

VWSIM: A Circuit Simulator

Warren A. Hunt, Jr., Vivek Ramanathan, and
J Strother Moore

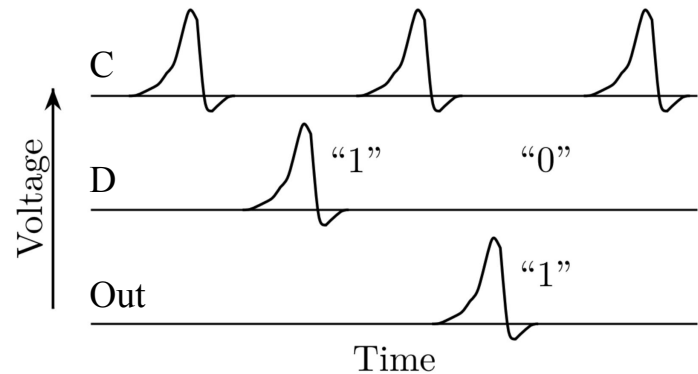
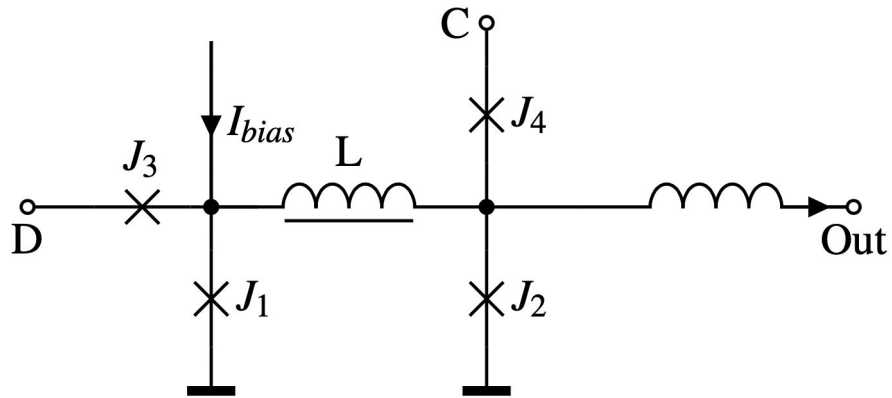
May 26, 2022

The University of Texas at Austin and ForrestHunt, Inc.
{hunt,vivek,moore}@forresthunt.com

ACL2 Workshop 2022

Faster, energy-efficient computing

- Can we build faster, more energy-efficient computers?
- Approach: Rapid Single Flux Quantum (RSFQ) circuits

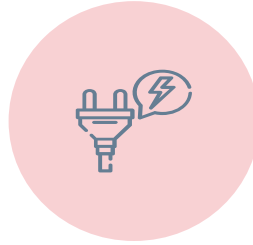


RSFQ properties



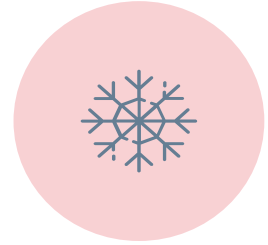
High-speed

Increases speed
by ~100x



Low-energy

Reduces energy
consumption by
~10-20x

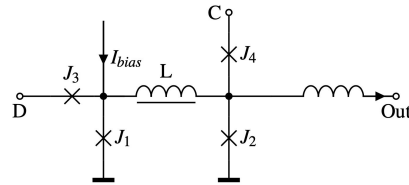


Very Cold

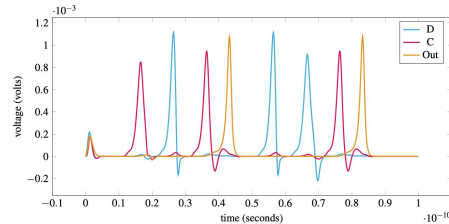
Operates at
~4 Kelvin

Circuit development workflow

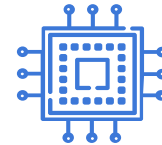
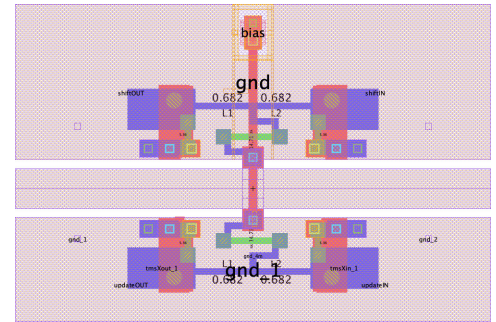
Design



