Logic synthesis in a nutshell

Revisit ABC from scratch with basics

Jingren Wang, May 7th, 2025

THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)

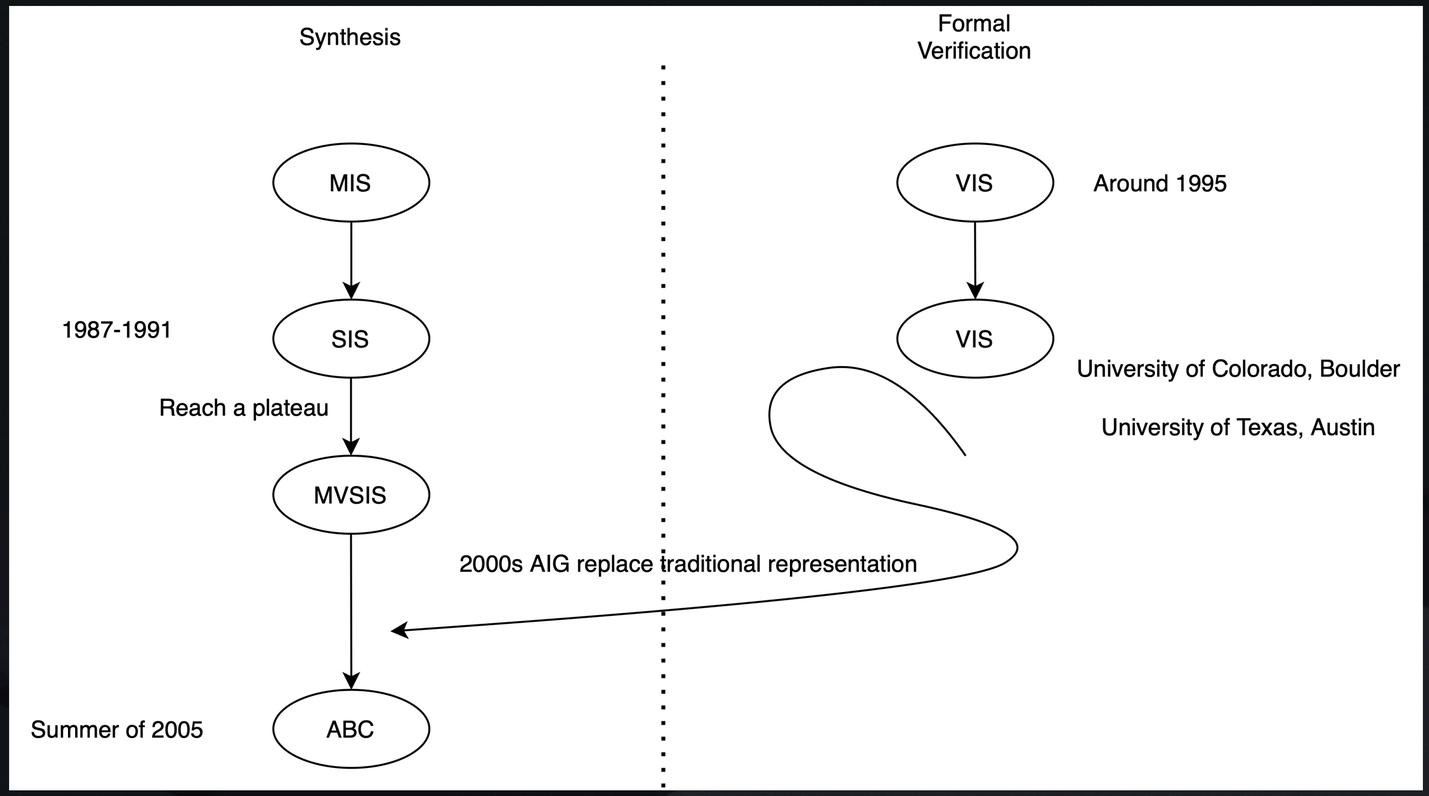
Overview

See this talk as a review!🧐

- History of UCB Synthesis and Verification toolset
- Where, when to do logic synthesis
- The advantage and drawback of ABC
- AIG, Cut, Window, Simulation, Choices, Don’t care etc.

