The Hong Kong Polytechnic University
香港理工大學

FPL 16

Department of Electronic and Information Engineering
Opening Minds • Shaping the Future • 啟迪思維 • 成就未來

Efficient and Reliable High-Level Synthesis Design Space Explorer for FPGAs

Dong Liu[1], Benjamin Carrion Schafer[2]

Department of Electronic and Information Engineering

The Hong Kong Polytechnic University

adam.d.liu@connect.polyu.hk[1], b.carrionschafer@polyu.edu.hk[2],

Design Automation and Reconfigurable Computing LAB
DARC

Outline

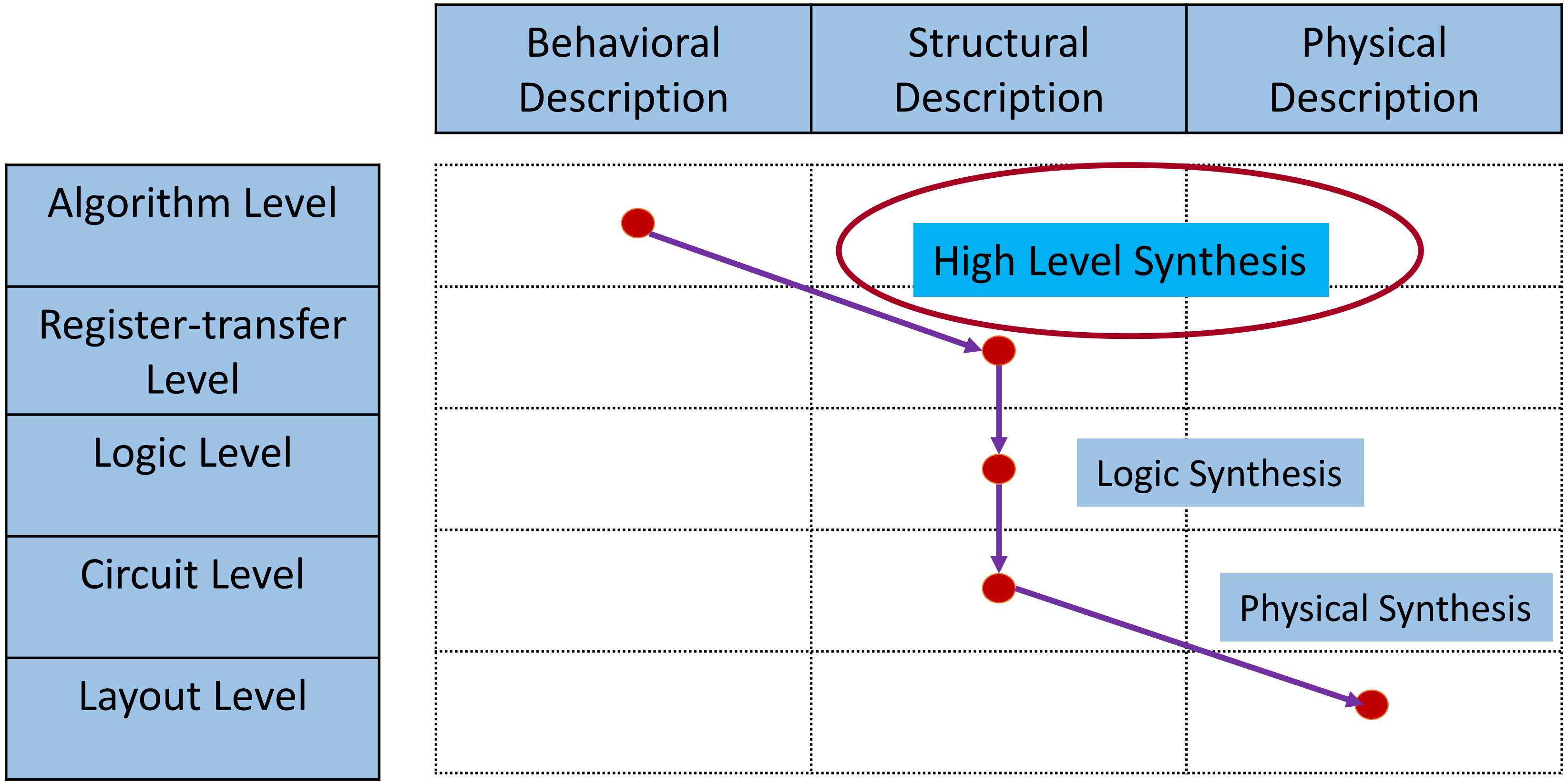
- Objectives
- Introduction
- Motivational Example
- Proposed Design Space Explorer
- Experiment Results
- Conclusion