Simulation



Layout & Fabrication



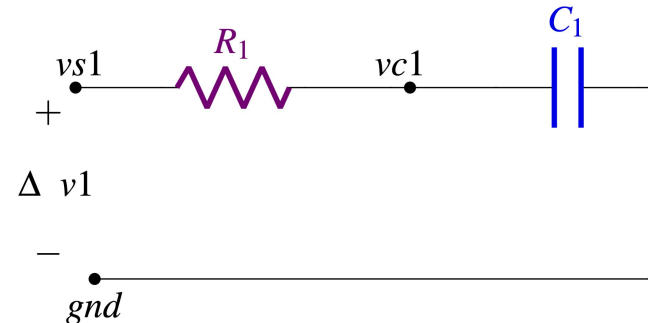
Why build a simulator in ACL2?

1. Understand the **mathematics of RSFQ** circuits (JJs)
2. Understand how **existing circuit simulators** work
3. **Program simulator** to perform collections of simulations
4. **Pause, save, and restart** simulations
5. Develop a **formal semantics** for RSFQ circuits
6. Develop an adequate **model for the behavior of RSFQ circuits**
7. **Prove termination and guards** to ensure absence of memory-reference errors

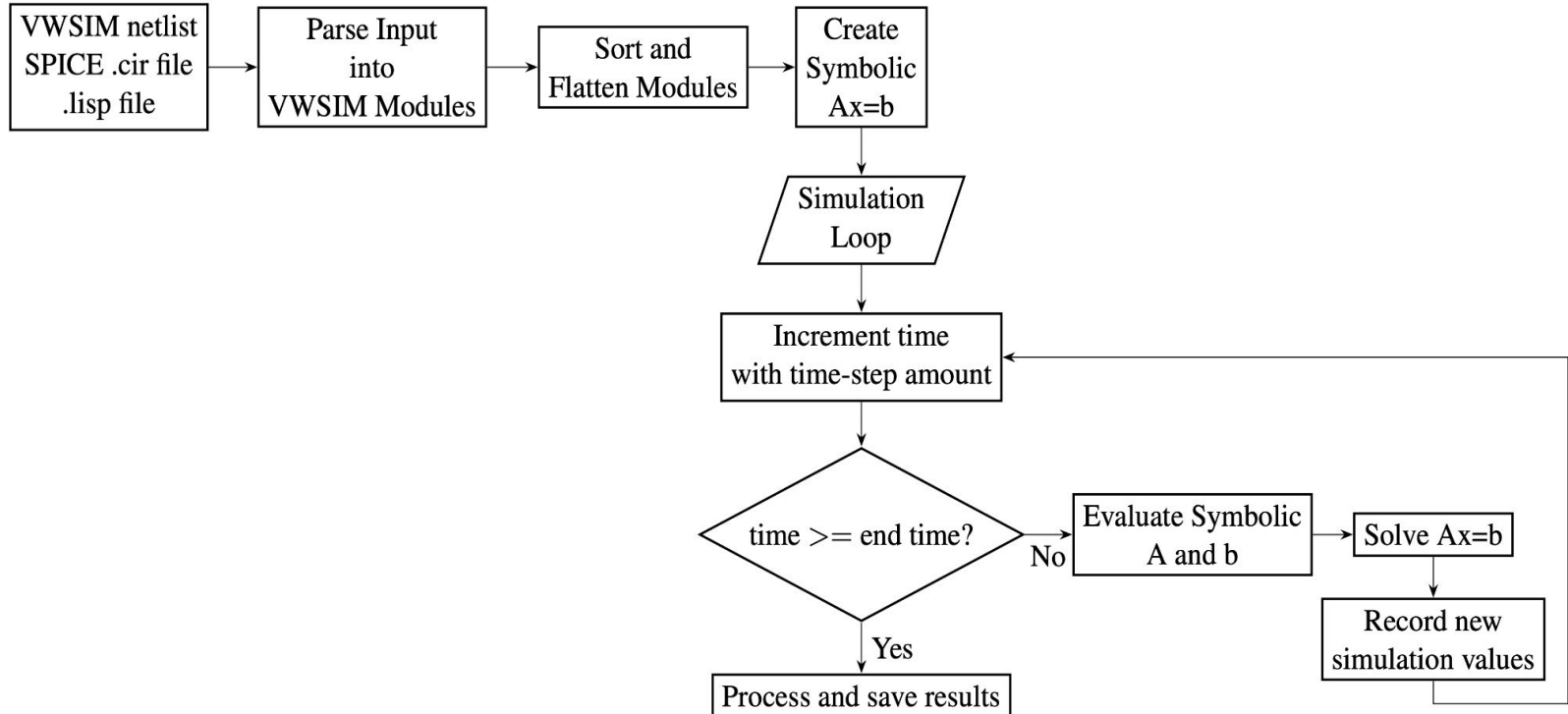
