Genetic Improvement and Approximation: From Hardware to Software

Lukáš Sekanina
Brno University of Technology, Faculty of Information Technology
Brno, Czech Republic
sekanina@fit.vutbr.cz

Genetic improvement and genetic approximation

- Error
- Acceptable error increase
- Genetic approximation
- Initial solution
- Genetic improvement

- Power
Motivation for Approximate computing

- Variability of circuit parameters for technology nodes < 45 nm is very HIGH
  - Low-power computing, but with unreliable components!
- High performance & low power computing is requested.
  - Many applications are error-resilient - the error can be traded for energy savings or performance.

Functional approximation by means of Genetic Improvement

Search for "approximate computing" in articles by Google Scholar (Jan, 2016)
Outline

- Genetic improvement of complex digital circuits
- Genetic approximation of complex digital circuits
- Genetic approximation of elementary SW functions for microcontrollers: Median
- Conclusions
HDL – Hardware Description Languages

alu4.pla
.alu
.ends

alu4.blif
.model /alu
.inputs i_0 i_1 i_2 i_3 i_4 i_5 i_6 i_7 i_8 i_9 i_10 i_11 i_12
.outputs o_0 o_1 o_2 o_3 o_4 o_5 o_6 o_7
.gate NAND A=i_2 B=_net203568 O=net196167
.gate NAND A=i_11 B=_net203428 O=net196385
.gate OR A=_net204803 B=_net200095 O=o_5
.gate NOR A=i_0 B=i_12 O=_net196891
.gate NAND A=_net203823 B=_net196167 O=net198561
.gate NAND A=i_1 B=_net198561 O=net198562

entity Acc is
port ( CLK: in std_logic;
RST: in std_logic;
SUM: out std_logic_vector(7 downto 0);
DONE: out std_logic);
end Acc;
architecture behav of Acc is
signal addr: std_logic_vector(2 downto 0);
signal val: std_logic_vector(7 downto 0);
signal add_out: std_logic_vector(7 downto 0);
signal reg_out: std_logic_vector(7 downto 0);
signal done: std_logic;
begin
begin
reg: process(CLK, RST)
begin
if RST = '1' then
reg_out <= X"00";
elif clk'event and clk = '1'
then
if stop = '0' then
reg_out <= add_out;
else
addr <= addr + 1;
stop <= '0';
end if;
end if;
end process;

end behav;
Digital circuit design with Cartesian GP [Miller 1999]

Example: CGP parameters
- \( n_r = 3 \) (#rows)
- \( n_c = 3 \) (#columns)
- \( n_i = 3 \) (#inputs)
- \( n_o = 2 \) (#outputs)
- \( n_a = 2 \) (max. arity)
- \( L = 3 \) (level-back parameter)
- \( \Gamma = \{\text{NAND}^0, \text{NOR}^1, \text{XOR}^2, \text{AND}^3, \text{OR}^4, \text{NOT}^5\} \)

Mutation-based \((1+\lambda)\) EA

Typical fitness function (circuit functionality):

\[
f = \sum_{i=1}^{K} |y_i - w_i|
\]

- Desired response
- Circuit response
- Number of test vectors

\( K = 2^{\text{inputs}} \) for combinational circuits.

Max: ~20 inputs
Max: ~ tens of gates
No scalable!!!
Functionality: Two types of specifications

- **Complete specifications**
  - A correct output value is requested for every possible input (e.g. for arithmetic circuits)
  - \(2^n\) test cases used to evaluate an \(n\)-input circuit
  - Impossible to improve the functionality of a correct solution, only non-functional parameters can be improved.

- **Incomplete specifications**
  - It is difficult to define correct output values for all possible inputs, e.g. filters, classifiers, predictors, ...
  - A circuit with an acceptable error is sought using a training set of \(k\) test cases, \(k \ll 2^n\)
  - GI can improve functional and non-functional parameters.
• SAT solver is used to decide whether candidate circuit $C_i$ and reference circuit $C_1$ are functionally equivalent.
  • If so, then $\text{fitness}(C_i) = \text{the number of gates in } C_i$;
  • Otherwise: discard $C_i$.