Objectives

- In this paper, the main objectives can be summarized as follows：
  - To investigate the quality of the exploration results when using the results (particularly area) reported after HLS to guide the explorer in finding the true Pareto-optimal design (after logic synthesis).
  - To propose a dedicated DSE for FPGAs based on a pruning with adaptive windowing method using a Rival Penalized Competitive Learning (RPCL) model to extract the design candidates to further (logic) synthesized.

Introduction: HLS Overview

- High Level Synthesis



Behavioral Description

Structural Description

Physical Description

Algorithm Level

Register-transfer Level

Logic Level

Circuit Level

Layout Level

High Level Synthesis

Logic Synthesis

Physical Synthesis

VIVADO

ALTERA PREFERRED BOARD FOR OpenCL™

CyberWorkBench® Pioneering C-based LSI Design

Mentor Graphics® Catapult-C

cadence™ CtoS

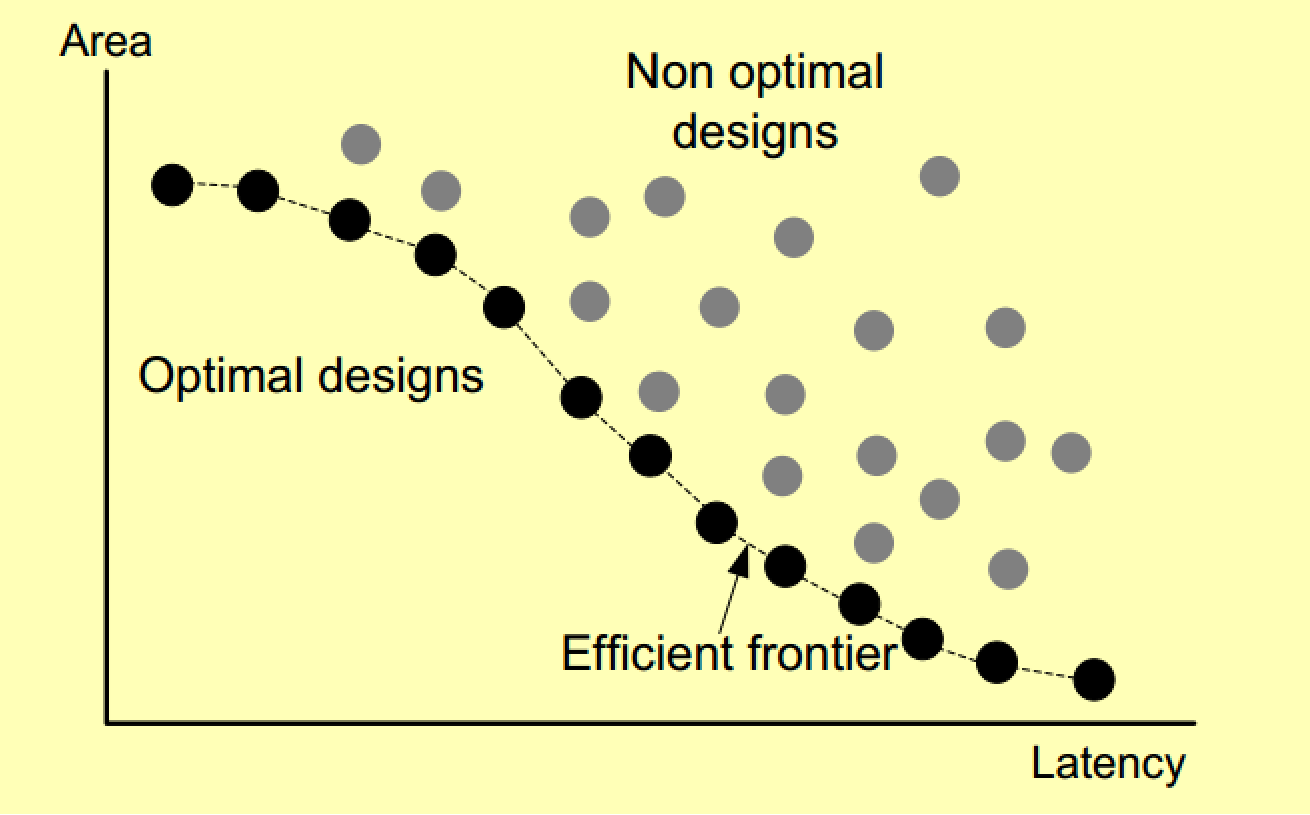
LegUp

Synphony C Compiler

Introduction: HLS Advantages

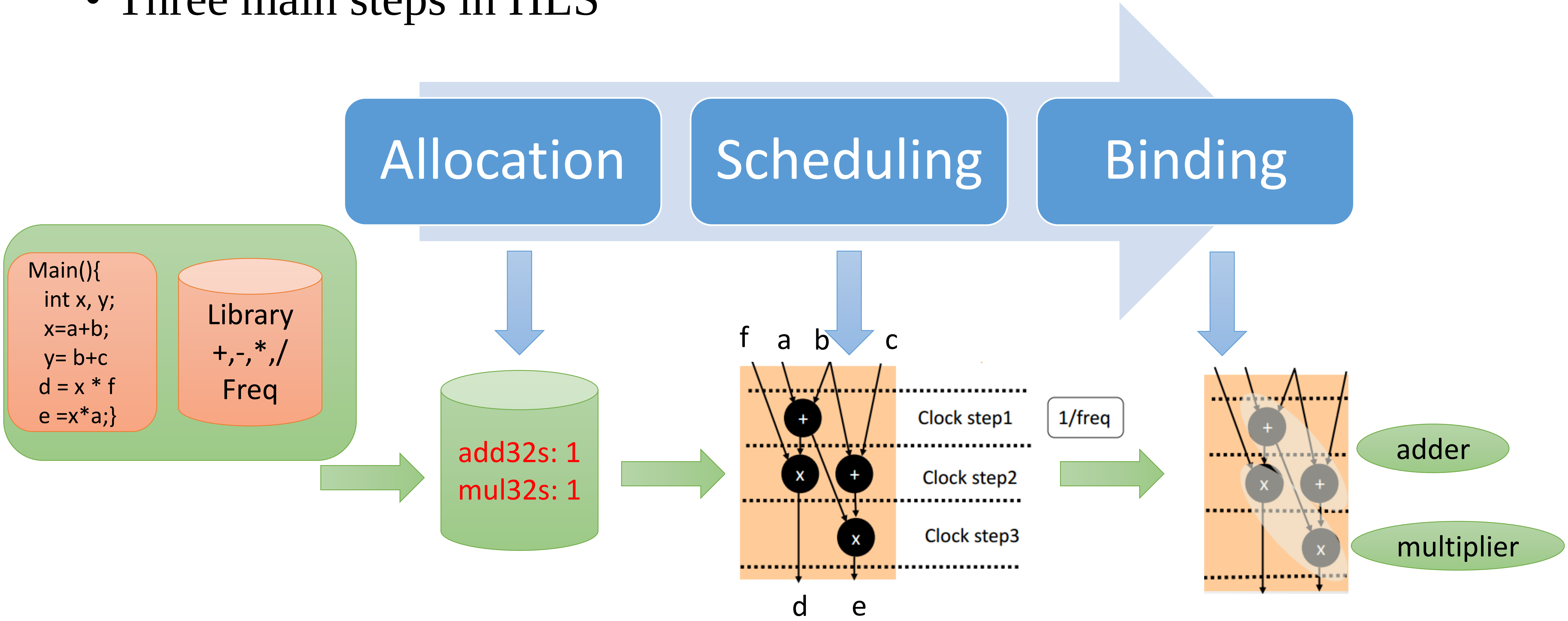
- Many advantages over traditional RTL based design
- One distinct advantage of HLS
  - Micro-architectural DSE
    - Design Space: Set of feasible designs
    - Objectives
      - Performance (Latency, throughput)
      - Area
      - Power

```
/*pragma unroll_times = all*/
  for (i = 7; i > 0; i--) {
        fifo[i] = fifo[i- 1];
 }
   fifo[0] = in0;
   sum= fifo[0];
```