What VWSIM produces

- VWSIM simulates a circuit over a time interval given a start time, time-step size, and stop time
- The values that can be produced for each time step are:
 1. Voltages of wires (with respect to a reference node)
 2. Currents through devices
 3. Phases of wires (with respect to a reference node)

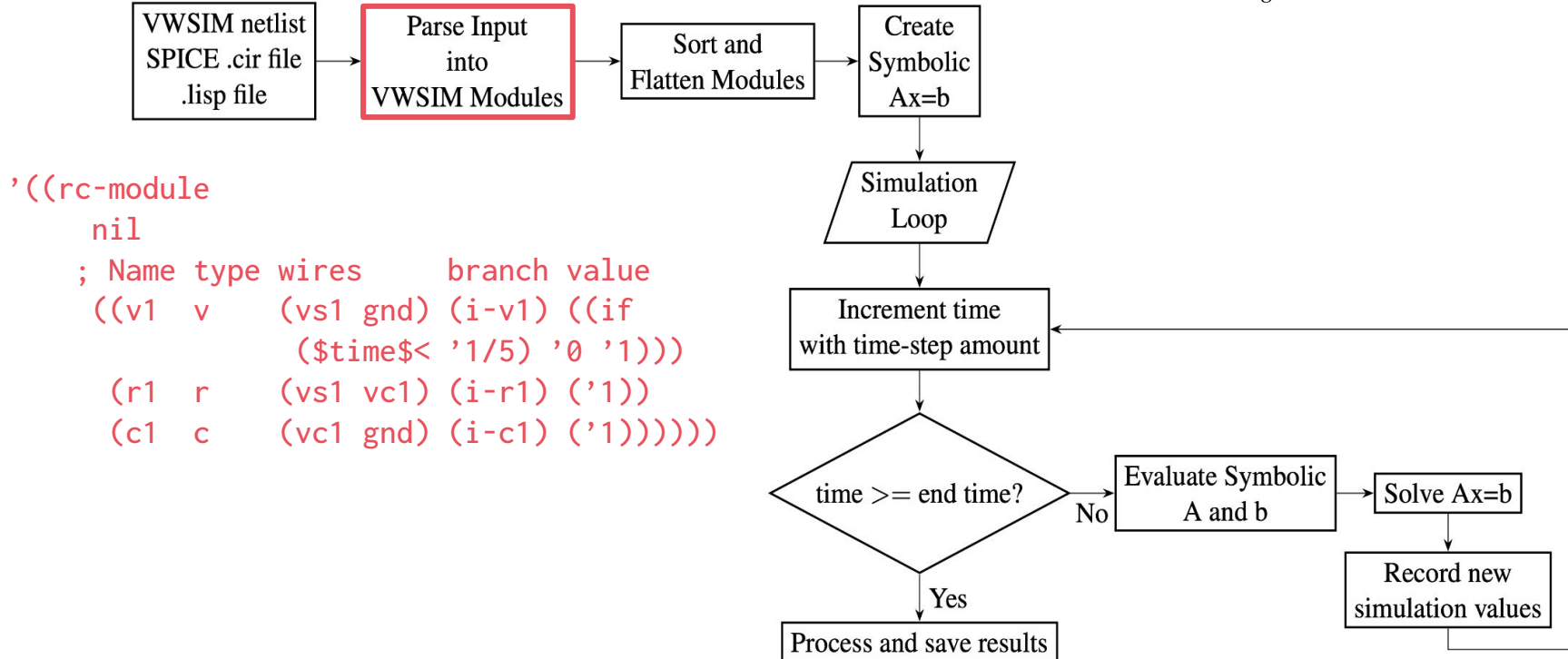
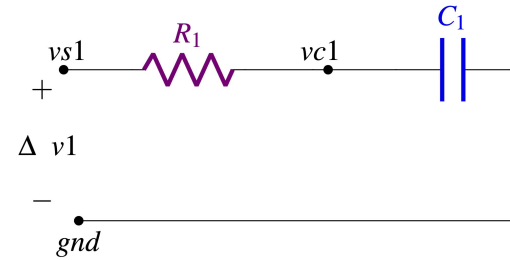
```
((($TIME$ 0.00 0.20 0.40 0.60 0.80 1.00 1.20 1.40 1.60 1.80)
(I-C1 0.00 0.91 0.74 0.61 0.50 0.41 0.33 0.27 0.22 0.18)
(VC1 0.00 0.09 0.26 0.39 0.50 0.59 0.67 0.73 0.78 0.82))
```