A bit of history

Espresso? ☕



Ref: https://people.eecs.berkeley.edu/~alanmi/publications/2010/cav10_abc.pdf

AIG

Just the one wildly used!🤔

- Merge View/Separate view/General node/Discrete node/Generic node

Task: Check how to do it manually: https://github.com/arminbiere/aiger

- Based on the problem scale: Truth table->SOP->BDD->AIG

Task: Permutation on TT: https://wjrforcyber.github.io/files/Kitty_Permutation.pdf

- Truth table/SOP/BDD are not discarded.
- Canonical or not/what is under the hood
- Other IR(GIG): MIG/XMG/XAG...
- Even more: semi-tensor product

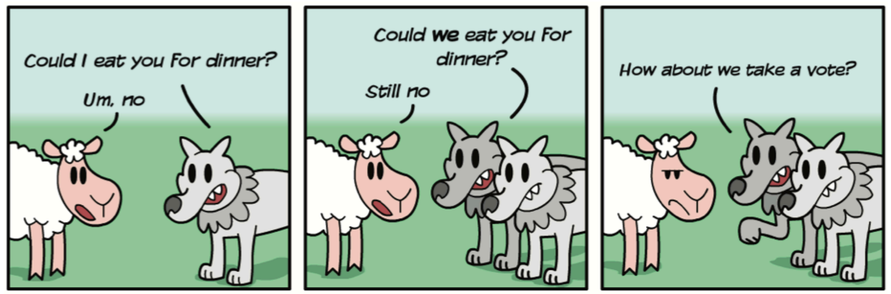


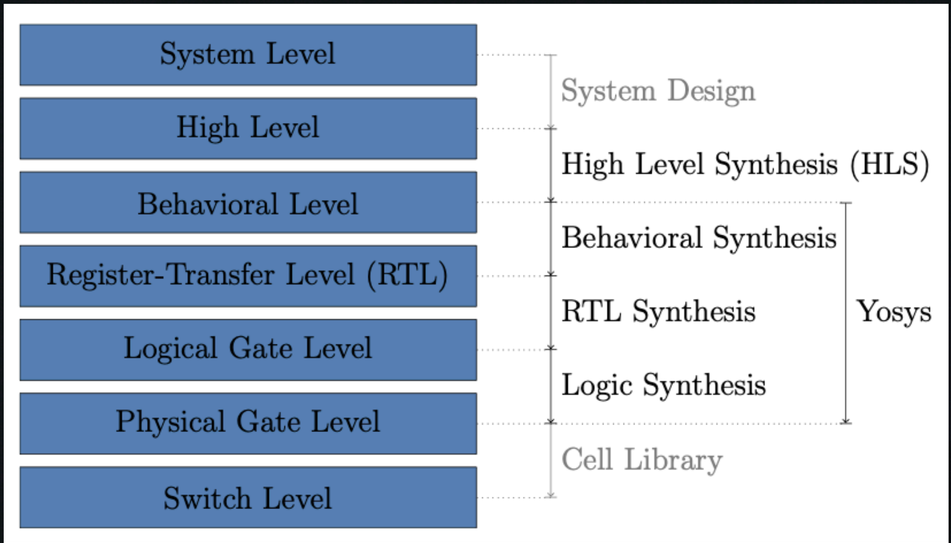
Figure 1.4 – Explanation of the majority function: the majority function of three inputs evaluates to true if and only if at least two of the three inputs are true. Source *http://redpanels.com/36/*

Ref: Data Structures and Algorithms for Logic Synthesis in Advanced Technologies

$$MAJ3(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$$

Where/When

From Yosys perspective!📁



We are only concentrating on logic synthesis.

Ref: https://yosys.readthedocs.io/_/downloads/en/latest/pdf/

AIG

Always structural hashed!🥽

- No structural hash.
- One-level structural hash.
- Two-level structural hash.

