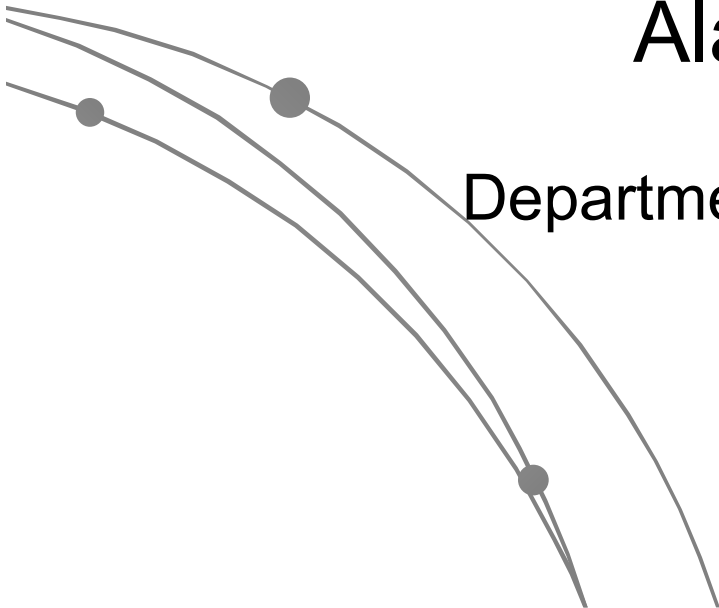


# ABC: The Way It Should Have Been Designed

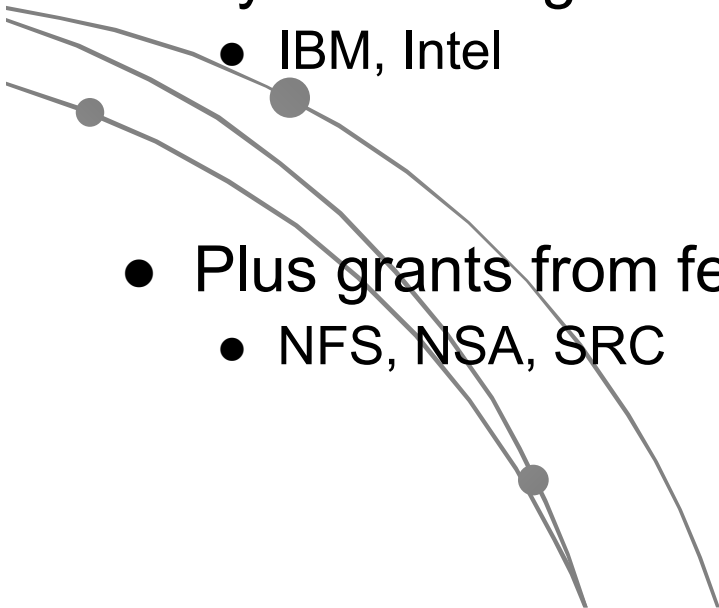
Alan Mishchenko

Department of EECS, UC Berkeley



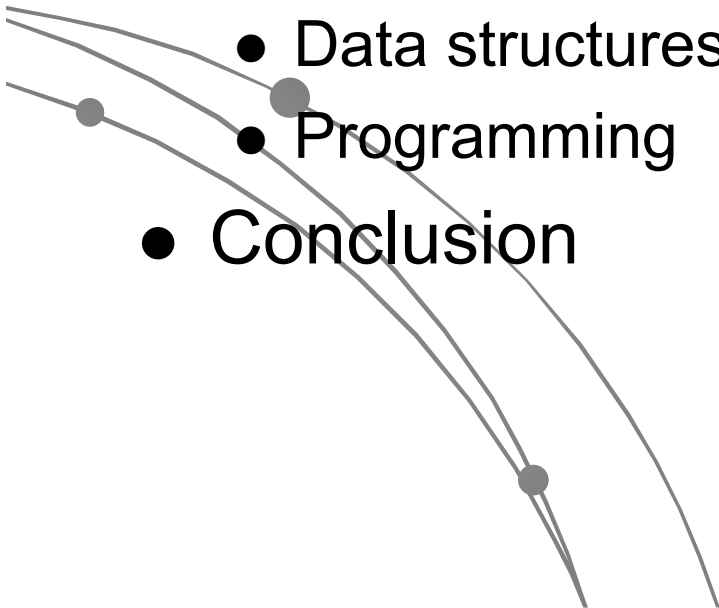
# Industrial Supporters (since 2005)

