The Logic Synthesis Homework Is Not Done Yet

Petr Fišer, Jan Schmidt

Dept. of Digital Design
Faculty of Information Technology
CTU in Prague, Czech Republic
It started with...

- ... *cordic* benchmark circuit, structural description
- LUT mapping by ABC ('dch; if; mfs' script)

```
.model cordic
.inputs x0 ey1 a3 ey0 ex1 a2 x1 ey2
   a5 a4 y2 ex2 v a6 ex0 x2 z2 z0 x3
   z1 y3 y1 y0
.outputs d dn
.gate nor2  a=ex2 b=ey0  O=n25
.gate inv1  a=ex1  O=n26
.gate and2  a=n25 b=n26  O=n27
.gate nor2  a=ey2 b=ey1  O=n28
...
```

```
.model cordic
.inputs y3 a3 ey0 x1 ex0 x3 a2 z2 x0
   y2 y0 ey2 ey1 y1 a4 z1 ex2 a5 a6
   z0 v x2 ex1
.outputs dn d
.gate nor2  a=ex2 b=ey0  O=n25
.gate inv1  a=ex1  O=n26
.gate and2  a=n25 b=n26  O=n27
.gate nor2  a=ey2 b=ey1  O=n28
...
```
It started with...

- ... cordic benchmark circuit, structural description
- LUT mapping by ABC ('dch; if; mfs' script)

```
.model cordic
.inputs x0 ey1 a3 ey0 ex1 a2 x1 ey2
  a5 a4 y2 ex2 v a6 ex0 x2 z2 z0 x3
  z1 y3 y1 y0
.outputs d dn
.gate nor2  a=ex2 b=ey0  O=n25
.gate inv1  a=ex1  O=n26
.gate and2  a=n25 b=n26  O=n27
.gate nor2  a=ey2 b=ey1  O=n28
...
```

---

```
.model cordic
.inputs y3 a3 ey0 x1 ex0 x3 a2 z2 x0
  y2 y0 ey2 ey1 y1 a4 z1 ex2 a5 a6
  z0 v x2 ex1
.outputs dn d
.gate nor2  a=ex2 b=ey0  O=n25
.gate inv1  a=ex1  O=n26
.gate and2  a=n25 b=n26  O=n27
.gate nor2  a=ey2 b=ey1  O=n28
...
```

36 LUTs

390 LUTs
Already known fact

Synthesis is not immune to neutral details in the input

- results of different quality
  - size
  - delay
  - power
  - ...

- semantically unimportant changes
  - ordering of variables
  - ordering of HDL statements
  - signal names
  - ...

How seriously?

- units of percent reported
  [Puggelli, Welp, DAC'11]
- reality is much worse
  [Fišer, Schmidt, IWLS'11]

Why it happens?

- local search
- iterative heuristics
- greedy nature
  $\Rightarrow$ depends on lexicographic order
Outline

- Random behavior of EDA algorithms
  - non-expectable results
    - insufficiency
    - different means of measuring quality needed
- Poor quality of algorithms
  - insufficiency
    - attempts to remedy
Randomness of Algorithms
Randomness

The algorithm output depends on bias introduced by designer

- input perturbations [Puggelli, Welp, DAC'11]
- external bias, external randomness [Fišer, Schmidt, DDECS'14]

⇒ random behavior of deterministic algorithms
⇒ their robustness compromised
Example of (non-)robustness

Case study:

- **Processes tried:**
  - ABC, optimization & standard cells mapping ("dch; map")
  - ABC, optimization & 4-LUT mapping ("dch; if; mfs")
  - two commercial FPGA mapping tools
- 228 LGSynth benchmark circuits measured
- permuted inputs and outputs in the source file
- repeated 100,000-times (100,000 permutations)

