

AUTOTUNING HLS FOR FPGAS USING OPENTUNER AND LEGUP

Pedro Bruel (phrb@ime.usp.br)

Alfredo Goldman (gold@ime.usp.br)

Sai Rahul Chalamalasetti (sairahul.chalamalasetti@hpe.com)

Dejan Milojicic (dejan.milojicic@hpe.com)

ReConFig, December 5, 2017



*Institute of Mathematics and Statistics
University of São Paulo*



1. FPGAs, HLS & Autotuning
2. Background
3. Experiments & Results
4. Conclusion



The slides and all source code are hosted at [GitHub](#):

- [Code & Data](#): `github.com/phrb/legup-tuner`
- [Slides](#): `github.com/phrb/slides-reconfig-2017-autotuning`

FPGAs:

- Logic Blocks and Interconnections
- Reconfigurable

Tradeoff:

- Energy Efficiency and Performance
- Programmability



Using FPGAs in Bing:

1,632 Servers with FPGAs Running Bing Page Ranking Service (~30,000 lines of C++)

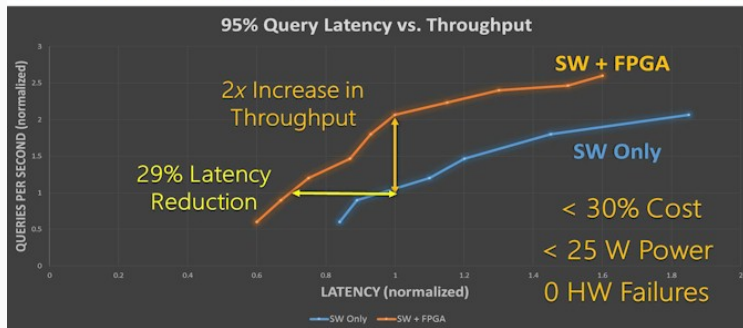


Image: enterprisetech.com/2014/09/03/microsoft-using-fpgas-speed-bing-search/ [Accessed in 27/11/17]

FPGAs: HIGH-LEVEL SYNTHESIS

HLS can generate **lower-latency applications**:

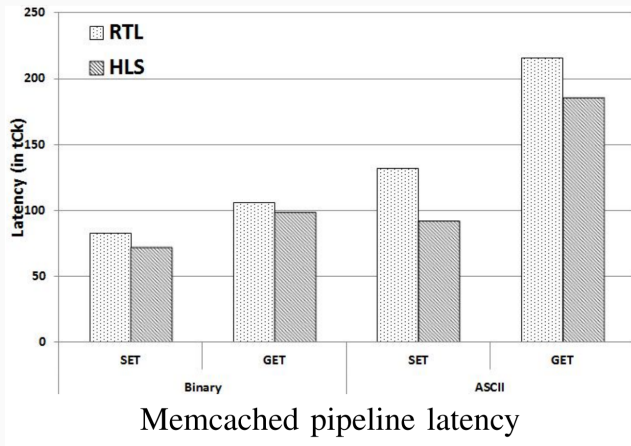


Image: Karras, Kimon, Michaela Blott, and Kees Visser. "High-Level Synthesis Case Study: Implementation of a Memcached Server." arXiv preprint arXiv:1408.5387 (2014).

FPGAs: HIGH-LEVEL SYNTHESIS

Qualitatively, with less effort:

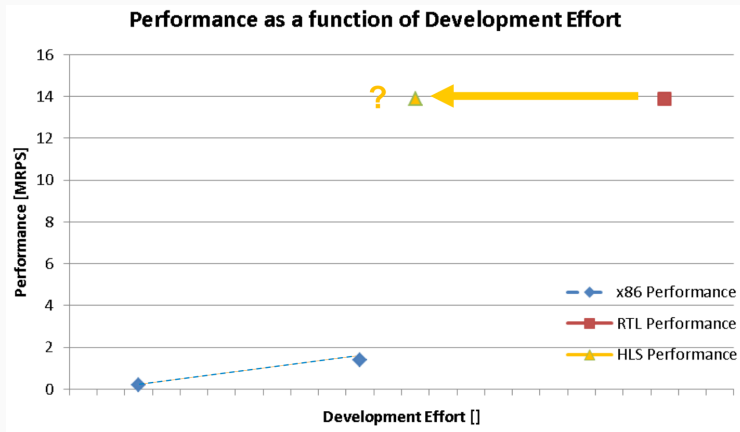


Image: Blott, Michaela, et al. "Achieving 10Gbps line-rate key-value stores with FPGAs." Presented as part of the 5th USENIX Workshop on Hot Topics in Cloud Computing. 2013.

This is an **old issue**:

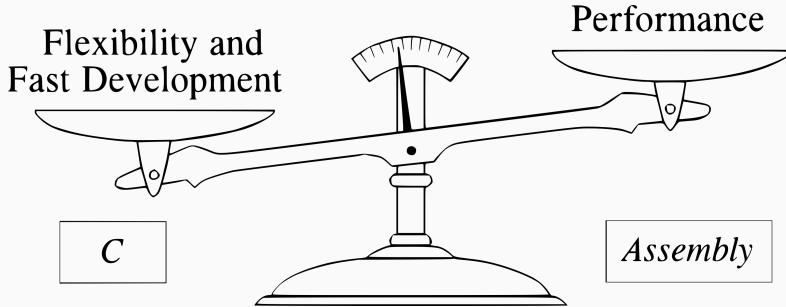


Image: Smith, Steven W. "The scientist and engineer's guide to digital signal processing." 1997

Schkufza *et al.*, "Stochastic program optimization." Communications of the ACM 59.2 (2016):

- Optimizing assembly instructions
- Using stochastic methods to explore the design space

FPGAs: HIGH-LEVEL SYNTHESIS

LegUp HLS flow:

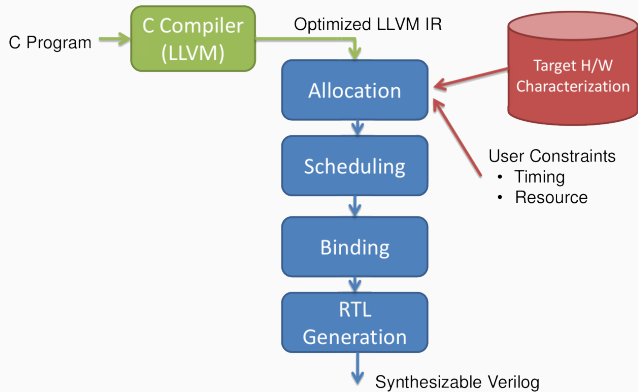


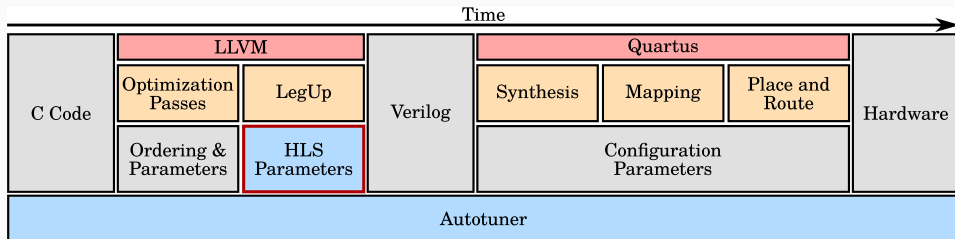
Image: Canis, Andrew Christopher. LegUp: Open-Source High-Level Synthesis Research Framework. Diss. University of Toronto, 2015.

Why use autotuning for HLS?