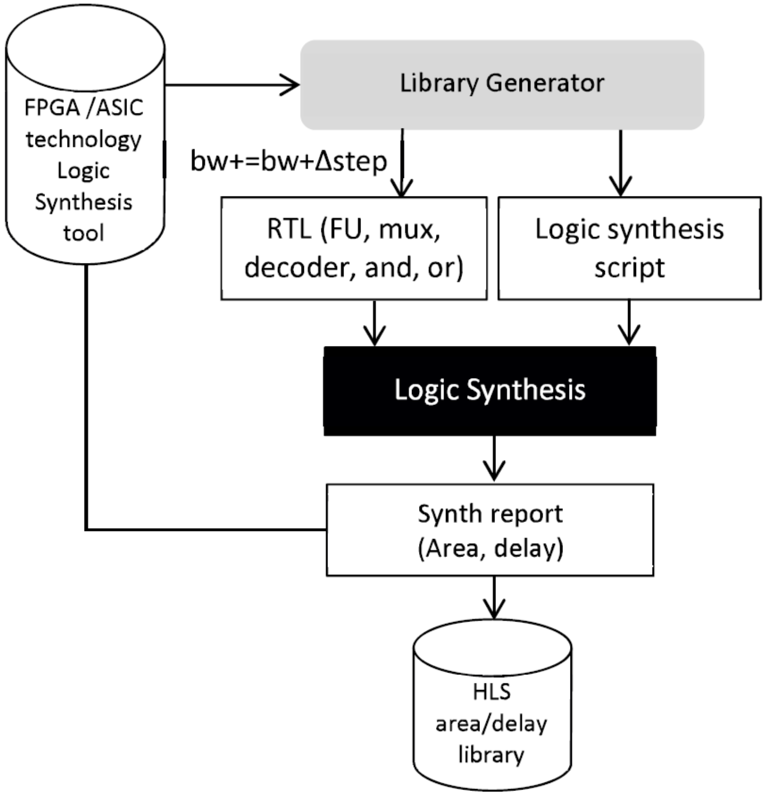
High-Level Synthesis Flow

- Three main steps in HLS



High-Level Synthesis Library Generator

- Importance of library generator (LIBGEN) on delay and area
  - To assist to successfully schedule operations in a control step
  - To provide the area and delay information of FUs from logic synthesis (LS) report
  - Notes: FPGA vendors provide pre-characterized libraries for their own FPGA


FPGA /ASIC technology Logic Synthesis tool
Library Generator
bw+=bw+Δstep
RTL (FU, mux, decoder, and, or)
Logic synthesis script
Logic Synthesis
Synth report (Area, delay)
HLS area/delay library

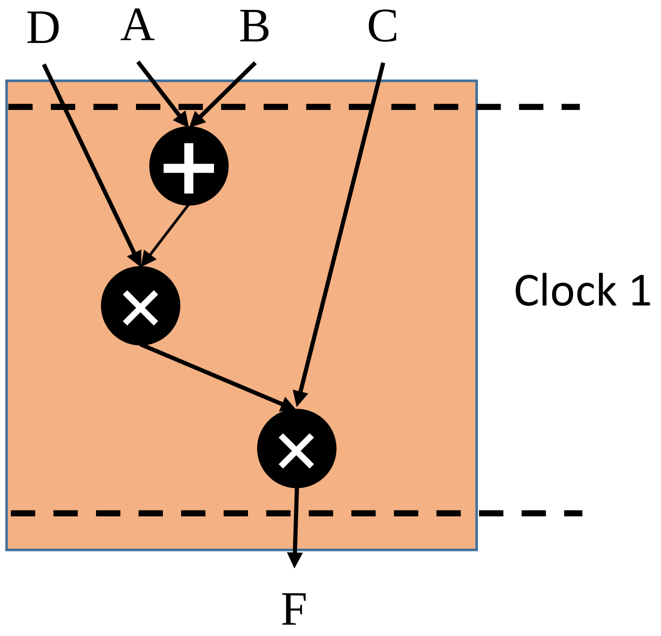
- **Overview of LIBGEN**
  - ***Step1:*** Generate RTL code for basic primitives (adders. decoder....)
  - ***Step 2:*** Perform logic synthesis and extract area and delay data
  - ***Step 3:*** Repeat ***Step 1*** & ***Step 2*** for different bit-widths of the same primitives

