Smart Bit-width Allocation for Low Power Optimization in a SystemC based ASIC Design Environment

by

Arindam Mallik, Debjit Sinha, Prith Banerjee, Hai Zhou

presented by

Prof. Robert Dick

Department of Electrical Engineering & Computer Science Northwestern University
Introduction

- Power consumption – A primary design criteria
- Levels of abstraction
  - Majority of EDA tools aim at Gate and RTL level
  - Largest gains are available at the behavioral level
- Few tools exist for design time analysis of precision in floating to fixed point translation
Outline

- Introduction
- Problem Definition
- Quantization Algorithm
- Experimental Results
- Conclusion & Future Work
Motivation

- Reduce bit-width precision of computation units such as adders and multipliers.
- Advantage – Immediate reduction in area and power
- Disadvantage – Round-off error increases
- We investigated system level tradeoffs of round-off errors with power consumption in the hardware implementation – an algorithm to get an optimized solution
Bitlength assignment

\[ a = \text{Input}_a \]
\[ b = \text{Input}_b \]
\[ c = a + b \]
\[ d = a * b \]
\[ e = c + d \]

\[ \text{Bit}_a = \text{Bit}_{\text{Input}_a} \]
\[ \text{Bit}_b = \text{Bit}_{\text{Input}_b} \]
\[ \text{Bit}_c \geq \max(\text{Bit}_a, \text{Bit}_b) + 1 \]
\[ \text{Bit}_d \geq \text{Bit}_a + \text{Bit}_b \]

\[ \text{Bit}_e \geq \max(\text{Bit}_c, \text{Bit}_d) + 1 = \text{Bit}_a + \text{Bit}_b + 1 \]
Fixed Point Synthesis Problem

- Contains fixed point data types to model hardware accurately.
- The Behavioral Synthesis tool (CoCentric Compiler) – unable to synthesize fixed point
- **Possible solution!**
  Use data scaling
  - use of sc_int<> data type
  - more bits will capture more precision
  - SCALE_FACTOR – powers of two
Schematic of the Algorithm

High level description in SystemC with floating point

Data profiling & Inequality formation

Solution space recognition

SystemC simulation with minimum bits allocated to each variable

Scaling input parameters by the SCALE_FACTOR

Optimal point search algorithm

E < Given error constraint

YES

NO

Power/Area optimized ASIC design satisfying error constraint
Data Profiling

- Proposed for ASICs – we have a prior knowledge about the type of input data

- Used a small percentage of actual inputs
  - Finds the optimal solution based on Training Sets – a small percentage
  - After development of optimal hardware, we use a larger set to verify
  - Use of safety factor
Inequality Formation Example

while (1) {
    output_data_ready.write(false);
    wait_until(input_valid.delayed() == true);
    sample_tmp = sample.read();  \(\leadsto sample_{tmp}^{\text{BIT}} \geq sample^{\text{BIT}}\)
    acc = 0;
    acc = sample_tmp * coefs[0];
    for (int I = NUMTAPS; I > 0; i--) {
        //synopsys unroll
        pro = shift[i-1] * coefs[i];  \(\leadsto pro_{\text{BIT}} \geq shift_{\text{BIT}} + coeff_{\text{BIT}}\)
        acc = acc + shift[i-1] * coefs[i];  \(\leadsto acc_{\text{BIT}} \geq pro_{\text{BIT}} + \log_2(\text{NUMTAPS})\)
    }

    for (int I = NUMTAPS - 1; I >= 0; i--) {
        //synopsys unroll
        shift[i+1] = shift[i];
    }
    shift[0] = sample_tmp;  \(\leadsto shift_{\text{BIT}} \geq sample_{\text{BIT}}\)
    // write output values
    result.write(acc);  \(\leadsto result_{\text{BIT}} \geq acc_{\text{BIT}}\)
    output_data_ready.write(true);
    wait();
};
Solution Space Recognition

- Using Range Propagation on inputs
  - $sample_{BIT} \geq 7$
  - $coeff_{BIT} \geq 8$

- $acc$ and $result$ have the largest bit-length
  - $32 \geq result_{BIT}$
  - $32 \geq acc_{BIT}$

- Solving the set of inequalities gives us
  - $19 \geq sample_{BIT} \geq 7$
  - $20 \geq coeff_{BIT} \geq 8$
Different Search Procedure

- **Greedy Search Algorithm**
  - Heuristic solution based on greedy strategy
  - Consists of two steps
    - Sub-optimal point search
    - Local search

- **Smart Search Algorithm**
  - Optimal algorithm proved for designs with 2 inputs (independent variables)
  - Heuristic approach for more than 2 inputs (scope of future work)
Problems in Greedy Search

- For each bit modification we need two simulations.
- Heuristic – Not guaranteed to be optimal.
- Exhaustive search for this problem has exponential time complexity.
Smart Search Algorithm

- For designs with 2-dimensions we proved it is optimal and always finds the solution in linear time.