- HLS aims to ease FPGA programming for software engineers
- But configuring HLS requires FPGA programming knowledge
- Autotuning can help software engineers use HLS

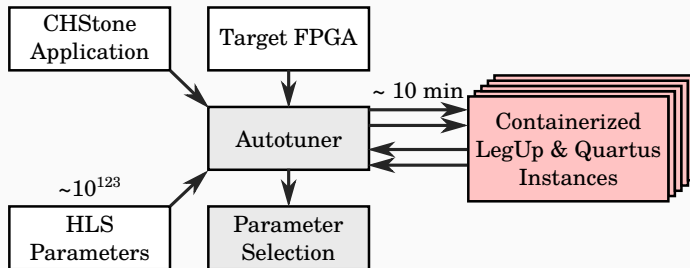
FPGAs: AUTOTUNING FOR HIGH-LEVEL SYNTHESIS

Our HLS **compilation flow**:



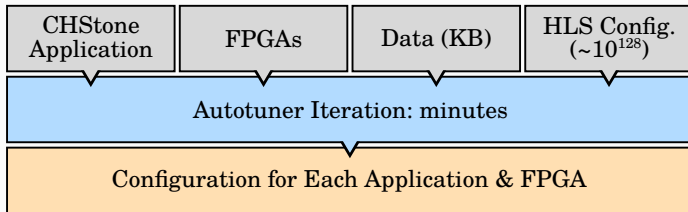
FPGAs: AUTOTUNING FOR HIGH-LEVEL SYNTHESIS

Our **autotuner setup**:



FPGAs: AUTOTUNING FOR HIGH-LEVEL SYNTHESIS

Our HLS **autotuning workflow**:



Huang, Qijing, *et al.*, "The effect of compiler optimizations on high-level synthesis-generated hardware." *ACM Transactions on Reconfigurable Technology and Systems* (2015):

- Optimizes parameters & ordering of a subset of LLVM passes
- Uses LegUp as an HLS tool
- Targets the Cyclone II device family
- Uses the CHStone benchmark
- Performs Exhaustive Search

Xu et al., "A Parallel Bandit-Based Approach for Autotuning FPGA Compilation." FPGA (2017):

- Does not optimize HLS
- Optimizes parameters of the Verilog-to-Routing (VTR) FPGA compilation tool
- Uses distributed OpenTuner instances

C-based High-Level Synthesis [benchmark](#) (CHStone):

Table 1: Autotuned CHStone Applications

Application	Short Description
blowfish	Symmetric-key block cypher
aes	Advanced Encryption Algorithm (AES)
adpcm	Adaptive Differential Pulse Code Modulation dec. and enc.
sha	Secure Hash Algorithm (SHA)
motion	Motion vector decoding from MPEG-2
mips	Simplified MIPS processor
gsm	Predictive coding analysis of systems for mobile comms.
dfsin	Sine function for double-precision floating-point numbers
dfmul	Double-precision floating-point multiplication
dfdiv	Double-precision floating-point division
dfadd	Double-precision floating-point addition

Metric composition problem:

- 8 hardware metrics obtained from Quartus
- But the autotuner optimizes a single value

Our solution so far:

- Normalize metrics using initial values
- Optimize the normalized sum of the 8 metrics

Normalized sum of metrics:

$$f(M, W) = \frac{\sum_{\substack{m_i \in M \\ w_i \in W}} w_i \left(\frac{m_i}{m_i^0} \right)}{\sum_{w_i \in W} w_i}$$

Where M is the set of hardware metrics and W is the set of weights

An **Optimization Scenario** consists of:

- An **optimization objective**: performance, area, . . .
- **Weights** for hardware metrics

Our scenarios:

- 3 **specific** scenarios & 1 **balanced** scenario
- Weights: **powers of two** from 1: **irrelevant** to 8: **high**

WEIGHTED OPTIMIZATION SCENARIOS

Table 2: **Weights** for **Optimization Scenarios** (**High** = 8, **Medium** = 4, **Low** = 2)

Metric	Area	Perf. & Lat	Performance	Balanced
<i>LUT</i>	High	Low	Low	Medium
<i>Registers</i>	High	High	Medium	Medium
<i>BRAMs</i>	High	Low	Low	Medium
<i>DSPs</i>	High	Low	Low	Medium
<i>FMax</i>	Low	High	High	Medium
<i>Cycles</i>	Low	High	Low	Medium