[Vašíček, Sekanina: Genetic Programming and Evolvable Machines 12(3), 2011]
Creating an auxiliary circuit $G$

If $C_1$ and $C_2$ are not functionally equivalent then there is at least one assignment to the inputs for which the output of $G$ is 1.
Tseitin transform to create CNF for circuit $G$

Example: $y = \text{not} \ (x)$

$\begin{array}{c|c|c}
x & y & g \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}$

$g = (\neg x \lor \neg y) \ (x \lor y)$

$(x_2 + x_8)(\overline{x_2} + \overline{x_8})$

$(x_8 + x_9)(x_3 + x_9)(\overline{x_8} + \overline{x_3} + x_9)$

$(x_1 + x_{10})(x_9 + x_{10})(\overline{x_1} + \overline{x_9} + \overline{x_{10}})$

$(\overline{x_7} + x_{10} + x_{11})(x_7 + \overline{x_{10}} + x_{11})(\overline{x_7} + x_{10} + x_{11})(x_7 + x_{10} + x_{11})$

$(\overline{x_6} + x_9 + x_{12})(x_6 + \overline{x_9} + x_{12})(\overline{x_6} + x_9 + x_{12})(x_6 + x_9 + x_{12})$

$(x_{11} + x_{12} + \overline{x_{13}})(x_{12} + x_{11})(x_{12} + \overline{x_{12}})$

$(x_{13})$
SAT solver in action

SAT solver: MiniSAT

variables: 13, clauses: 30, time elapsed: 0.03ms

result: SATISFIABLE / NONEQUIVALENT

model / counter example: 001111101011

\[(x_1 + x_4)(\overline{x_1} + \overline{x_4})\]

\[(x_2 + x_5)(\overline{x_2} + \overline{x_5})\]

\(x_6\)

\(\overline{x_7}\)
Experiment 1: Minimization of the number of gates

<table>
<thead>
<tr>
<th>circuit</th>
<th>PI</th>
<th>PO</th>
<th>SIS</th>
<th>ABC</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>CGP</th>
<th>impr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>apex1</td>
<td>45</td>
<td>45</td>
<td>1394</td>
<td>1862</td>
<td>1439</td>
<td>1272</td>
<td>1368</td>
<td>847</td>
<td>33.4%</td>
</tr>
<tr>
<td>apex2</td>
<td>39</td>
<td>3</td>
<td>151</td>
<td>225</td>
<td>221</td>
<td>195</td>
<td>299</td>
<td>90</td>
<td>40.4%</td>
</tr>
<tr>
<td>apex3</td>
<td>54</td>
<td>50</td>
<td>1405</td>
<td>1737</td>
<td>1494</td>
<td>1332</td>
<td>1515</td>
<td>1038</td>
<td>22.1%</td>
</tr>
<tr>
<td>apex5</td>
<td>117</td>
<td>88</td>
<td>751</td>
<td>768</td>
<td>728</td>
<td>609</td>
<td>921</td>
<td>613</td>
<td>-0.7%</td>
</tr>
<tr>
<td>cordic</td>
<td>23</td>
<td>2</td>
<td>67</td>
<td>61</td>
<td>67</td>
<td>49</td>
<td>90</td>
<td>32</td>
<td>34.7%</td>
</tr>
<tr>
<td>cps</td>
<td>24</td>
<td>109</td>
<td>1128</td>
<td>1109</td>
<td>1150</td>
<td>975</td>
<td>967</td>
<td>585</td>
<td>39.5%</td>
</tr>
<tr>
<td>duke2</td>
<td>22</td>
<td>29</td>
<td>406</td>
<td>356</td>
<td>417</td>
<td>366</td>
<td>357</td>
<td>260</td>
<td>27.0%</td>
</tr>
<tr>
<td>e64</td>
<td>65</td>
<td>65</td>
<td>192</td>
<td>384</td>
<td>183</td>
<td>191</td>
<td>255</td>
<td>129</td>
<td>29.5%</td>
</tr>
<tr>
<td>ex4p</td>
<td>128</td>
<td>28</td>
<td>488</td>
<td>523</td>
<td>468</td>
<td>467</td>
<td>555</td>
<td>349</td>
<td>25.3%</td>
</tr>
<tr>
<td>misex2</td>
<td>25</td>
<td>18</td>
<td>111</td>
<td>121</td>
<td>94</td>
<td>89</td>
<td>108</td>
<td>71</td>
<td>20.2%</td>
</tr>
<tr>
<td>vg2</td>
<td>25</td>
<td>8</td>
<td>95</td>
<td>113</td>
<td>88</td>
<td>83</td>
<td>109</td>
<td>78</td>
<td>6.0%</td>
</tr>
</tbody>
</table>