- Lowering of the precision may be done using Binary Search (order \( \log n \)) – in worst case this degrades performance.

- Proposed heuristic for designs with more than two independent variables.
Example of Search Algorithms

<table>
<thead>
<tr>
<th>Bitlength (SCALE_FACTOR)</th>
<th>8(1)</th>
<th>10(4)</th>
<th>12(16)</th>
<th>14(64)</th>
<th>16(256)</th>
<th>18(1024)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_A</td>
<td>5.16</td>
<td>2.19</td>
<td>2.17</td>
<td>1.72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input_A</td>
<td>3.9</td>
<td>0.603</td>
<td>0.584</td>
<td>0.376</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input_A</td>
<td>0.486</td>
<td>0.399</td>
<td>0.222</td>
<td>0.182</td>
<td>0.164</td>
<td></td>
</tr>
<tr>
<td>Input_A</td>
<td>0.397</td>
<td>0.367</td>
<td>0.151</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input_A</td>
<td>0.388</td>
<td>0.347</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Optimal Point**
- Input_A (12, S_F 32)
- Input_B (14, S_F 64)

**E = 0.19 (<0.2%)**

**Suboptimal Pt**

**Greedy Search Algorithm**

**Smart Search Algorithm**
Simulation Results

<table>
<thead>
<tr>
<th>Error Constraint</th>
<th>Bit-width of the Independent Variables</th>
<th>Power Consumed (10^4 nW)</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>FIR16</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>input = 7; coeff = 8</td>
<td>91.1</td>
<td>12454</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>input = 9; coeff = 9</td>
<td>101</td>
<td>13243</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>input = 12; coeff = 14</td>
<td>376</td>
<td>19844</td>
</tr>
<tr>
<td></td>
<td><strong>INTFIR</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>input = 7; coeff = 13</td>
<td>450</td>
<td>104209</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>input = 10; coeff = 13</td>
<td>565</td>
<td>138909</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>input = 13; coeff = 14</td>
<td>1070</td>
<td>157775</td>
</tr>
<tr>
<td></td>
<td><strong>DECFIR</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>input = 7; coeff = 13</td>
<td>6790</td>
<td>23690</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>input = 9; coeff = 13</td>
<td>8980</td>
<td>32959</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>input = 11; coeff = 13</td>
<td>16100</td>
<td>42674</td>
</tr>
<tr>
<td></td>
<td><strong>LMS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>input = 10</td>
<td>30900</td>
<td>132984</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>input = 13</td>
<td>40700</td>
<td>175692</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>input = 14</td>
<td>45400</td>
<td>197479</td>
</tr>
</tbody>
</table>

Greedy Search Algorithm gave the optimal solution in 11 out of 12 cases but with extra number of simulations.
## Comparison for Simulations

<table>
<thead>
<tr>
<th>Error Constraint</th>
<th>Number of Simulations Required to Reach the Optimal Point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Greedy Search</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td><strong>FIR16</strong></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>4</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>7</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>17</td>
</tr>
<tr>
<td><strong>INTFIR</strong></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>1</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>5</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>23</td>
</tr>
<tr>
<td><strong>DECFIR</strong></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>1</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>3</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>5</td>
</tr>
<tr>
<td><strong>LMS</strong></td>
<td></td>
</tr>
<tr>
<td>E &lt;= 5%</td>
<td>4</td>
</tr>
<tr>
<td>E &lt;= 1%</td>
<td>5</td>
</tr>
<tr>
<td>E &lt;= 0.5 %</td>
<td>6</td>
</tr>
</tbody>
</table>
Conclusion and Future Work

- Presented a power optimization algorithm for with Quantization Error Constraints
  - Targeted at ASIC solutions
  - Different search algorithms to find the optimal configuration

- Future Work –
  - Full automation
  - Improved search algorithm