Experiments

- We performed 10 autotuning runs for each scenario
- Each run took 1.5 hours
- We kept track of how each hardware metric changed over time
- We used the Weighted Normalized Sum (WNS) as autotuning objective

Presentation of results for each scenario:

- Mean of the relative changes, over 10 tuning runs
- For each hardware metric, and for WNS
- For each CHStone application
- Bluer is always better than Redder

RESULTS: COMPARING DEFAULT & RANDOM STARTS

Comparing **Default** & **Random** starts:

- A **Default** start already provides a **sensible optimization**
- A **Random** start might favor the **exploration of new, unintuitive solutions**
- In our case, the **Default** start was provided to the **StratixV FPGA** by **LegUp**

RESULTS: COMPARING DEFAULT & RANDOM STARTS

<i>aes</i>	0.79	0.91	1.00	0.56	1.00	0.47	1.00	1.12
<i>adpcm</i>	0.68	1.13	1.00	0.54	1.00	0.56	0.60	0.98
<i>sha</i>	1.00	1.03	1.00	0.82	1.00	0.55	1.00	0.89
<i>motion</i>	1.02	1.00	0.60	0.85	1.00	0.57	1.00	0.94
<i>mips</i>	1.00	0.93	1.00	0.44	1.00	0.45	0.98	1.24
<i>gsm</i>	0.83	1.17	1.00	0.48	1.00	0.56	0.52	0.99
<i>dfsini</i>	0.79	0.97	1.00	0.61	1.00	0.60	1.49	1.53
<i>dfmul</i>	1.00	1.06	0.90	0.47	0.90	0.47	1.31	1.10
<i>dfdiv</i>	0.83	1.07	0.80	0.73	0.80	0.65	1.32	1.49
<i>dfadd</i>	1.00	0.94	1.00	0.82	1.00	0.71	1.00	0.93
<i>blowfish</i>	—	—	—	—	—	—	—	—
	<i>LUTs</i>	<i>Pins</i>	<i>BRAM</i>	<i>Regs</i>	<i>Blocks</i>	<i>Cycles</i>	<i>DSP</i>	<i>FMax</i>

Figure 1: Comparison of the absolute values for Random and Default starting points in the Balanced scenario

RESULTS: COMPARING DEFAULT & RANDOM STARTS

Comparing **Default** & **Random** starts:

- The **Default** start achieves **better absolute values** than the **Random** start, on most cases
- The next results were obtained using the **Default** start

RESULTS: THE BALANCED SCENARIO

Metric	<i>Balanced</i>
<i>LUT</i>	Medium
<i>itRegisters</i>	Medium
<i>BRAMs</i>	Medium
<i>DSPs</i>	Medium
<i>FMax</i>	Medium
<i>Cycles</i>	Medium

<i>aes</i>	0.94	0.90	1.00	1.00	0.97	1.00	0.98	1.00	0.76
<i>adpcm</i>	0.84	0.73	1.13	1.00	0.91	1.00	1.05	0.63	0.49
<i>sha</i>	0.98	1.00	1.03	1.00	0.96	1.00	0.86	1.00	0.97
<i>motion</i>	0.98	0.97	1.00	1.00	0.99	1.00	0.95	1.00	0.95
<i>mips</i>	0.95	1.00	1.07	1.00	0.90	1.00	1.01	0.80	0.89
<i>gsm</i>	0.95	0.95	1.17	1.00	0.99	1.00	0.95	0.49	1.08
<i>dfsint</i>	0.93	1.03	1.03	1.00	1.05	1.00	1.07	0.65	0.73
<i>dfmul</i>	0.85	1.00	1.13	0.90	0.88	0.90	1.06	0.31	0.76
<i>dfdiv</i>	0.84	1.00	1.07	0.80	1.15	0.80	1.34	0.41	0.57
<i>dfadd</i>	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97
<i>blowfish</i>	0.99	1.00	1.00	1.00	0.98	1.00	0.98	1.00	0.99
	WNS	<i>LUTs</i>	<i>Pins</i>	<i>BRAM</i>	<i>Regs</i>	<i>Blocks</i>	<i>Cycles</i>	<i>DSP</i>	<i>FMax</i>