**Results:**

<table>
<thead>
<tr>
<th></th>
<th>ABC Gate mapping</th>
<th>ABC LUT mapping</th>
<th>tool #1</th>
<th>tool #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. difference</td>
<td>14.49%</td>
<td>17.84%</td>
<td>0.22%</td>
<td>9.23%</td>
</tr>
<tr>
<td>Max. difference</td>
<td>86.47%</td>
<td>97.01%</td>
<td>17.26%</td>
<td>66.62%</td>
</tr>
</tbody>
</table>

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
Modeling the phenomenon

- Randomness from outside
  - independent random events
  - Gaussian distribution expected

experiment:
- `br2` circuit
- LUT mapping - ABC ('dch; if; mfs')
  - Gaussian distribution seems to be working

Practice:
- limited by optimum
  - truncated Gaussian distribution
Another behavior

- **cordic circuit, LUT mapping ('dch; if; mfs')**
- three maxima,
- three components
- **Gaussian mixtures**
  - EM algorithm applied

![Histogram graph](attachment:image.png)
Example behaviors

- *br2 circuit*, LUT mapping ('dch; if; mfs')
  - one maximum,
  - two components
- **Gaussian mixtures**
  - EM algorithm applied
How frequent are they?

Model classes: benchmarks counts (out of 264)

<table>
<thead>
<tr>
<th>Local max.</th>
<th>Components</th>
<th>Gate mapping</th>
<th>LUT mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invariant</td>
<td>Invariant</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>53</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>&gt;2</td>
<td>&gt;2</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Observations:
- sometimes unique solutions are produced. Mostly for "simple" circuits
- the most frequent case is a single local maximum and two Gaussian components
- cases with two or more local maxima are rare
Conclusions

- "Simple" processes are severely sensitive to randomness
  - ~15% quality difference on average
- "Simple" processes produce results forming "classes of quality"
- Gaussian mixture is a good distribution model

↓

Quality must be measured more precisely
Measuring Quality
Judging quality

- **Comparison of two algorithms (1)**

Which one is better?

<table>
<thead>
<tr>
<th>Bitstream length [-]</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>350</td>
<td>0</td>
</tr>
<tr>
<td>450</td>
<td>0</td>
</tr>
<tr>
<td>550</td>
<td>10</td>
</tr>
<tr>
<td>650</td>
<td>30</td>
</tr>
<tr>
<td>750</td>
<td>20</td>
</tr>
<tr>
<td>850</td>
<td>10</td>
</tr>
<tr>
<td>950</td>
<td>5</td>
</tr>
<tr>
<td>1050</td>
<td>5</td>
</tr>
<tr>
<td>1150</td>
<td>5</td>
</tr>
<tr>
<td>1250</td>
<td>5</td>
</tr>
<tr>
<td>1350</td>
<td>5</td>
</tr>
<tr>
<td>1450</td>
<td>5</td>
</tr>
<tr>
<td>1550</td>
<td>5</td>
</tr>
<tr>
<td>1650</td>
<td>5</td>
</tr>
<tr>
<td>1750</td>
<td>5</td>
</tr>
<tr>
<td>1850</td>
<td>5</td>
</tr>
<tr>
<td>1950</td>
<td>5</td>
</tr>
<tr>
<td>2050</td>
<td>5</td>
</tr>
</tbody>
</table>

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
Judging quality

• Comparison of two algorithms (2)

Which one is better?
Quantification of Quality and Robustness

- **Quality** of algorithms cannot be measured by a single value
  - average (mean) values from randomized measurements

  \[
  \text{relative quality} = \frac{\text{actual size}}{\text{best known size}}
  \]

- **Robustness** measure required
  - with of the histograms

  \[
  \text{relative difference} = \frac{\max(GM) - \min(GM)}{\max(GM)}
  \]
Study of ABC processes

- 264 benchmark circuits
- LUT mapping ('dch; if; mfs')
- Relative quality measured

Observations:
- maximum ca. 1.3
  (30% above known optimum)
- 69 cases above 2.0
- 14 cases above 10
  (10-times the known optimum)
Study of ABC processes

- 264 benchmark circuits
- LUT mapping ('dch; if; mfs')
- Relative difference measured