High-Level Synthesis Library Generator Importance

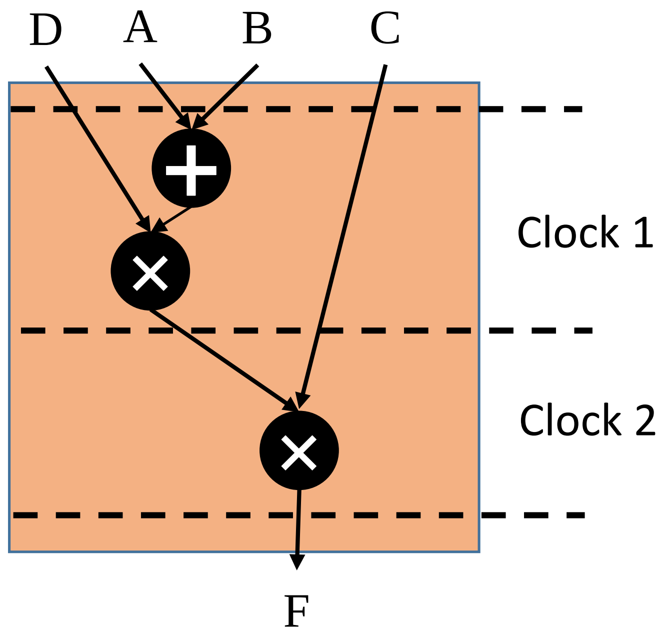
- Example of impact of LIBGEN to scheduling step (Latency)

X = A+B
E= X*D
F = E*C

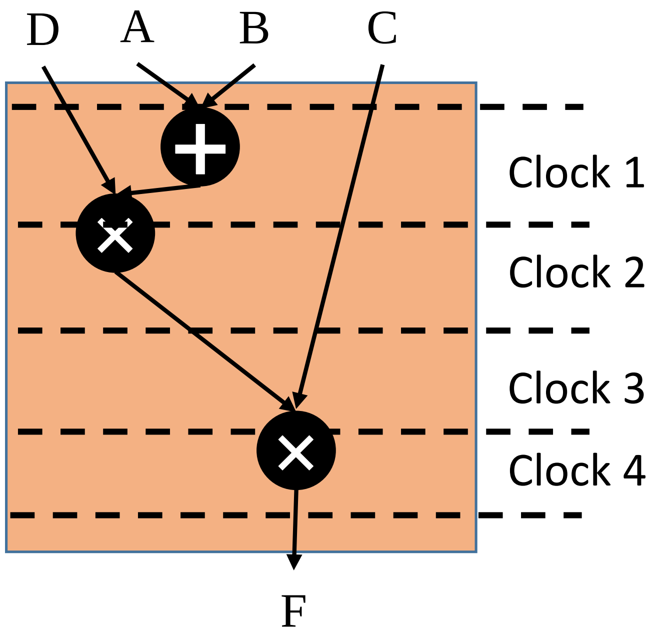
1/freq	12 ns
delay of ⊗	5ns
delay of ⊕	2 ns



1/freq	12 ns
delay of ⊗	10ns
delay of ⊕	2 ns



1/freq	12 ns
delay of ⊗	20ns
delay of ⊕	2 ns



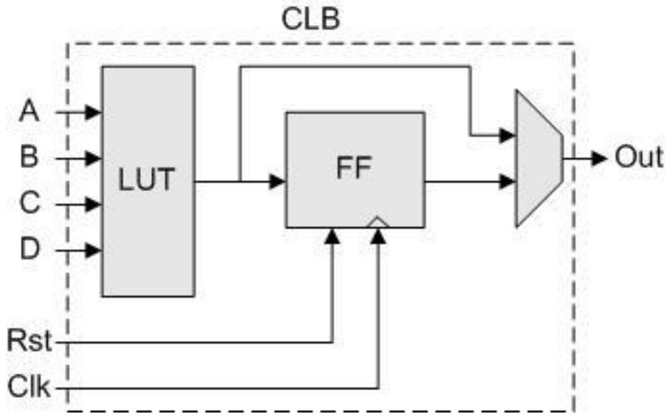
Note: enough FUs are provided

High-Level Synthesis Library Generator

- Limitations/Drawbacks of area estimation of LIBGEN
  - How the LS synthesize different FUs is unknown, e.g. different types of adders
  - Rough estimation: the area reported by HLS tool is only the sum of areas of all basic primitive