How VWSIM works

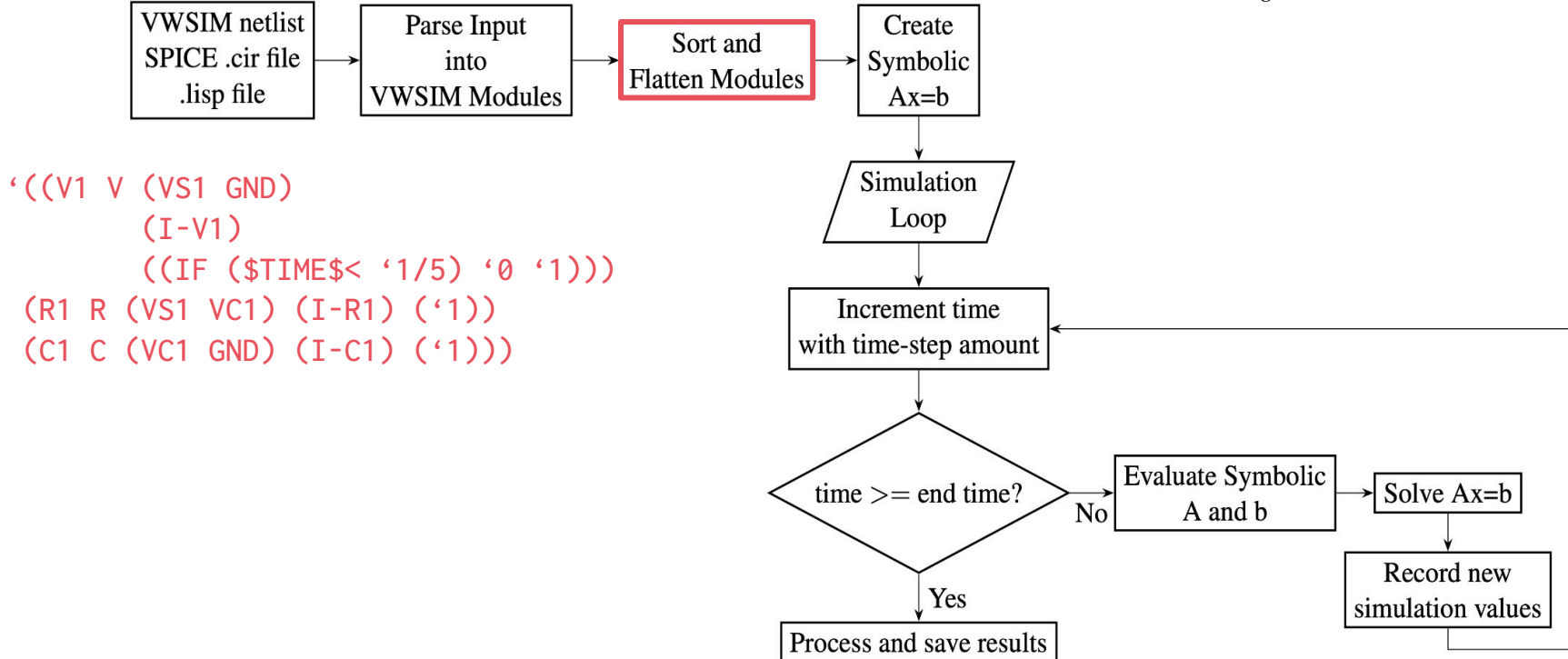
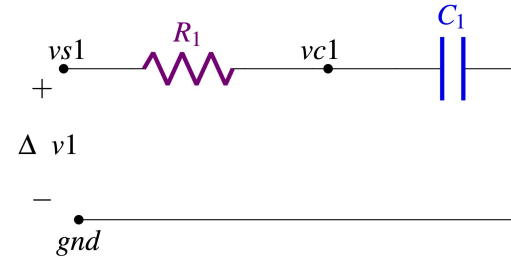