$(a \vee b) \wedge (b \vee d)$ $(a \wedge d) \vee b$

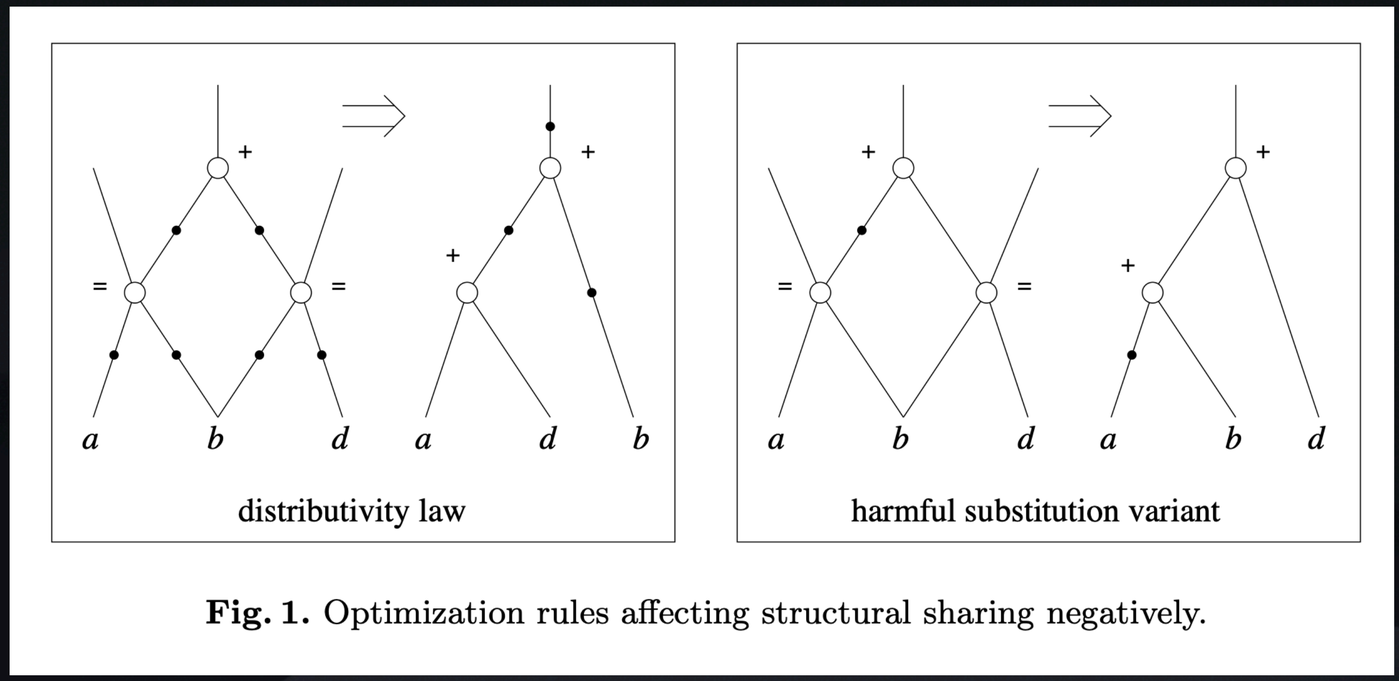


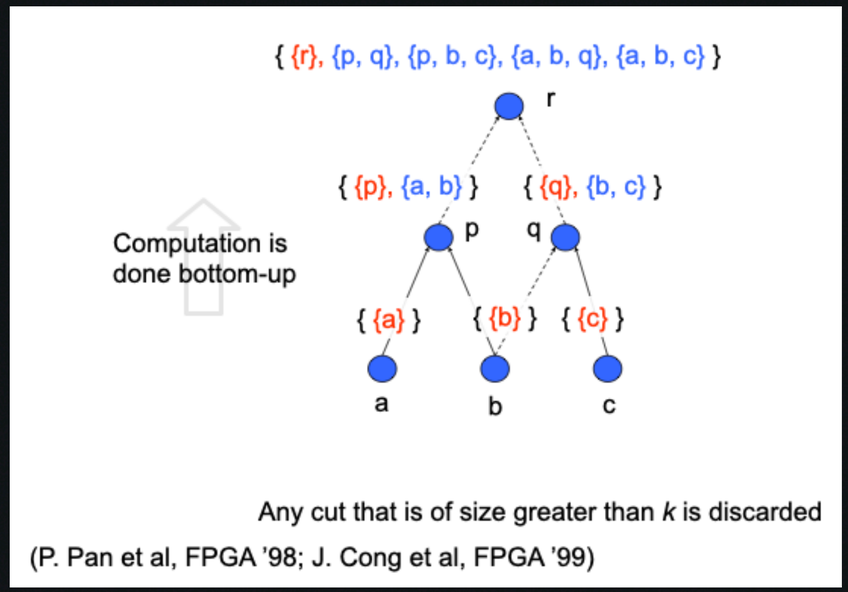
Fig. 1. Optimization rules affecting structural sharing negatively.

Ref: Local Two-Level And-Inverter Graph Minimization without Blowup

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut



Ref: https://people.eecs.berkeley.edu/~alanmi/presentations/priority07.ppt

Cartesian product

$$A \bowtie B \equiv \{u \cup v \mid u \in A, v \in B, \mid u \cup v \mid \leq k\}$$