Observations:
- maximum ca. 0.13
  (13% variation)
- much higher differences are common
  (2 = 100%)
Conclusions

- Processes must be evaluated statistically
  - average quality values
  - robustness – span of the values
- "Simple" processes lack both quality and robustness

Both quality and robustness must be improved
First Attempt to Remedy

Iteration
ABC iteration

Idea: making the process "better"

→ ABC iteration

cordic
LUT mapping ('dch; if; mfs')
process iterated (repeated)

definitely better quality
Idea: making the process "better"

⇒ **ABC iteration**

*alu4*

SC mapping ('dch; map')

process iterated (repeated)

definitely better quality
Observation:

- iteration does improve rel. quality
How many iterations are needed?

Experiment:
- 490 circuits, academic benchmarks, industrial designs
- standard cells mapping ('dch; map')
- 5000 iterations
- After how many iterations the process converged:

<table>
<thead>
<tr>
<th>Iterations</th>
<th># of circuits</th>
<th># of circuits - cumulative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>100 (20%)</td>
</tr>
<tr>
<td>2–10</td>
<td>169</td>
<td>269 (55%)</td>
</tr>
<tr>
<td>11–20</td>
<td>30</td>
<td>299 (61%)</td>
</tr>
<tr>
<td>21–100</td>
<td>44</td>
<td>343 (70%)</td>
</tr>
<tr>
<td>101–500</td>
<td>22</td>
<td>365 (74%)</td>
</tr>
<tr>
<td>501–1000</td>
<td>15</td>
<td>380 (78%)</td>
</tr>
<tr>
<td>1001–2000</td>
<td>25</td>
<td>405 (83%)</td>
</tr>
<tr>
<td>2001–5000</td>
<td>85</td>
<td></td>
</tr>
</tbody>
</table>

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
How many iterations pay off?

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
The stopping problem

- **Most of circuits require iteration**
  - For majority of circuits 20 iterations are sufficient
    - → acceptable amount
  - But for some circuits, even 5000 is not enough
    - → does this effort pay off?

- **General question:** When to stop the iteration?
  - General answer:
    - Let's sacrifice 10% of run-time for 2% improvement.

- **General problem:** How to find out online?
The stopping problem

- What about behaviors like this:

- max512 benchmark circuit

- IWLS 2005 vga_lcd benchmark circuit
ABC iteration & robustness

Idea: improve quality of the process $\Rightarrow$ improve robustness (?)

- **Processes tried:**
  - ABC, optimization & standard cell mapping ('dch; map'), 20 iterations
  - ABC, optimization & 4-LUT mapping ('dch; if; mfs'), 20 iterations
  - 264 circuits, 100,000 permutations

**Results:**

<table>
<thead>
<tr>
<th></th>
<th>ABC gate mapping</th>
<th>ABC LUT mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 iteration:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. difference</td>
<td>14.49%</td>
<td>17.84%</td>
</tr>
<tr>
<td>Max. difference</td>
<td>86.47%</td>
<td>97.01%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ABC gate mapping</th>
<th>ABC LUT mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>20 iterations:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. difference</td>
<td>15.0%</td>
<td>20.01%</td>
</tr>
<tr>
<td>Max. difference</td>
<td>87.3%</td>
<td>85.71%</td>
</tr>
</tbody>
</table>
On marching camels

- *cordic*, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

1 iteration
On marching camels

- *cordic*, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
On marching camels

- \textit{cordic}, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

3 iterations

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
On marching camels

- **cordic**, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

4 iterations
On marching camels

- **cordic**, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
On marching camels

- *cordic*, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations
On marching camels

- *cordic*, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

7 iterations
On marching camels

- **cordic**, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014

8 iterations
On marching camels

- cordic, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations

9 iterations
On marching camels

- **cordic**, LUT mapping ('dch; if; mfs'), 1-10 iterations, 2 M permutations
On marching camels

- alu4 circuit
- gate mapping
- 1-100 iterations

"good" results with low certainty
avg = 2625  
σ = 24

bad results with high certainty
avg = 1905  
σ = 300

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
Quality vs. robustness

Observation:

- iteration trades quality for robustness
The alu4 problem

- alu4 circuit
- gate mapping
- 1-100 iterations

Why it is not truncated?