How VWSIM works



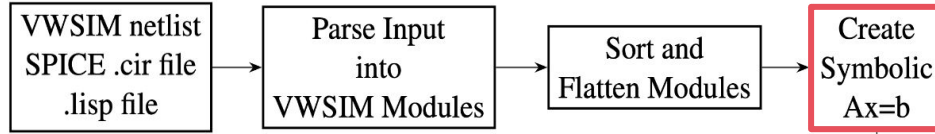
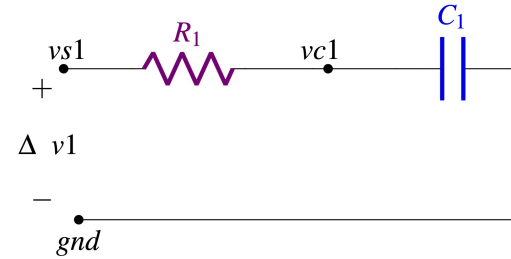
```
'((rc-module
  nil
  ; Name type wires      branch value
  ((v1  v      (vs1 gnd) (i-v1) ((if
    ($time$< '1/5) '0 '1)))
   (r1  r      (vs1 vc1) (i-r1) ('1))
   (c1  c      (vc1 gnd) (i-c1) ('1)))))
```


How VWSIM works



```
'((V1 V (VS1 GND)
(I-V1)
((IF ($TIME$< '1/5) '0 '1)))
(R1 R (VS1 VC1) (I-R1) ('1))
(C1 C (VC1 GND) (I-C1) ('1)))
```