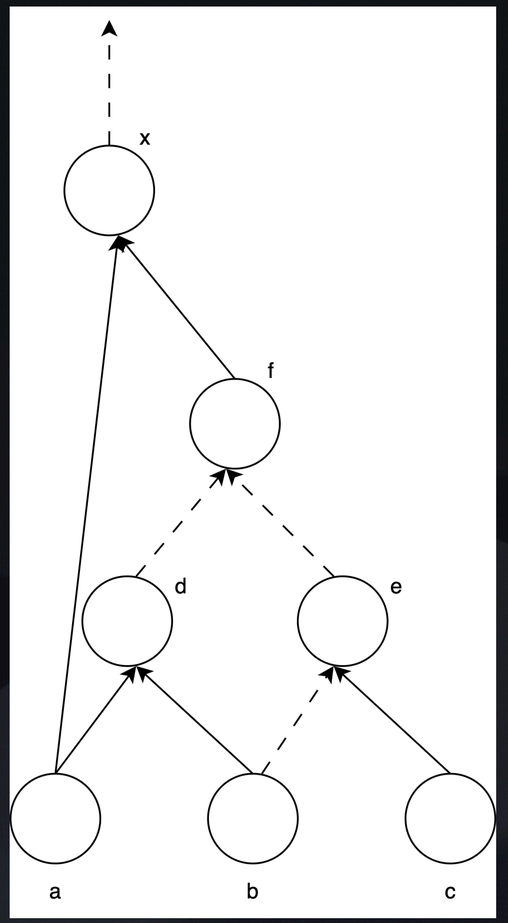
$$\Phi(n) \equiv \begin{cases} \{\{n\}\}, & \text{if } n \text{ is a PI} \\ \{\{n\}\} \cup \left(\Phi(n_1) \bowtie \Phi(n_2)\right), & \text{otherwise} \end{cases}$$

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut

Why dominated?



$\{a, d, b, c\}$
is *dominated* by
$\{a, b, c\}$

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut

In ABC, how does it do detect on duplication/ dominance/feasibility:

Signatures-encode in bit level.

$$sign(C) = \sum_{n \in C} 2^{ID(n) \bmod M}$$

Given $M = 8$, cut $\{32,68,69\}$, cut $\{32,68,70\}$, cut $\{36,64,69\}$, calculate the signature of each one of them.

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut

What's the problem of K-feasible cut?

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut

Some extra term:

1. Term *factor:* A parenthesised representation of a tree network which has internal gate AND and OR.

The expression

$$F = a \cdot c + a \cdot d + b \cdot c + b \cdot d + e$$

can be factored into

$$F = (a + b) \cdot (c + d) + e$$

Ref: Electronic Design Automation: Synthesis, Verification, and Test (Systems on Silicon)

2. FFLC(Factored Form Literal Count)

Task: Try find out how FFLC is related to AIG structure.
Ref: Improving Standard-Cell Design Flow using Factored Form Optimization

3. Term *cofactor*.

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut

Factor tree/tree nodes/dag nodes

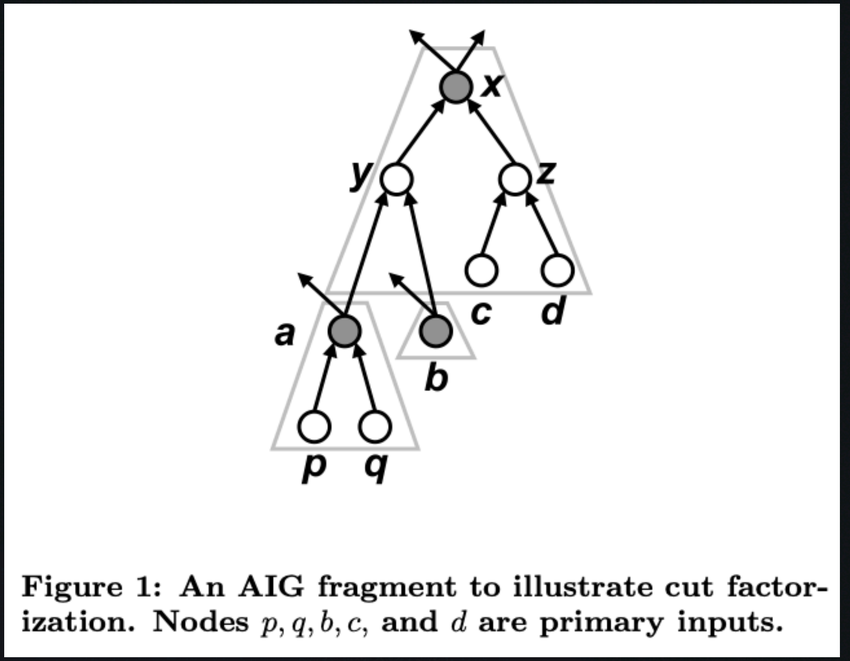


Figure 1: An AIG fragment to illustrate cut factorization. Nodes $p, q, b, c,$ and d are primary inputs.