- CAD tool companies
  - Synopsys, Mentor, Cadence, Verific, Magma (Synopsys), Atrenta (Synopsys), Jasper (Cadence), Oasys (Mentor)
- FPGA companies
  - Xilinx, Altera, Synplicity (Synopsys), Actel (Microsemi), Abound Logic (Lattice), Tabula
- System design companies
  - IBM, Intel
- Plus grants from federal and industrial funding agencies
  - NFS, NSA, SRC

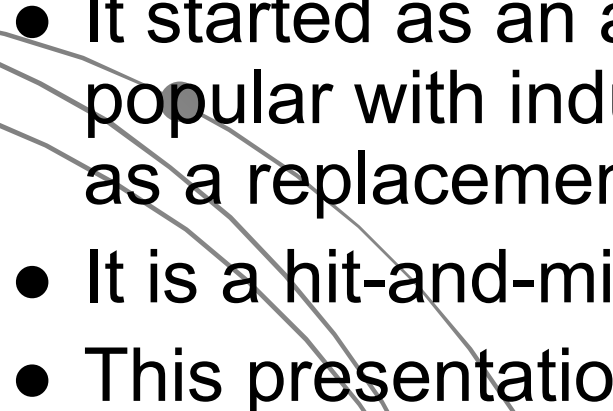


# Overview

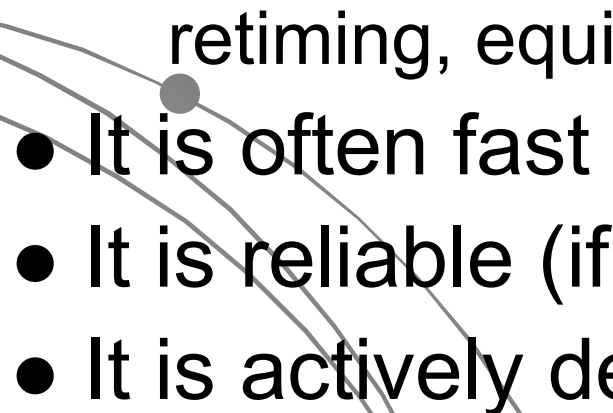
- Introduction
- Hits and misses
- Lessons learned
  - Front-end and back-end
  - Optimization flow
  - Data structures
  - Programming
- Conclusion



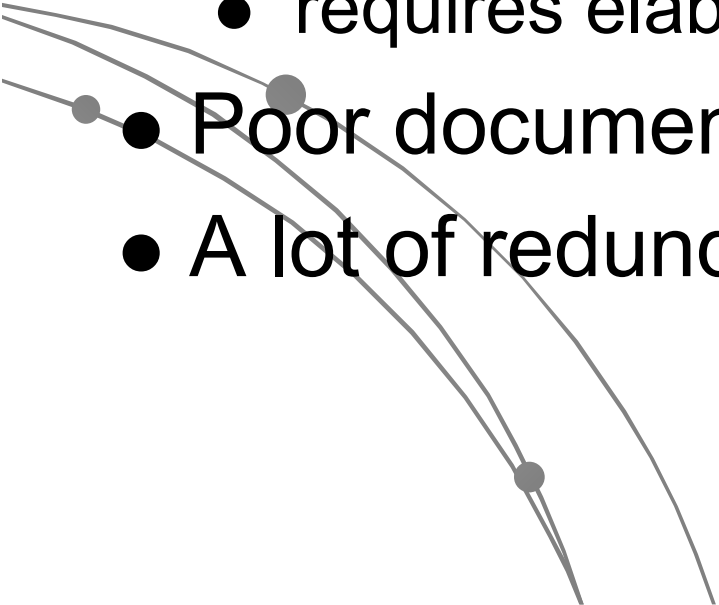
# Introduction

- Disclaimer: This presentation may be boring if one does not develop or does not consider developing an “industrial-strength academic tool” such as ABC
  - ABC has a 15-year history
  - It started as an academic tool and soon became popular with industry (especially with start-ups) as a replacement for SIS
  - It is a hit-and-miss in terms of its usefulness
  - This presentation shares the lessons learned
- 