Figure 2: Relative improvement for all metrics in the *Balanced* scenario

RESULTS: THE AREA SCENARIO

Metric	Area
LUT	High
Registers	High
BRAMs	High
DSPs	High
FMax	Low
Cycles	Low

<i>aes</i>	0.96	0.92	1.00	1.00	0.93	1.00	0.97	1.00	0.86
<i>adpcm</i>	0.83	0.78	1.11	1.00	0.96	1.00	1.04	0.63	0.52
<i>sha</i>	0.96	1.00	1.06	1.00	0.84	1.00	0.83	1.00	1.08
<i>motion</i>	0.97	0.94	1.00	1.00	0.97	1.00	0.92	1.00	0.95
<i>mips</i>	0.91	1.00	1.06	1.00	0.89	1.00	1.00	0.72	0.84
<i>gsm</i>	0.89	0.83	1.06	1.00	0.86	1.00	0.89	0.82	1.14
<i>dfsint</i>	0.94	1.00	1.06	1.00	1.00	1.00	1.02	0.76	0.80
<i>dfmul</i>	0.82	1.00	1.06	1.00	0.90	1.00	1.21	0.27	0.86
<i>dfdiv</i>	0.77	1.00	1.22	0.67	1.03	0.67	1.62	0.28	0.77
<i>dfadd</i>	0.99	1.00	1.11	1.00	0.94	1.00	1.00	1.00	1.04
<i>blowfish</i>	0.99	1.00	1.00	1.00	0.97	1.00	0.91	1.00	1.01
	WNS	LUTs	Pins	BRAM	Regs	Blocks	Cycles	DSP	FMax

Figure 3: Relative improvement for all metrics in the *Area* scenario

RESULTS: THE PERFORMANCE SCENARIO

Metric	Performance
<i>LUT</i>	Low
<i>Registers</i>	Medium
<i>BRAMs</i>	Low
<i>DSPs</i>	Low
<i>FMax</i>	High
<i>Cycles</i>	Low

<i>aes</i>	0.82	0.75	1.00	1.00	0.92	1.00	1.05	1.00	0.63
<i>adpcm</i>	0.84	0.94	1.17	1.00	0.96	1.00	0.94	0.69	0.74
<i>sha</i>	0.92	1.00	1.06	1.00	0.92	1.00	0.88	1.00	0.96
<i>motion</i>	0.99	1.00	1.00	1.00	1.01	1.00	1.00	1.00	0.98
<i>mips</i>	0.90	1.00	1.06	1.00	0.93	1.00	1.03	0.83	0.80
<i>gsm</i>	0.95	0.92	1.00	1.00	0.95	1.00	0.90	1.00	1.02
<i>dfsin</i>	0.87	1.11	1.06	1.00	1.27	1.00	1.25	0.53	0.56
<i>dfmul</i>	0.77	1.00	1.17	0.83	0.85	0.83	1.04	0.21	0.70
<i>dfdiv</i>	0.81	1.00	1.06	1.00	1.03	1.00	1.25	0.50	0.59
<i>dfadd</i>	0.97	1.00	1.00	1.00	0.97	1.00	1.00	1.00	0.94
<i>blowfish</i>	0.94	1.00	1.00	1.00	0.98	1.00	0.91	1.00	0.95
	WNS	LUTs	Pins	BRAM	Regs	Blocks	Cycles	DSP	FMax

Figure 4: Relative improvement for all metrics in the *Performance* scenario

RESULTS: THE PERFORMANCE & LATENCY SCENARIO

Metric	Perf. & Lat
LUT	Low
Registers	High
BRAMs	Low
DSPs	Low
FMax	High
Cycles	High