$$Area = Area(FU) + Area(MUX) + Area(DEC) + Area(MISC)$$

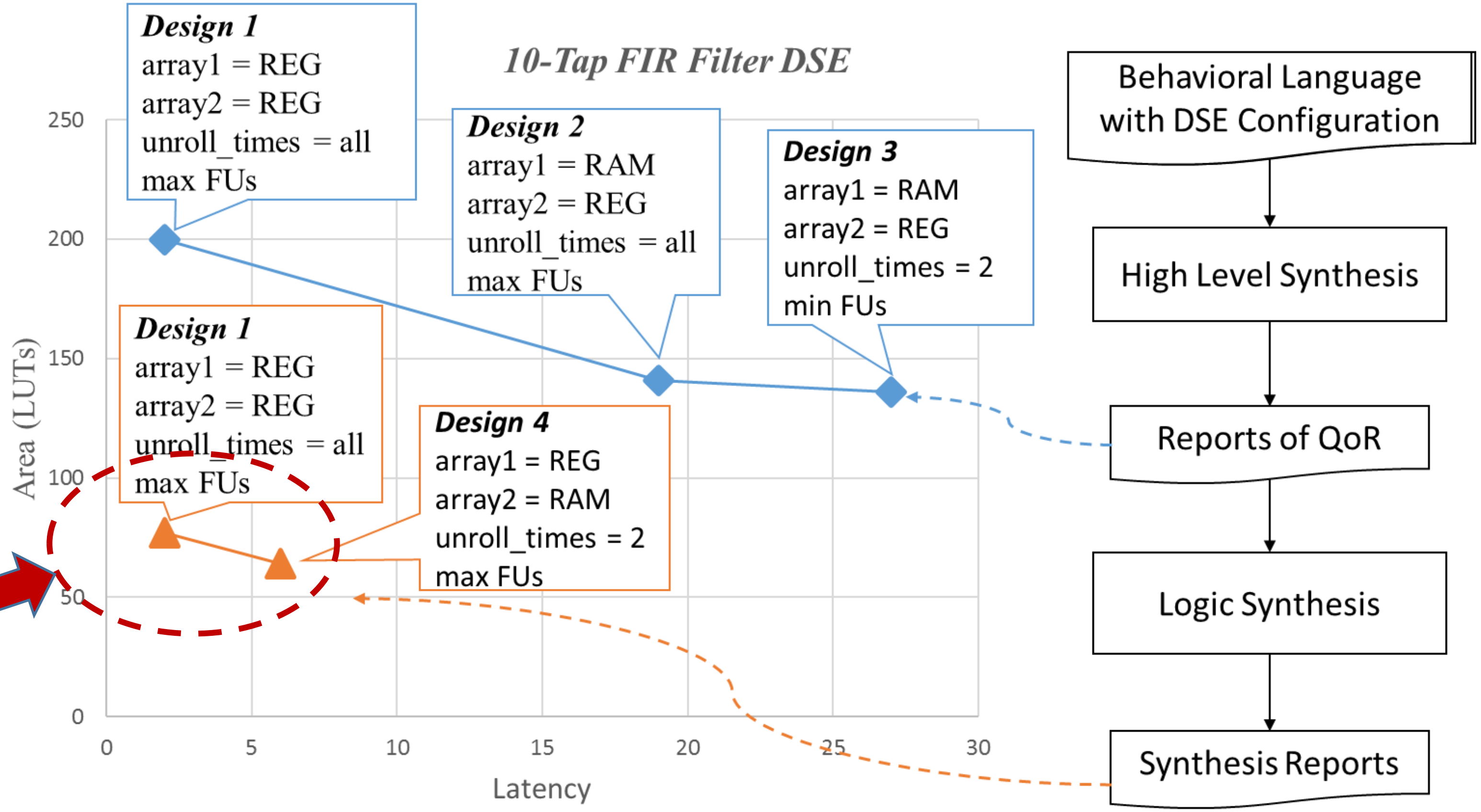
  - For FPGA, estimation is not accurate since the LS tools may merge multiple of basic primitives into one same LUT
  - Also, FPGAs have hard-macros which HLS tool need to consider



Motivational Example

- DSE Results (Area vs. Latency) of 10-tap FIR filter with HLS and Logic Synthesis

```
//fir.c
…
ary [] = {} /*pragma array = ?*/;
Coeff[] = {} /* pragma array = ?*/;
…
/*pragma unroll_times = ?*/
for (i = 0; i<10; i++)
    sum+= ary[i] * coeff[i];
```



