IDEAL: An Intelligent Design Environment for Asynchronous Logic

R. Manohar[1],S. Ataei[1], W. Hua[1], Y. Yang[1], K. Pingali[2], J. He[2], Y.-S. Lu[2], S. Maleki[2], M. Burtscher[3], A. Kothari[3]
[1]Yale University, [2]University of Texas at Austin, [3]Texas State University

Design & Security: Intelligent Design of Electronics Assets (IDEA)

Overview

Transistors get smaller and faster

causes

Communication delay via wires dominates that of local computation by transistors

causes

Clocked design paradigm becomes a poorer and poorer abstraction

solves

EDA for asynchronous circuits for modular, hierarchical design

causes

Designs become larger and larger

causes

Long turn-around time hinders design productivity

solves

Parallel EDA flow & its development framework

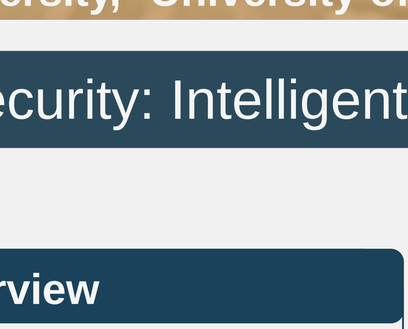
Goals: 24 hour turn-around time, no human-in-the-loop, asynchronous logic

Background

Asynchronous design

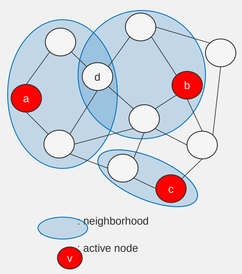
- FPGA: 3x faster
- CPU: 10x lower power
- GPS digital: 10x lower power
- Neuromorphic
  - 100x lower power
  - *Most large-scale neuromorphic projects use async logic*

Technology matured over 25 years, but minimal automation; no tool flow available.



Amorphous, run-time parallelism in Galois

- Active elements
  - Nodes/edges where computation is needed
- Operator
  - Computation at active elements
  - Neighborhood: read/write set
  - Morph operators: dynamic graphs
  - Label-computation operators for edge/node value change
- Schedule: ordering constraints

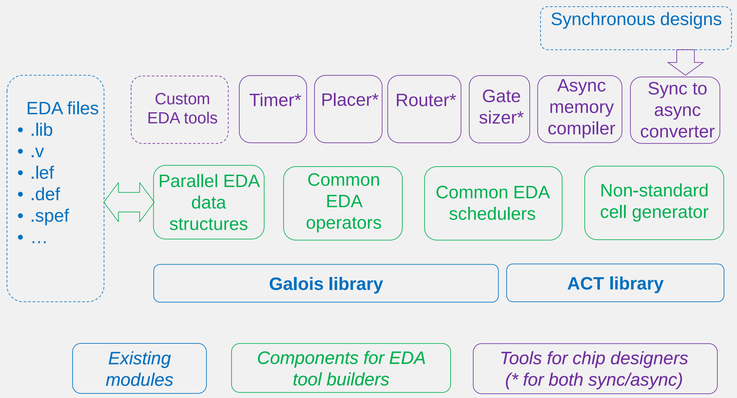


Approach

Tool development

- Build *difference* between synchronous flow and asynchronous circuit requirements
  - Timing analysis
  - Logic synthesis: import synchronous logic
  - Memory compiler
  - Placement respecting timing constraints
  - Global routing respecting timing
  - Global routing respecting timing
  - Borrow components from synchronous flow

Proposed Elements



Key Technical Challenges

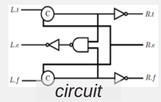
Challenge	Approach	Status
Many circuit families	Unified framework for timing and logic synthesis.	Theory done; first timer done.
Legacy IP	Automatic async import.	Single clock done.
Tool runtime	Parallel data structures; Galois-ed core algorithms.	Timing ready; placer, routing: preliminary.
Non-standard gates	Automated cell library generator.	Characterizer, transistor placement done.
Memory design	Memory compiler for pipelined SRAM.	65nm ready; 28nm & 12nm in development.
Fast mixed-mode simulation	Integrate with Xyce (Sandia).	First demonstration of Xyce/sim complete.

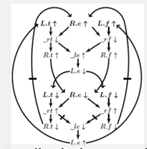
Results / Impact

Timing analysis

Design	#gates	#pins	Tseq (s)	Tpar (s)
tv80	47,641	231,497	47.99	5.59 (8.6x)
usb_funct	114,213	553,349	18.68	7.56 (2.5x)
syn1	211,900	1,017,855	949.9	59.1 (16x)
syn2	358,115	1,734,599	1290.5	98.6 (13x)
des_perf	652,188	3,160,501	10,543	875.8 (12x)

3rd place in TAU gate sizing contest (2019) using core timer

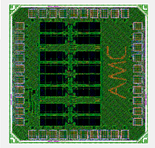




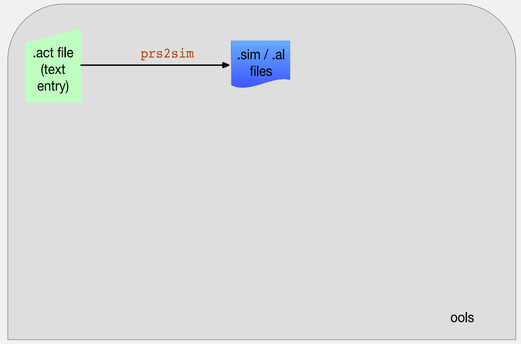
cyclic timing graph

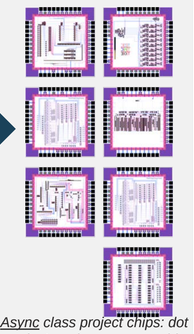
Memory

Size	F	Sync.	Async.
4K	0.6µm	26ns / 102mW	8.3ns / 75mW
16K	0.6µm	26ns / 168mW	11.5ns / 101mW
16K	65nm	*no open solution*	.49ns / 22.8mW



Custom flow





Async class project chips: dot product engine, xor-scrambler, CPU

Current Status

- Github releases
  - Galois: https://github.com/IntelligentSoftwareSystems/Galois
  - ACT: http://github.com/asyncvlsi/act
  - AMC: http://github.com/asyncvlsi/AMC
- Known external users
  - 3 companies, 2 research groups

Contact

Technical Point of Contact:
Prof. Rajit Manohar <rajit.manohar@yale.edu>
Electrical Engineering and Computer Science
Yale University

The research described above was supported with funding from the Defense Advanced Research Projects Agency (DARPA).
The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.