Ref: Factor cuts https://people.eecs.berkeley.edu/~alanmi/publications/2006/iccad06_cut.pdf

1-step expansion: $\{a, b, z\}$ to $\{p, q, b, z\}$

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut(CF) { Tree cuts (Local cuts) / Reduced cuts (Global cuts) }
- Priority cut
- Reconvergence-driven cut

Tree cuts

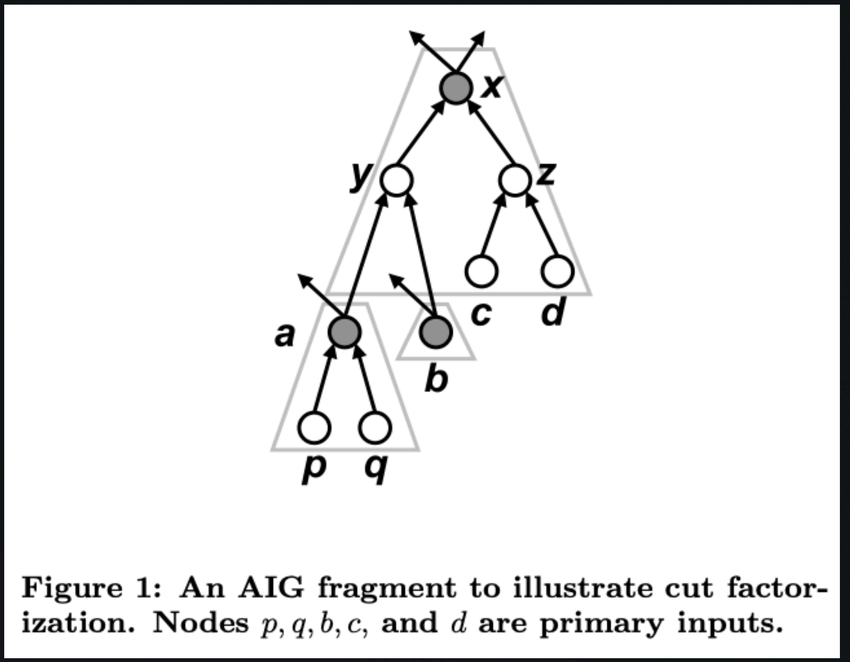


Figure 1: An AIG fragment to illustrate cut factorization. Nodes $p, q, b, c,$ and d are primary inputs.

Very explicit, the cut has boundaries.

$$\Phi_\Gamma(n) \equiv \begin{cases} \{\{n\}\}, & \text{if } n \text{ is a PI} \\ \{\{n\}\} \bigcup \left(\Phi_\Gamma^\dagger(n_1) \bowtie \Phi_\Gamma^\dagger(n_2) \right), & \text{otherwise} \end{cases}$$

Auxiliary function: $\Phi_\Gamma^\dagger(n) \equiv \begin{cases} \varnothing, & n \in F \\ \Phi_\Gamma(n), & \text{otherwise} \end{cases}$

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut(CF) { Tree cuts (Local cuts) / Reduced cuts (Global cuts)
- Priority cut
- Reconvergence-driven cut

Reduced cuts

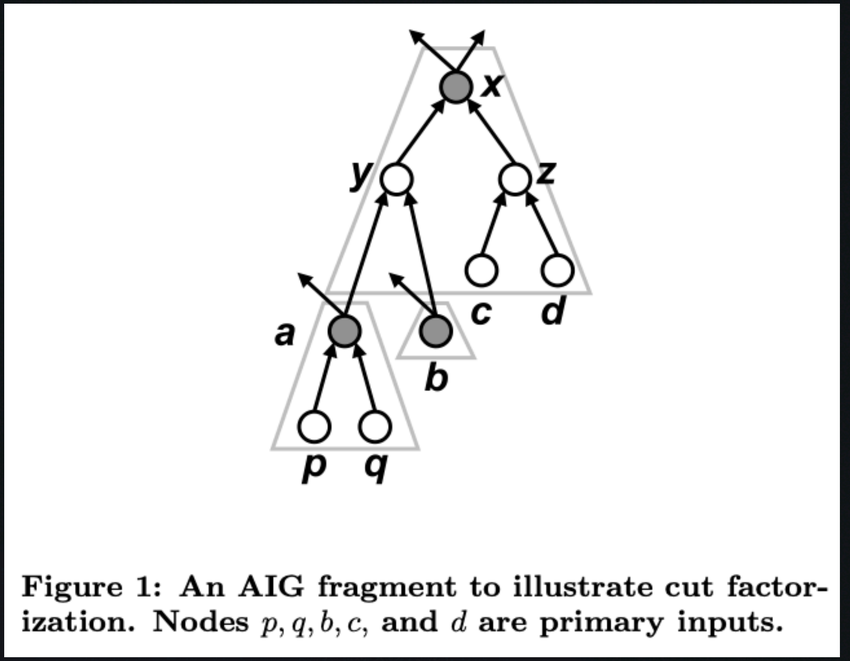


Figure 1: An AIG fragment to illustrate cut factorization. Nodes $p, q, b, c,$ and d are primary inputs.

