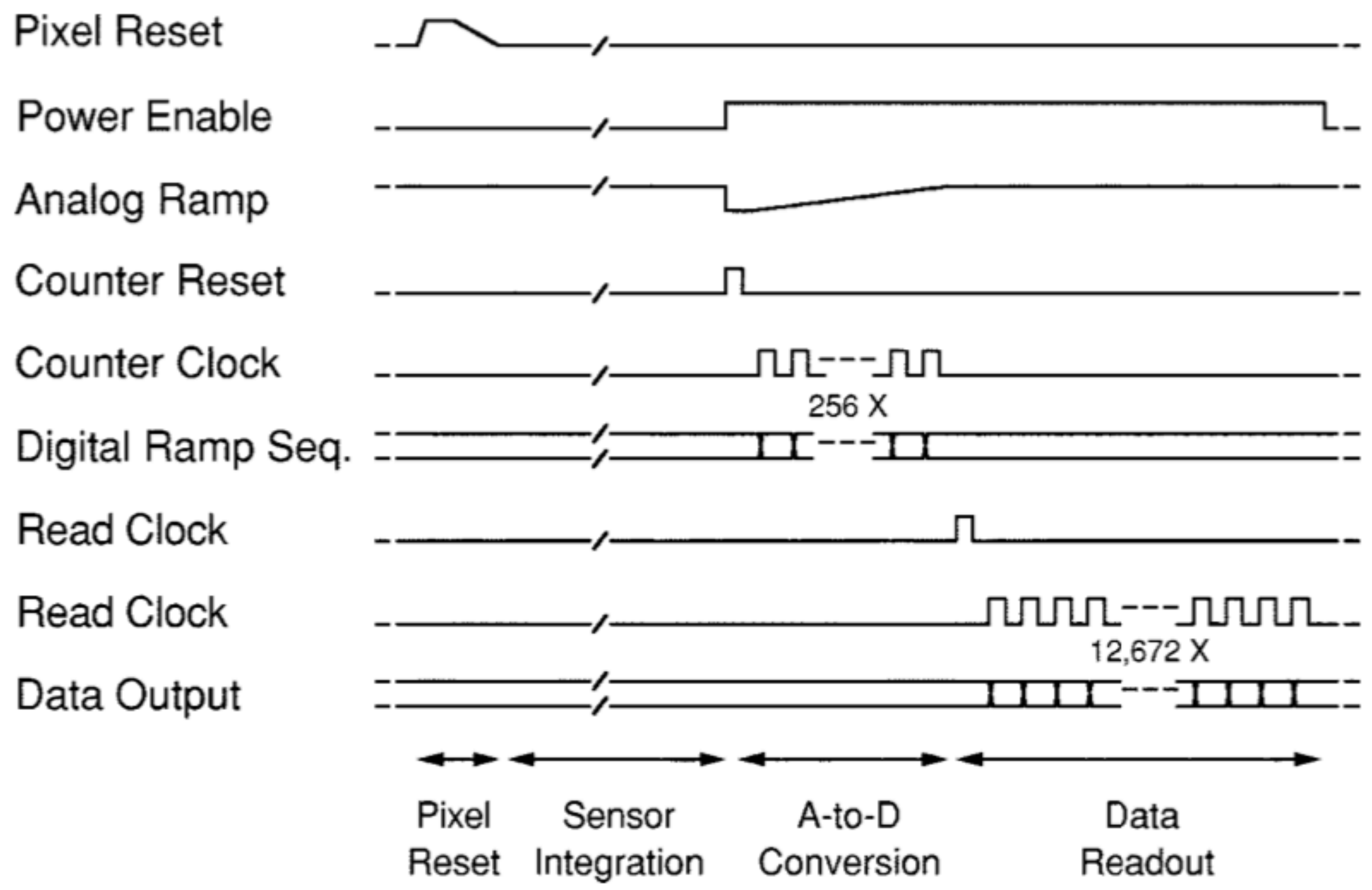