How VWSIM works

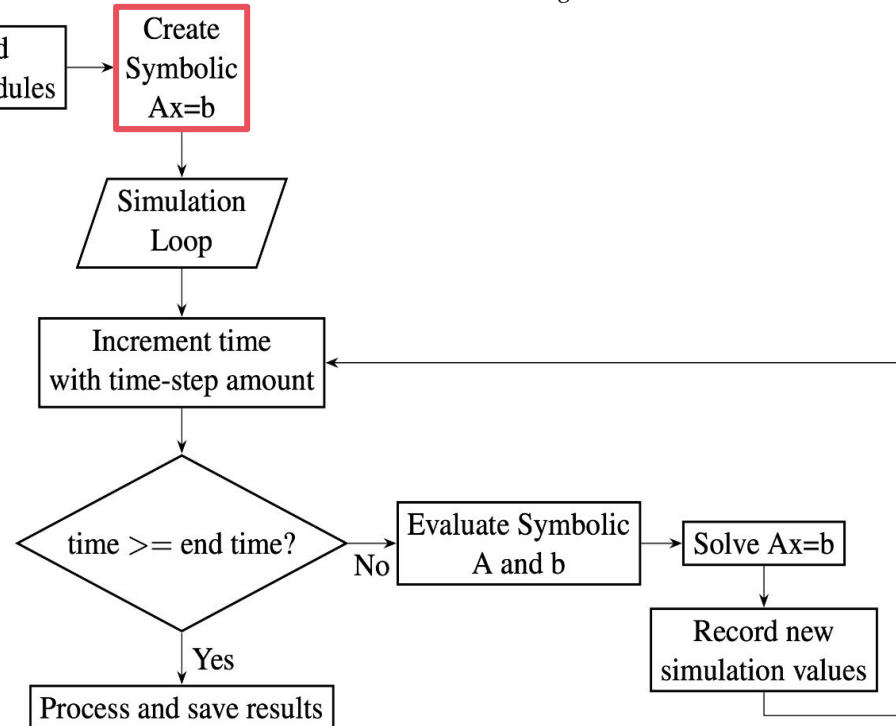


```

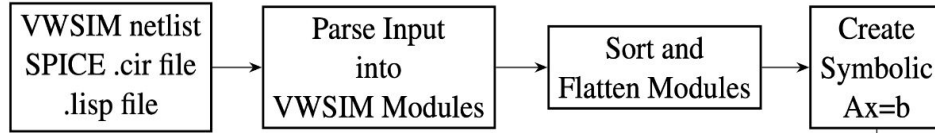
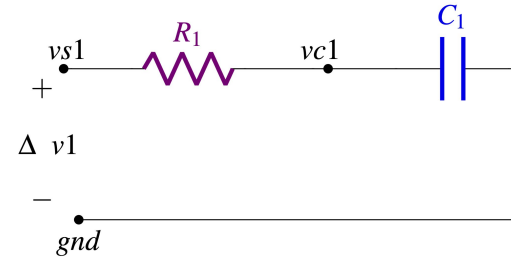
'(:A ((2 . '1))
  ((1 . (F0- (F/ $HN$ (F* '2 '1))))
  (3 . '1))
((0 . '1) (2 . F-1/X '1)
  (3 . (F0- (F-1/X '1))))
((1 . '1) (2 . (F0- (F-1/X '1)))
  (3 . (F-1/X '1)))
  
```

```

'(:B (IF ($TIME$< '1/5) '0 '1)
  (F+ (F- VC1 GND)
    (F* (F/ $HN$ (F* '2 '1)) I-C1))
  '0
  '0)
  
```



How VWSIM works

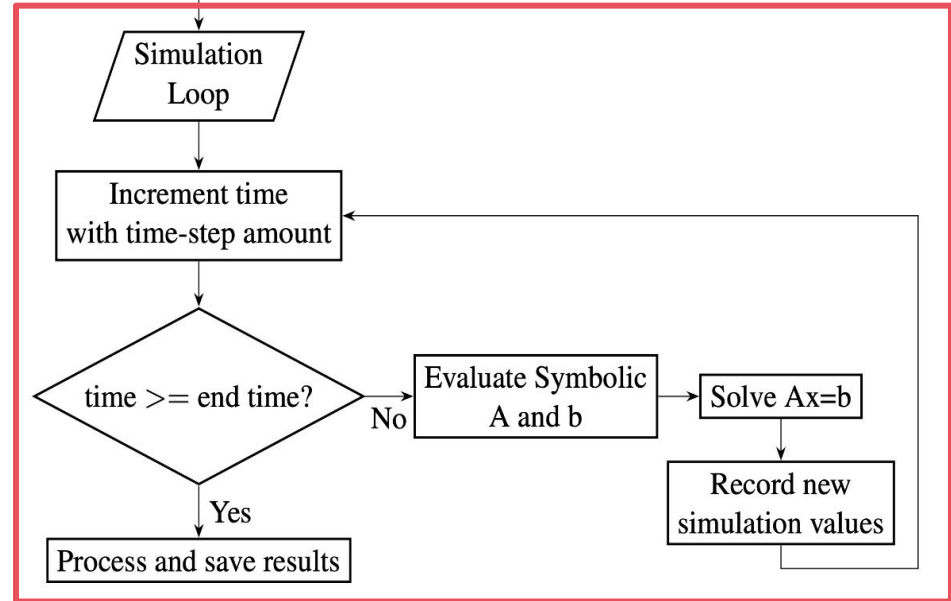


```

'(:A ((2 . '1))
      ((1 . (F0- (F/ $HN$ (F* '2 '1))))
      (3 . '1))
      ((0 . '1) (2 . F-1/X '1)
      (3 . (F0- (F-1/X '1))))
      ((1 . '1) (2 . (F0- (F-1/X '1)))
      (3 . (F-1/X '1))))
  