Very explicit, the cut has boundaries.

$$\Phi_\rho(n) \equiv \begin{cases} \{\{n\}\}, & \text{if } n \text{ is a PI} \\ \{\{n\}\} \bigcup \left(\Phi_\rho(n_1) \bowtie \Phi_\rho(n_2) \backslash \Phi_\Gamma(n) \right), & \text{otherwise} \end{cases}$$

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut(PF)
  - Leaf-dag cuts (Local cuts)
  - Dag cuts (Global cuts)
- Priority cut
- Reconvergence-driven cut

Task: Try find out why we need this? What's the main difference?

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut

Two ways of generating cuts: Bottom-up(traditional way of generating cut) and Top-down(factor cut).

Ref: Data Structures and Algorithms for Logic Synthesis in Advanced Technologies

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut → Just cuts with *cost function*.
- Reconvergence-driven cut

AIG

Deep Dive into cuts!✂️

- Cut
- K-feasible cut
- Factor cut
- Priority cut
- Reconvergence-driven cut

Maintain as few growth on leaves as possible while extending cut volume

AIG

MFFC and why it's important!🪟

- MFFC = Maximum Fanout Free Cone
- Remove the target node = remove its MFFC
- MFFC could be extend to MFFW (Maximum Fanout Free Window)

In ABC, how does it get MFFC of each node?

Reference/Dereference - A general technique to collect the support variables and nodes in the MFFC.

*Check the command *print_mffc*.