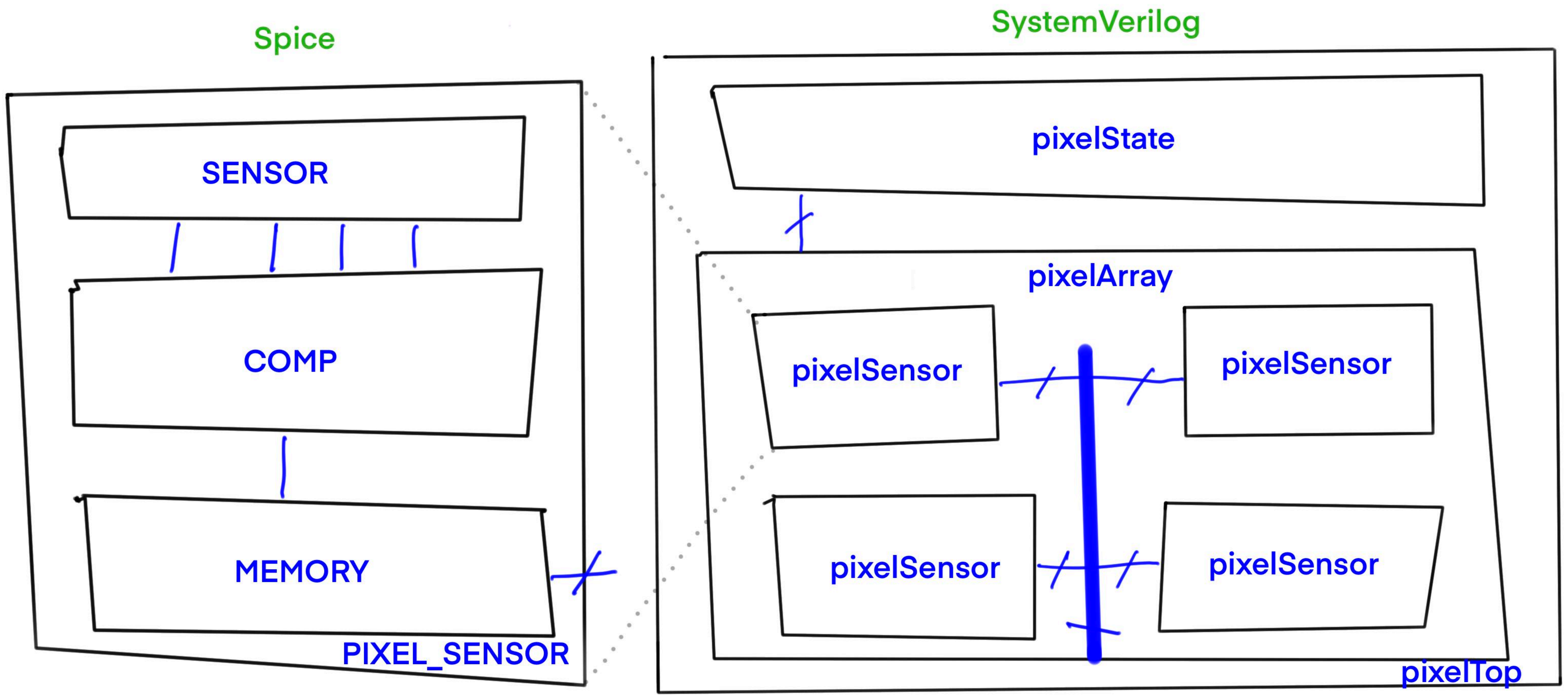