# ABC Hits

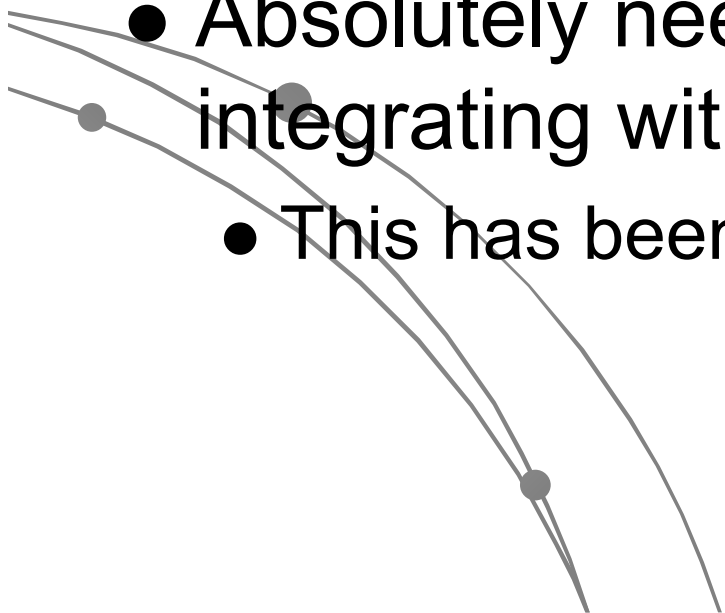
- It is based on what we believe to be cutting-edge research ideas
  - It offers a low-cost and often competitive implementation of fundamental algorithms
    - AIG rewriting, tech-mapping, SAT sweeping, retiming, equivalence checking, etc
  - It is often fast and low-memory
  - It is reliable (if we use it in a known way)
  - It is actively developed and supported
- 

# ABC Misses

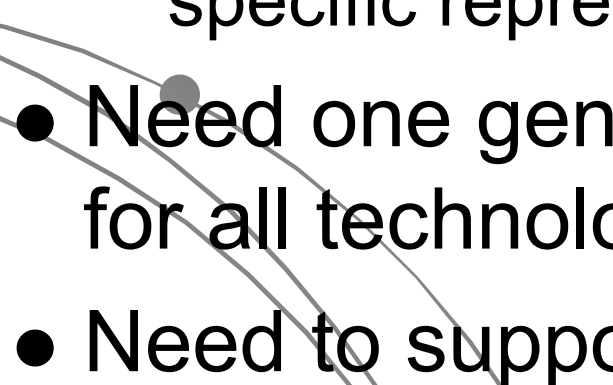
- Inadequate Verilog parser
  - Does not natively support much of the “industrial stuff” (complex flops, multiple clocks, memories, design constraints, etc)
    - requires elaborate workarounds to be useful
  - Poor documentation
  - A lot of redundant source code
- 

# Lessons: Front-End and Back-End

- Having a variety of formats is useful, but...
- Reading and writing Verilog is a must!
  - If a general-enough Verilog parser cannot be developed, integrate with Yosys
- Absolutely need well-documented APIs for integrating with external tools!
  - This has been addressed to some extent