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
The alu4 problem

Recall:

- the convergence

Why it is not truncated?

alu4

SC mapping ('dch; map')

process iterated (repeated)

5000 iterations (!)
The alu4 problem

... and

Heavily suboptimal results

Best known result: 73 gates
Conclusions

- **ABC iteration helps**
  - increases quality

- **ABC iteration does not increase robustness**
  - the contrary is true

- **20 iterations are sufficient for majority of circuits**
  - but even 1000 is not enough for large and difficult circuits
    - difficult to determine the stopping condition
    - but the improvement is by units of per cent only

- There are circuits for which ABC iteration does not work well

- Practical observation (& theory):
  - When the random distribution becomes truncated, the process approaches global optimum

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
Some More "Nasty" Circuits

... and treatments for them
Properties:
- 23 inputs, 2 outputs
- two cones of logic
- bad multi-level description

ABC mapping results (1 iteration of 'dch; if; mfs'):
- ranging between 20 – 680 LUTs

Treatment 1:
- Iteration
- Result (after 20 iterations): 10 LUTs (probably the optimum)

Lessons learned:
- Bad structure can sometimes be efficiently handled by iteration
Properties:
- 23 inputs, 2 outputs
- two cones of logic
- bad multi-level description

ABC mapping results (1 iteration of 'dch; if; mfs'):
- ranging between 20 – 680 LUTs

Treatment 2:
- Destroying the structure (collapsing into SOP)
- Result (after 1 iteration): 13 LUTs

Lessons learned:
- Destroying bad structure be collapsing helps
Logic Examples with Known Upper Bound

- **Proposed in**
  

- **Originally for testing of LUT mappers**

- **Construction:**
  - the basis is an artificially constructed circuit $g_5$ with optimum of 7 4-LUTs
  - $g_{25}$ constructed as 10 interconnected $g_5 \Rightarrow 70$ LUTs

29 gates 18,905 product terms 1,167,054 gates

$g_{25} \xrightarrow{\text{collapse into SOP}} \xrightarrow{\text{SIS tech_decomp}} \xrightarrow{\text{LEKU-CD}}$
ABC LUT mapping ('dch; if; mfs')

- 18,877 LUTs
- 3,426 LUTs
- 70 LUTs (lower bound)
LEKU

Treatment for LEKU:

- Completely destroy structure
  - collapse into SOP
  - construct global BDD
- ABC mapping (5000 iterations)

⇒ 80 LUTs

Lessons learned:

- Artificially introduced bad structure (bad = too complex) makes standard synthesis insuperable problems
- Destroying the structure helps
Construction:

- 8 LUTs
- 11 LUTs
- 662 PLA terms

**alu1** → add parity tree → collapse into SOP → **alu1_p**

ABC mapping results:

- ~ 300 LUTs

Treatment:

- BDS – BDD-based decomposition, capable of XOR decomposition
- Result (after ABC mapping): 24 LUTs

Lessons learned:

- Artificially introduced bad structure (bad = destroyed) makes standard synthesis insuperable problems
- Proper decomposition helps
Properties:
- 16-input symmetric single-output function, XOR intensive, collapsed
  ⇒ PLA, 481 terms
- multi-level description – decomposed into simple gates

ABC mapping results (5000 iterations of 'dch; map'):
- 204 gates

Treatment 1:
- BDS – BDD-based decomposition, capable of XOR decomposition
- Result (after ABC mapping): 15 gates

Lessons learned:
- Artificially introduced bad structure (bad = destroyed) makes standard synthesis insuperable problems
- Proper decomposition helps
Properties:

- 16-input symmetric single-output function, XOR intensive, collapsed
  ⇒ PLA, 481 terms
- multi-level description – decomposed into simple gates

ABC mapping results (5000 iterations of 'dch; map'):

- 204 gates

Treatment 2:

- collapsing back to PLA
- Result (after ABC mapping): 15 gates

Lessons learned:

- Artificially introduced bad structure (bad = decomposed to simple gated) makes standard synthesis insuperable problems
- Destroying the structure helps
 Properties:
- 14 inputs, 8 outputs, multi-level description

ABC mapping results (5000 iterations of 'dch; map'):
- ~ 1500 gates

Treatment:
- Black magic
- Best result (collapsed, CGP, 5000 iterations of 'dch; map'): 73 gates

Lessons learned:
- There are benchmark circuits that:
  1. have bad multi-level description (collapsing helps)
  2. are hard to synthesize, no standard tool is able to discover good structure
  - genetic programming constructs the structure "randomly"
How does industry perform

- Two commercial LUT mapping tools tested
- Compared to:
  - extreme effort of ABC (5000 iterations of LUT mapping - 'dch; if; mfs')
  - "good process" – what worked well

<table>
<thead>
<tr>
<th>Circuit</th>
<th>&quot;Standard&quot; ABC process</th>
<th>Good process (or optimum)</th>
<th>Tool #1</th>
<th>Tool #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>cored</td>
<td>1 ABC iteration</td>
<td>20 ABC iterations (optimum)</td>
<td>10</td>
<td>35</td>
</tr>
<tr>
<td>LEKU-CD</td>
<td>5000 ABC iterations</td>
<td>??? (optimum)</td>
<td>70</td>
<td>failed</td>
</tr>
<tr>
<td>alu1_p</td>
<td>5000 ABC iterations</td>
<td>BDS + ABC</td>
<td>11</td>
<td>80</td>
</tr>
<tr>
<td>t481</td>
<td></td>
<td>For LUT mapping worked sufficiently</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alu4</td>
<td>5000 ABC iterations</td>
<td>collapse + 5000 ABC iterations + CGP</td>
<td>22</td>
<td>664</td>
</tr>
</tbody>
</table>

Petr Fišer, Jan Schmidt: The Logic Synthesis Homework Is Not Done Yet, GNLS 2014
Conclusions

- There are circuits for which "conventional" synthesis, both academic and commercial, seriously fails.
  - Reason for the failure: bad circuit structure
    - introduced intentionally to make the circuit more complex (LEKU),
    - or probably accidently (cordic, t481),
    - or intentionally, with good hopes (alu1_p)
    - alu4 - ???
  - Deadly transformations:
    - completely destroying the structure (collapsing, global BDDs)
    - decomposition into simple gates
  - There are different (and special) cures:
    - completely destroying the structure (collapsing)
    - special decomposition (XOR, MUX)
    - complete reconstruction of the circuit (CGP)
Major question

Do such circuits appear in practice?

- If so, the tools are not prepared for what designers write
- If not, why such circuits commonly appear in benchmark sets?
- Anyway, the tools should be able to cope with them, just for the case of unintentional designer's bad move
1. Because of serious influence of "external randomness", tools must be tested more precisely
   - statistical properties – average values, relative differences
   - immunity to randomness: robustness

2. Iteration helps. Known fact
   - study of how much it does
   - iteration trades quality for robustness
   - open question: when to stop?
   - truncated Gaussian distribution can be a hint of approaching global optimum
Final summary & topics for discussion (2)

3. **Insufficiency of tools**
   - there are benchmark circuits for which synthesis fails
   - there are special cures
     - … but who knows in advance, which one to use
   \[ \Rightarrow \text{bad structures should not be introduced} \]

4. **Benchmarking problem**
   - many benchmark sets, benchmark origins and transformations are not exactly known
   - are benchmark circuits real circuits intended for synthesis, or just artefacts made to complicate things?
   - crucial need for optimum results or recording the best known ones