CGP + SAT solver:
ES(1+1), 1 mut/chrom, seed: SIS, Gate set: {AND, OR, NOT, NAND, NOR, XOR}
100 runs (12 hours each)
Average area improvement: 25%

ABC, SIS – conventional open academic synthesis tools – very fast (seconds, minutes)
C1, C2, C3 – commercial synthesis tools

[Vašíček, Sekanina: DATE 2011]
Experiment 1: Convergence curves

- More time $\Rightarrow$ better results in the case of CGP
- Current circuit synthesis and optimization tools provide far from optimum circuits!
Experiment 2: SAT solving combined with simulation

SAT solver is called only if the circuit simulation performed for a small subset of vectors has indicated no error in the candidate circuit.

100 combinational circuits (≥15 inputs) - IWLS2005, MCNC, QUIP benchmarks

Heavily optimized by ABC

1: alcom (N_G = 106 gates; N_PI = 15 inputs; N_PO = 38 outputs)

100: ac97ctrl (N_G = 16,158; N_PI = 2,176; N_PO = 2,136)
Experiment 2: SAT solving combined with simulation

CGP + SAT solver + circuit simulation

Y-axis: Gate reduction w.r.t. ABC after 15 minutes, 34% on average
▲ Gate reduction w.r.t. ABC after 24 hours

[Vašíček Z.: EuroGP 2015]
Genetic approximation

- Relaxed equivalence checking is needed for approximate computing
  - What is the distance between functionality of two circuits?
  - How to calculate this distance for complex circuits when a simulation using a data set is not accurate?

The Hamming distance can be obtained using Binary Decision Diagrams for (many useful) complex circuits in a short time!
Binary Decision Diagrams (BDD)

\[ f = ac + bc \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Truth table

Decision tree

Reduced Ordered BDD (ROBDD)

Operations over (RO)BDDs implemented by many libraries, e.g. Buddy.
Hamming distance using ROBDD

- Create ROBDD for the parent circuit $C_A$, the offspring circuit $C_B$ and the XOR gates.
- The error is the average Hamming distance

$$\text{Error}(A) = \frac{\sum_{i=1}^{n_o} \text{SatCount}(z_i)}{2^{n_i}}$$
Circuit approximation: Example

- Clmb (bus interface): 46 inputs, 33 outputs
- Original clmb: 641 gates, 19 logic levels, $|BDD| = 6966$, $|BDD_{opt}| = 627$ (SIFT in 2.3 s)
- Optimized by CGP (no error allowed):
  - Best: 410 gates, 12 logic levels -- in 29 minutes ($2.9 \times 10^6$ generations)
  - Median: 442 gates, 13 logic levels

Properly optimize before doing approximations!
Detailed error analysis for itc_b10 circuit