```

```

'(:B (IF ($TIME$< '1/5) '0 '1)
      (F+ (F- VC1 GND)
      (F* (F/ $HN$ (F* '2 '1)) I-C1))
      '0
      '0)
  
```



Running VWSIM

```
(vwsim <input>
  :sim-type <sim-type>
  :equations <equations>
  :spice-print <spice-print>
  :global-nodes <global-nodes>
  :time-step <time-step>
  :time-stop <time-stop>
  :time-start <time-start>
  :output-file <output-file>
  :concat-char <concat-char>
  :save-sim <save-sim>
  :save-sim-shorttp <save-sim-shorttp>
  :load-sim <load-sim>
  :save-var <save-var>
  :return-records <return-records>
)
```

See the paper and README for more details about each of these options

What have we proved?

1. Termination
2. Hundreds of guard proofs
 - a. We are currently working on guard verification of the $Ax=b$ solver
3. Some correctness properties

```
(defthm vw-eval-same-for-vw-eval-fold
  (implies (and (vw-eval-termp term)
                 (symbol-rational-list-alistp r)
                 (symbol-rational-list-alistp r-subset)
                 (record-subsetp r-subset r))
    (equal (vw-eval (vw-eval-fold term r-subset) r)
           (vw-eval term r))))
```

VWSIM optimizations

- Our first simulator was very slow (barely able to simulate a circuit with more than 10 circuit devices).
 - List-of-lists matrix representation and operations
 - Simulation results stored in list-of-lists format
- We have implemented the following optimizations:
 - Floating-point simulation
 - Sparse matrix representation
 - Array-based, sparse matrix solver
 - STOBJs for fast lookup and storage
 - Fast symbolic term evaluator

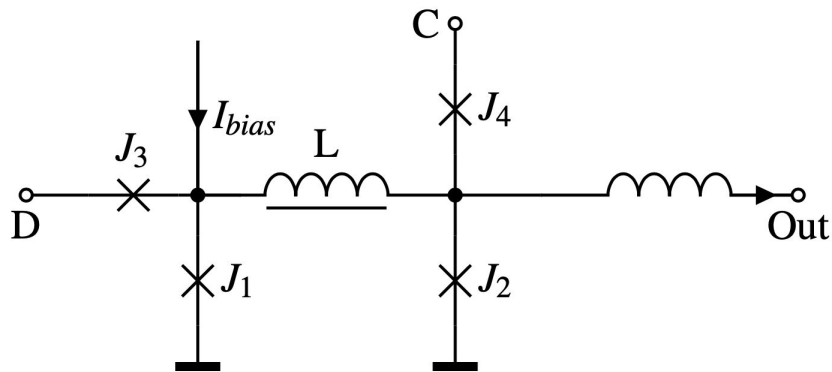
Floating-point simulation

- ACL2 does not currently support floating-point arithmetic
- We employ a *trick* to ensure the simulator can be defined in ACL2:

```
(defun nump (x)
  (declare (xargs :guard t))
  (and (acl2-numberp x)
       (zerop (imagpart x))))
```
- **nump** is equivalent to **rationalp** in the logic, but recognizes floating-point numbers in raw Lisp
- VWSIM exploits Common Lisp support for fast floating-point operations.