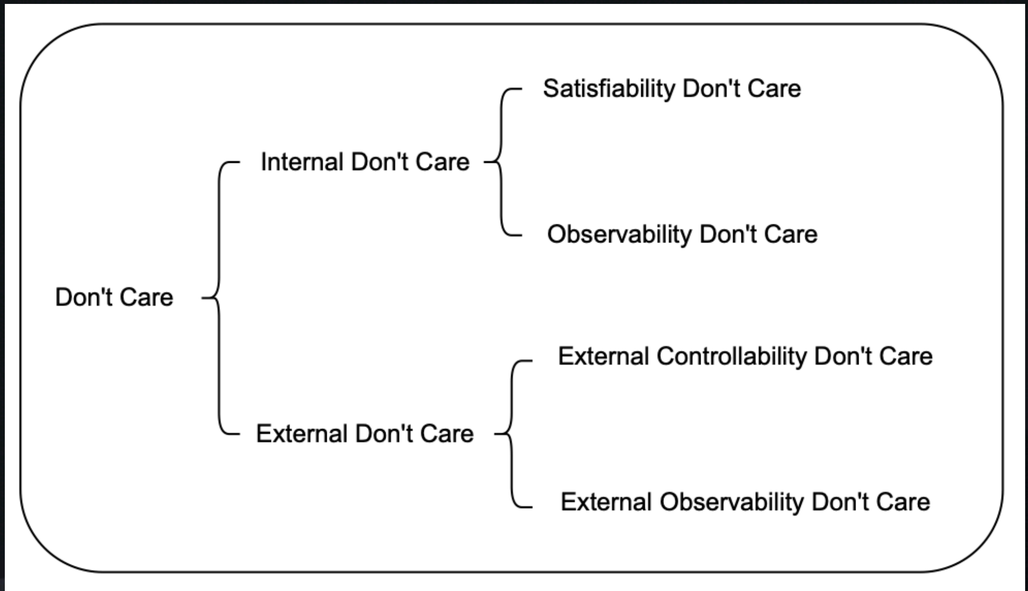
AIG

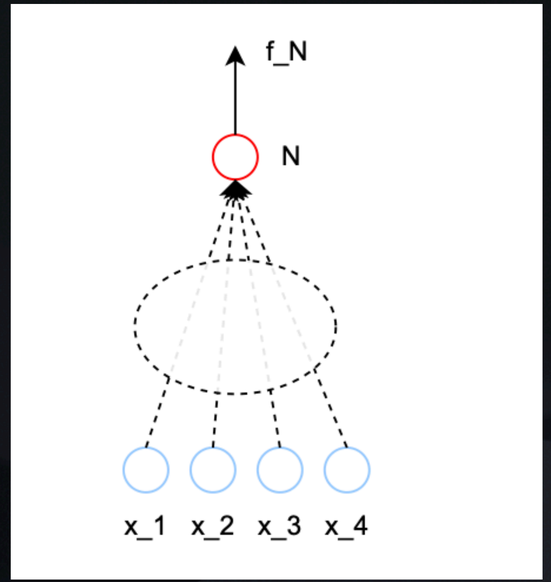
MFFC and why it's important!🪟

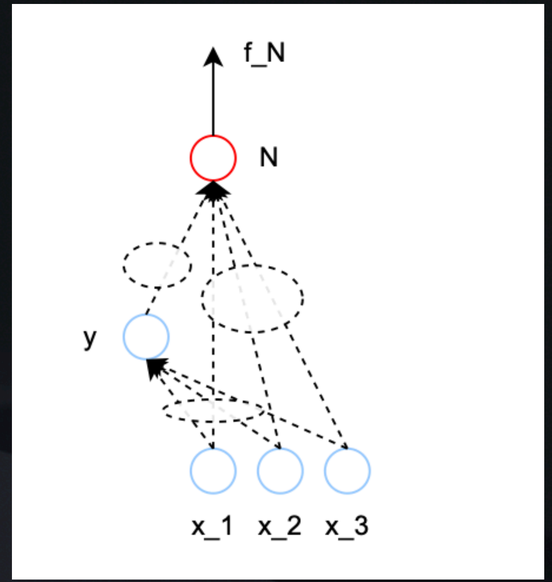
- MFFC = Maximum Fanout Free Cone
- Remove the target node = remove its MFFC
- MFFC could be extend to MFFW (Maximum Fanout Free Window)

Don’t care

Redundancy removal!🧹







Ref: Don’t care cheatsheet https://wjrforcyber.github.io/files/DONTCARE.pdf

Task: Try find out definition of external don’t care.

Task: Check the definition of .exdc in BLIF format, which DC does it indicate?

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

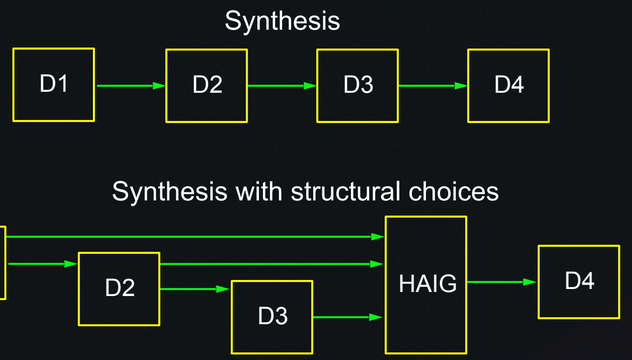
Resub

Check what is under the hood!🔬

- A brief introduction on ***resub***, and why it is important.
- Put effort on basics not algorithms.

AIG Choices

Extend exploration space!🔍



Ref: https://people.eecs.berkeley.edu/~alanmi/presentations/abc_2024.pptx

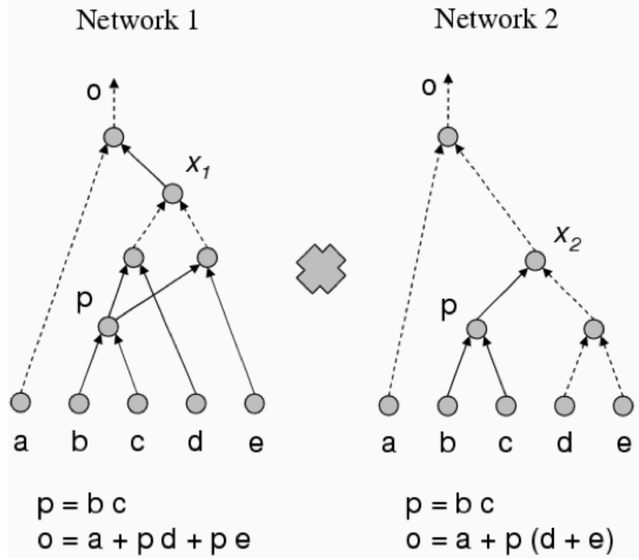


Figure 1. Equivalent networks before choicing.

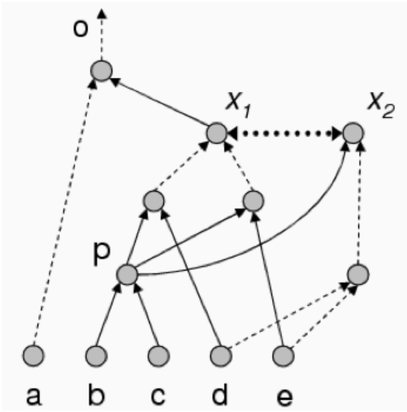


Figure 2. The choice network.

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

FRAIG

Functionally reduced AIG!🔍

- If the equivalent nodes accumulated, then choices are made.