The median function

corrupted image (10% pixels, impulse noise)

filtered image (9-input median filter)
Median as a comparator network

```c
#define PIX_SORT(a,b) {
    if ((a)>(b))
        PIX_SWAP((a),(b));
}
pixelvalue opt_med9 (pixelvalue * p)
{
    PIX_SORT(p[1], p[2]) ; PIX_SORT(p[4], p[5]) ; PIX_SORT(p[7], p[8]) ;
    PIX_SORT(p[0], p[1]) ; PIX_SORT(p[3], p[4]) ; PIX_SORT(p[6], p[7]) ;
    PIX_SORT(p[1], p[2]) ; PIX_SORT(p[4], p[5]) ; PIX_SORT(p[7], p[8]) ;
    PIX_SORT(p[0], p[3]) ; PIX_SORT(p[5], p[8]) ; PIX_SORT(p[4], p[7]) ;
    PIX_SORT(p[3], p[6]) ; PIX_SORT(p[1], p[4]) ; PIX_SORT(p[2], p[5]) ;
    PIX_SORT(p[4], p[7]) ; PIX_SORT(p[4], p[2]) ; PIX_SORT(p[6], p[4]) ;
    PIX_SORT(p[4], p[2]) ; return(p[4]) ;
}
```

Approximations conducted by means of CGP (and training images):
Approximate 9-median as SW for microcontrollers

<table>
<thead>
<tr>
<th>Impl.</th>
<th>Time [μs]</th>
<th>Energy [nWs]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STM32</td>
<td>PIC24</td>
</tr>
<tr>
<td>6-ops</td>
<td>2.8</td>
<td>54.5</td>
</tr>
<tr>
<td>10-ops</td>
<td>3.3</td>
<td>70.8</td>
</tr>
<tr>
<td>14-ops</td>
<td>3.9</td>
<td>86.8</td>
</tr>
<tr>
<td>18-ops</td>
<td>4.5</td>
<td>104.5</td>
</tr>
<tr>
<td>22-ops</td>
<td>5.0</td>
<td>116.7</td>
</tr>
<tr>
<td>26-ops</td>
<td>5.9</td>
<td>130.0</td>
</tr>
<tr>
<td>30-ops</td>
<td>6.0</td>
<td>142.0</td>
</tr>
<tr>
<td>34-ops</td>
<td>6.4</td>
<td>154.0</td>
</tr>
<tr>
<td>38-ops</td>
<td>6.9</td>
<td>165.5</td>
</tr>
<tr>
<td>qsort</td>
<td>28.5</td>
<td>1106.2</td>
</tr>
</tbody>
</table>

34.9% error prob., max. error dist. 2
52% power reduction

4.8% error prob., max. error dist. 1
21% power reduction

fully-working median

ops = operations in the source code.

```c
#define PIX_SORT(a,b) {
    if ((a)>(b)) {
        PIX_SWAP((a),(b));
    }
}
```

Conclusions

- Genetic improvement and genetic approximation introduced in the context of circuits described as netlists.

- Complete and incomplete specifications considered.

- The notion of relaxed equivalence checking was introduced.

- Future work
  - Efficient methods of relaxed equivalence checking
    - SAT-based, BDD-based, pseudo-Boolean polynomial representation-based etc.
  - Efficient search methods exploiting properties of a particular relaxed equivalence checking method
  - Real-world case studies
Acknowledgement

• EHW group at Brno University of Technology
  • Zdeněk Vašíček, Michal Bidlo, Roland Dobai
  • Michaela Šikulová, Radek Hrbáček, Vojtěch Mrázek, David Grochol, and other students

• Research projects
  • IT4Innovations Centre of Excellence – National supercomputing center
  • Advanced Methods for Evolutionary Design of Complex Digital Circuits, 2014 – 2016 (Czech Science Foundation)
  • Relaxed equivalence checking for approximate computing, 2016 – 2018 (Czech Science Foundation)
Thank you for your attention!

Lukáš Sekanina

Brno University of Technology, Faculty of Information Technology
Brno, Czech Republic
sekanina@fit.vutbr.cz