Example circuit and netlist

"D" latch circuit

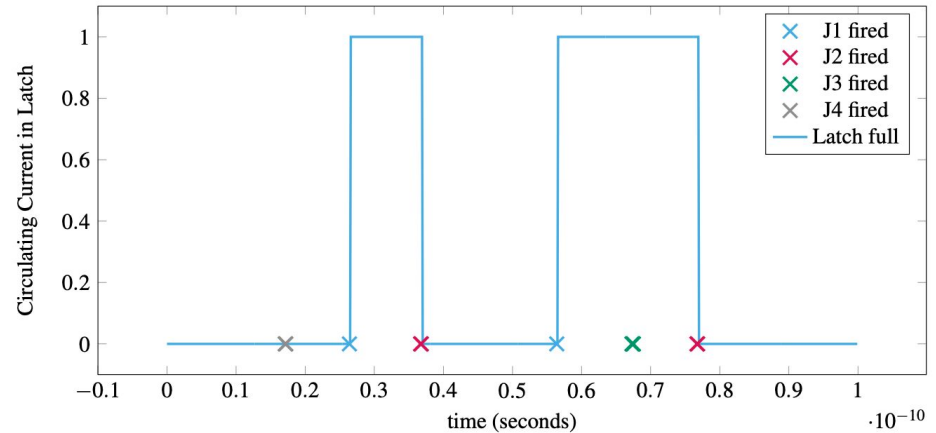
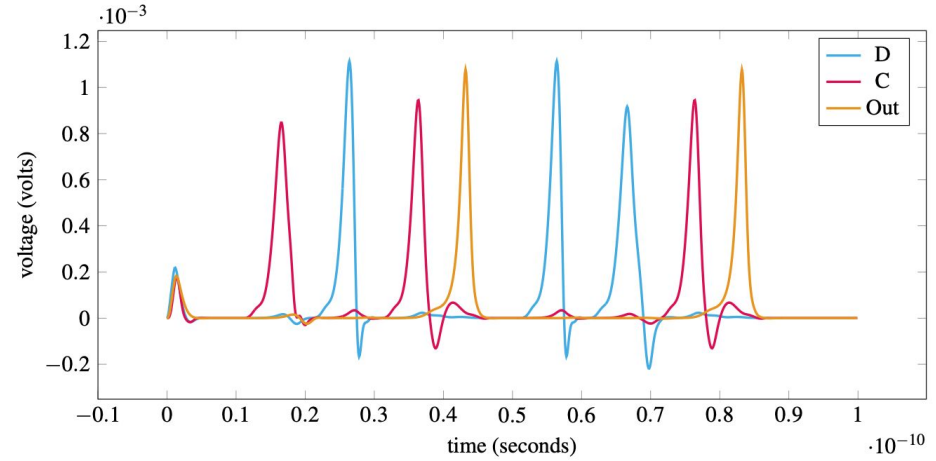
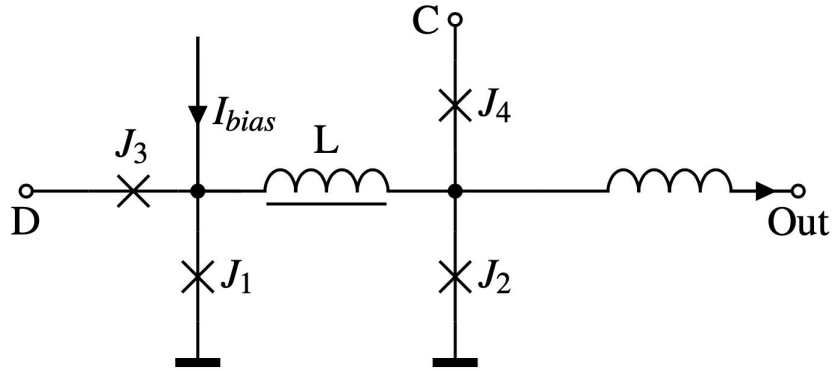


"D" latch VWSIM Netlist

```
(D_LATCH (D C OUT GND)
((LY L (NET@2 OUT)
  (LY)
  ('1/5000000000000))
(XJ2 DAMP_JJ (NET@2 GND))
(XJ4 DAMP_JJ (C NET@2))
(LL L (NET@1 NET@2)
  (LL)
  ('3/2500000000000))
(XBIAS1 BIAS (NET@1 GND))
(XJ1 DAMP_JJ (NET@1 GND))
(XJ3 DAMP_JJ (D NET@1))))
```


Example circuit simulation

"D" latch circuit



What next?

- Development has taken about $1\frac{1}{2}$ person-years
 - Initial definition, proofs, optimizations
- Run and test the simulator on many, many more circuits
 - Perform analysis on these circuits
- Guard verify the $Ax=b$ solver
- Improve VWSIM execution speed (currently 20% of state-of-the-art)
- Produce proofs of correctness for our RSFQ circuit designs



Conclusion

The development of the VWSIM simulator

- improved our understanding of RSFQ circuits
- enabled us to programmably test and validate circuit designs
- invigorated work on floating-point use and reasoning in ACL2
- is free-to-use and will be made available



Thanks!

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik and illustrations by Stories