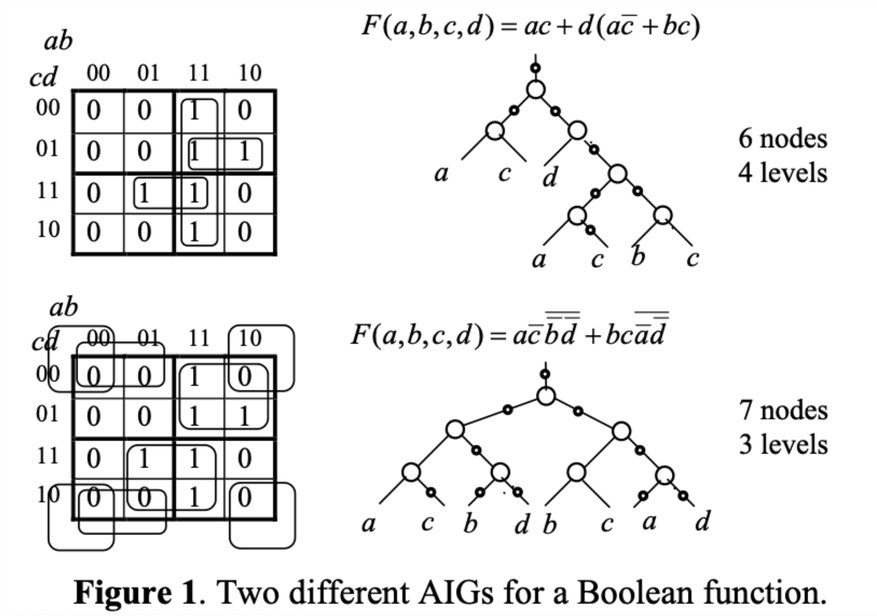


Figure 1. Two different AIGs for a Boolean function.

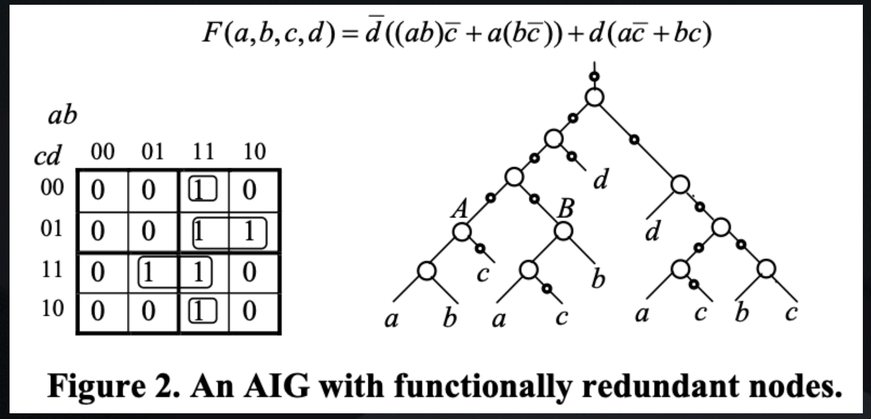


Figure 2. An AIG with functionally redundant nodes.

Task: Check out ***fraig_store*** *and* ***fraig_restore*** *in ABC.*

AIG Dangling nodes

Use choices with care! 👀

Dangling nodes are nodes without fanout in AIG, eliminate dangling nodes won't hurt the network equivalence.

ABC always contains a compact AIG intermediate representation, think about how does it affect the use of choices?

A bit on mapping

The gap still exists!🫠

- A view on the whole design from industry level.
- Lack of bridging method between separate views.
- Critical path, mapping without sizing, high-fanout issue, retiming...

ABC

Advantage of SOTA Synthesis Tool!🔧

- Contain most of the SOTA optimisation method.
- Low memory footprint.
- Ongoing research and maintenance.
- Most of the startups using ABC as a prototype.

ABC

Drawbacks of SOTA Synthesis Tool!🔧

- Poor Verilog support.
- Limited sequential elements support.
- Poor documentation and tons of redundant code.
- Although open source, maintaining the whole repo is hard.

A collection

Commands with papers!📖

- Me myself maintain a repository of ABC commands and related paper https://github.com/wjrforcyber/ABCPaperCheck
- Another recommendation: mockturtle: https://github.com/lsils/mockturtle
- A handy tool that helps with tests on AIGs: https://github.com/arminbiere/aiger

Alice's Adventures in Wonderland

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Ideas/Bugs and discussion

Do drop me emails!📧 *jingrenwangcyber@gmail.com*

1. Email is the most preferred way to get in touch.
2. MRE(Minimum Reproducible Example) is preferred.
3. Separate the dependencies between packages.
4. Join the discussion, expose problems, but always try by yourself before asking others.

Thanks!

THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)