Fig. 4. Pixel schematic.



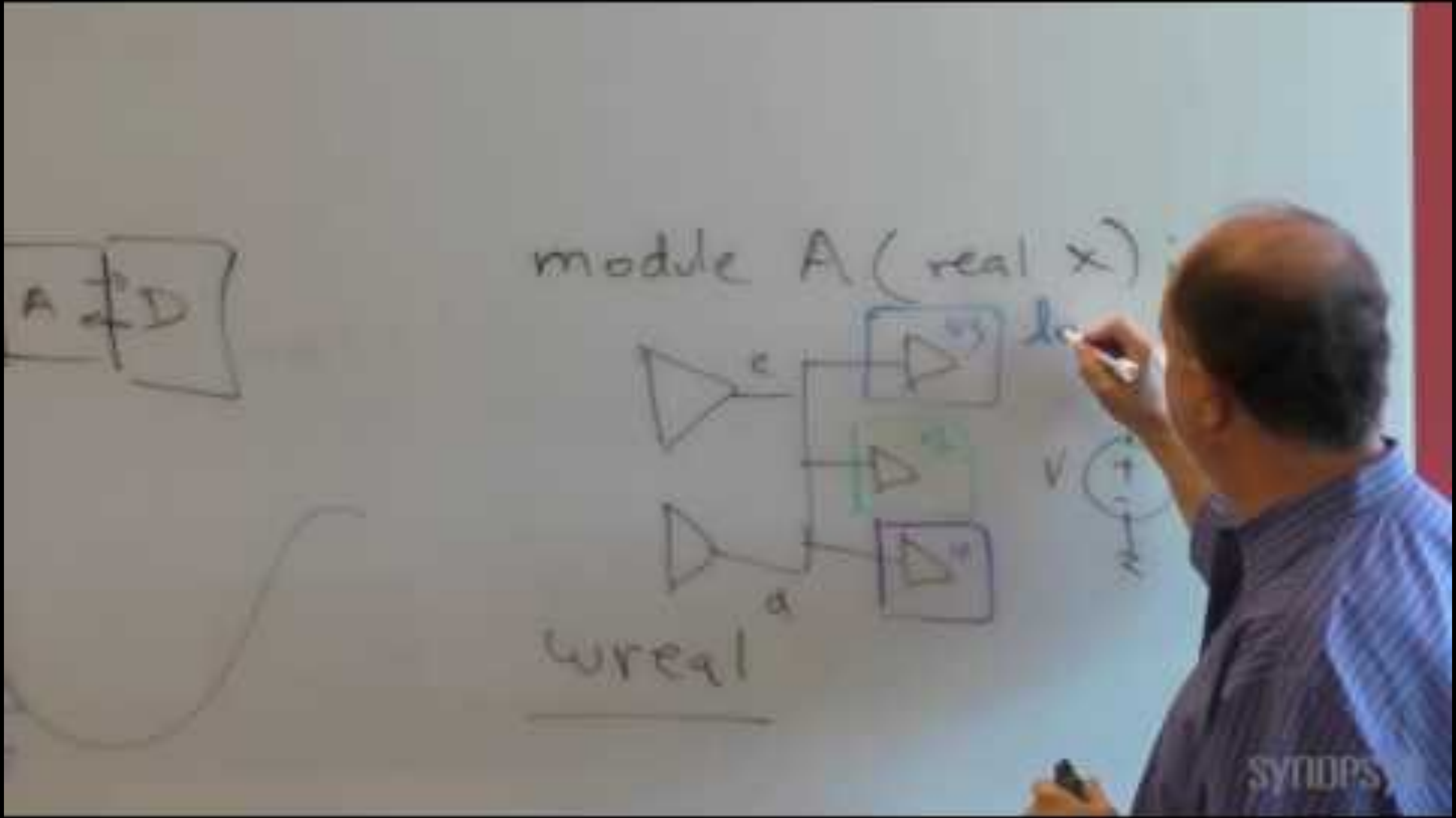

Spice
SystemVerilog
SENSOR
COMP
MEMORY
PIXEL_SENSOR
pixelState
pixelArray
pixelSensor
pixelSensor
pixelSensor
pixelSensor
pixelTop

What you got

github.com/wulffern/dicex

```
project/
├── spice/
│   ├── Makefile                  # See https://www.gnu.org/software/make/manual/html_node/Introduction.html
│   ├── pixelSensor.cir           # Almost empty circuit for pixelSensor
│   └── pixelSensor_tb.cir        # SPICE testbench for pixelSensor, emulates verilog
└── verilog/
    ├── Makefile
    ├── pixelSensor.fl            # Verilog file list
    ├── pixelSensor_tb.gtkw       # Save file for GTKWave
    ├── pixelSensor_tb.v          # Verilog testbench for pixelSensor
    └── pixelSensor.v             # Verilog model of analog pixelSensor circuit
```

```
module PIXEL_SENSOR
  (
   input logic      VBN1,
   input logic      RAMP,
   input logic      RESET,
   input logic      ERASE,
   input logic      EXPOSE,
   input logic      READ,
   inout [7:0] DATA

   );

   real             v_erase = 1.2;
   real             lsb = v_erase/255;
   parameter real   dv_pixel = 0.5;

   real             tmp;
   logic            cmp;
   real             adc;

   logic [7:0]      p_data;

   //------------------------------------------------------------------
   // ERASE
   //------------------------------------------------------------------
   // Reset the pixel value on pixRst
   always @(ERASE) begin
      tmp = v_erase;
      p_data = 0;
      cmp  = 0;
      adc = 0;
   end

   //------------------------------------------------------------------
   // SENSOR
   //------------------------------------------------------------------
   // Use bias to provide a clock for integration when exposing
   always @(posedge VBN1) begin
      if(EXPOSE)
        tmp = tmp - dv_pixel*lsb;
   end

   //------------------------------------------------------------------
   // Comparator
   //------------------------------------------------------------------
   // Use ramp to provide a clock for ADC conversion, assume that ramp
   // and DATA are synchronous
   always @(posedge RAMP) begin
      adc = adc + lsb;
      if(adc > tmp)
        cmp <= 1;
   end

   //------------------------------------------------------------------
   // Memory latch
   //------------------------------------------------------------------
   always_comb  begin
      if(!cmp) begin
         p_data = DATA;
      end

   end

   //------------------------------------------------------------------
   // Readout
   //------------------------------------------------------------------
   // Assign data to bus when pixRead = 0
   assign DATA = READ ? p_data : 8'bZ;

endmodule // re_control
```


A
D
module A (real x)
c
a
V
wreal
SYNOPSYS

Thanks!