<i>aes</i>	0.83	0.83	1.00	1.00	0.88	1.00	0.89	1.00	0.73
<i>adpcm</i>	0.76	0.78	1.11	1.00	0.95	1.00	0.98	0.62	0.47
<i>sha</i>	0.88	1.00	1.06	1.00	0.85	1.00	0.77	1.00	1.00
<i>motion</i>	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.95
<i>mips</i>	0.95	1.00	1.11	1.00	0.91	1.00	0.99	0.89	0.96
<i>gsm</i>	0.92	0.92	1.11	1.00	0.87	1.00	0.90	0.67	1.11
<i>dfsint</i>	0.97	1.00	1.17	1.00	0.99	1.00	0.99	0.61	1.00
<i>dfmul</i>	0.86	1.00	1.22	1.00	0.89	1.00	1.01	0.33	0.84
<i>dfdiv</i>	0.82	1.00	1.11	1.00	0.84	1.00	0.96	0.34	0.83
<i>dfadd</i>	0.98	1.00	1.17	1.00	0.94	1.00	0.98	1.00	1.01
<i>blowfish</i>	0.96	1.00	1.00	1.00	0.97	1.00	0.93	1.00	0.97
	WNS	LUTs	Pins	BRAM	Regs	Blocks	Cycles	DSP	FMax

Figure 5: Relative improvement for all metrics in the *Performance & Latency* scenario

RESULTS: COMPARING THE WEIGHTED NORMALIZED SUM (WNS)

<i>aes</i>	0.94	0.96	0.82	0.83
<i>adpcm</i>	0.84	0.83	0.84	0.76
<i>sha</i>	0.98	0.96	0.92	0.88
<i>motion</i>	0.98	0.97	0.99	0.98
<i>mips</i>	0.95	0.91	0.90	0.95
<i>gsm</i>	0.95	0.89	0.95	0.92
<i>dfsint</i>	0.93	0.94	0.87	0.97
<i>dfmul</i>	0.85	0.82	0.77	0.86
<i>dfdiv</i>	0.84	0.77	0.81	0.82
<i>dfadd</i>	1.00	0.99	0.97	0.98
<i>blowfish</i>	0.99	0.99	0.94	0.96
Average	0.93	0.91	0.89	0.90
	<i>Balanced</i>	<i>Area</i>	<i>Performance</i>	<i>Perf. & Lat.</i>

Figure 6: Relative improvement for WNS in all scenarios

In all scenarios:

- Heavily weighted metrics **improved the most**
- Lightly weighted or irrelevant metrics **did not worsen greatly**

Autotuning starting point:

- A random start **might achieve more relative improvement**
- But a sensible start **achieves better absolute values**

Issues with the **Weighted Normalized Sum**:

- The WNS is a **very naive metric combination strategy**
- The **multi-objective optimization** domain provides **better approaches**

FUTURE WORK

Bigger designs:

- CHStone designs are very simple
- We will use our autotuning approach for bigger FPGA designs

Compiler configuration:

- HLS usually includes a C/C++ compiler step
- We will include compiler parameters or passes in our autotuner

Other HLS tools:

- Other HLS tools provide different search spaces
- We will use our autotuning approach for more proprietary and open HLS tools
- We want to use cost models that predict hardware metrics before synthesis

Autotuning HLS for FPGAs:

- Designers can **select optimization objectives using weights**
- Good **WNS improvements** after **1.5 hours** of autotuning
- Improvements for targeted metrics **without worsening other metrics**
- A **sensible starting point** impacts autotuning results positively

AUTOTUNING HLS FOR FPGAS USING OPENTUNER AND LEGUP

Pedro Bruel (phrb@ime.usp.br)

Alfredo Goldman (gold@ime.usp.br)

Sai Rahul Chalamalasetti (sairahul.chalamalasetti@hpe.com)

Dejan Milojicic (dejan.milojicic@hpe.com)

ReConFig, December 5, 2017



*Institute of Mathematics and Statistics
University of São Paulo*