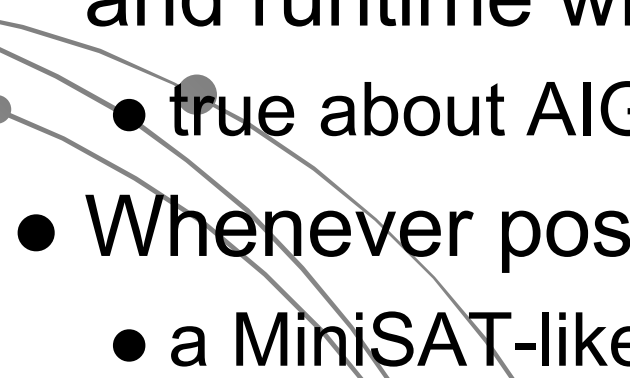


# Lessons: Optimization Flow

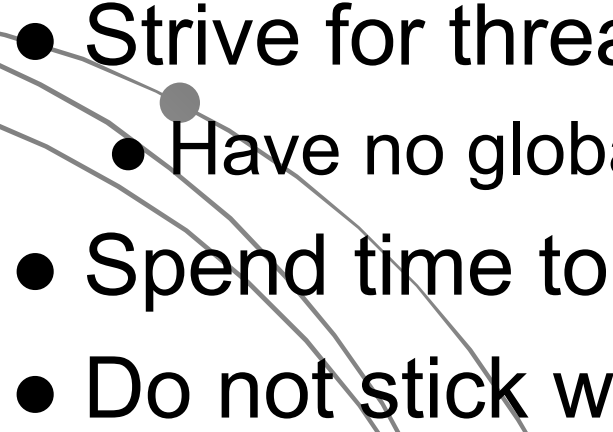
- AIG is a good unifying data-structure
    - Do not hesitate to base computations on AIGs
  - Need parametrizable optimizers
    - Rather than having optimizations geared to a specific representation (AIG/MIG/XMIG/etc)
  - Need one generic cut-based tech-mapper for all technologies (gates, LUTs, etc)
  - Need to support the “industrial stuff”!
- 



# Lessons: Data Structures

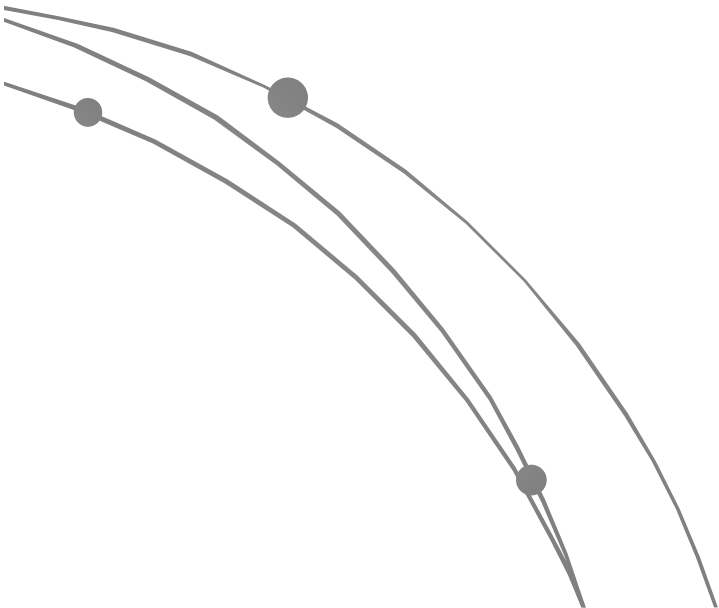
- Develop a clean minimalistic data-structure for each package (conversions between data-structures are easy and fast)
  - Reduce memory for large data-structures and runtime will be reduced
    - true about AIG, logic network, hierarchical netlist
  - Whenever possible, use 32-bit integers
    - a MiniSAT-like SAT solver is a good example
- 

# Lessons: Programming

- Strive for maintainability
    - Minimize dependency between packages
  - Strive for reproducibility
    - Implement your own floating point number
  - Strive for thread-safety
    - Have no global and static variables
  - Spend time to build a set of handy tools
  - Do not stick with C (can mix C and C++)
- 

# Conclusion

- Talked about ABC
- Reviewed gains and losses
- Learned from past mistakes - hopefully 😊



# Abstract

- Twelve years ago, in September 2005, the first public version of ABC was released. It featured technology-independent synthesis by DAG-aware rewriting, technology mapping for standard cells and lookup tables, and simple combinational equivalence checking, all based on the And-Inverter Graphs (AIG) data-structure used to unify the computation flow. In the coming years ABC has been adopted as an optimization engine and a research environment by a number of academic and industrial users. The use that followed exposed a number of shortcomings in the original design of ABC. This talk focuses on what is present and, more importantly, what is missing in ABC, and how ABC could be redesigned to make it more versatile and user-friendly. The motivation for this talk is to help academic researchers maximize the usefulness of their tools and set a new standard for future versions of ABC.