Design 1
array1 = REG
array2 = REG
unroll_times = all
max FUs

Design 2
array1 = RAM
array2 = REG
unroll_times = all
max FUs

Design 3
array1 = RAM
array2 = REG
unroll_times = 2
min FUs

Design 1
array1 = REG
array2 = REG
unroll_times = all
max FUs

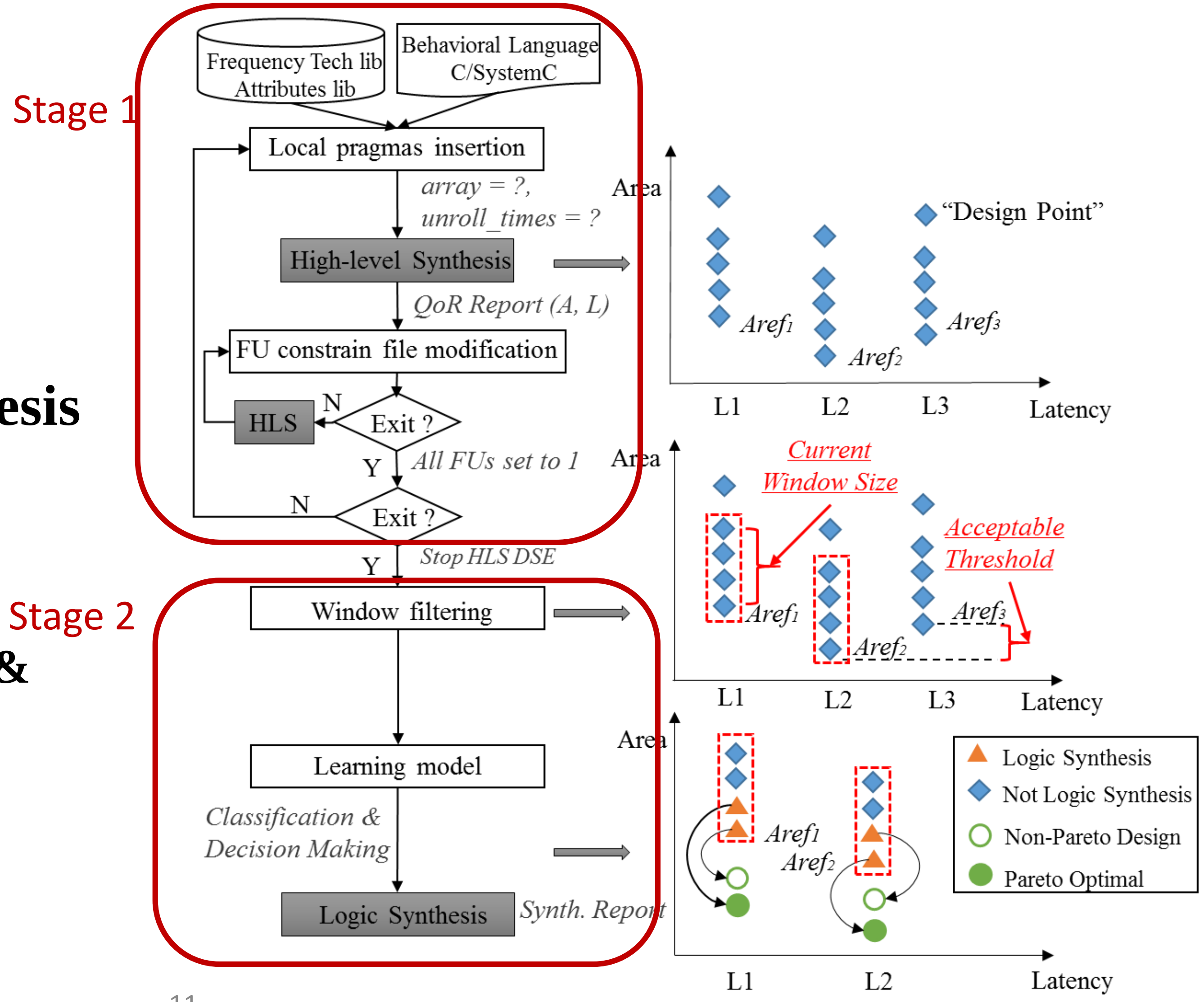
Design 4
array1 = REG
array2 = RAM
unroll_times = 2
max FUs

True Pareto-optimal Designs

Proposed Design Space Explorer

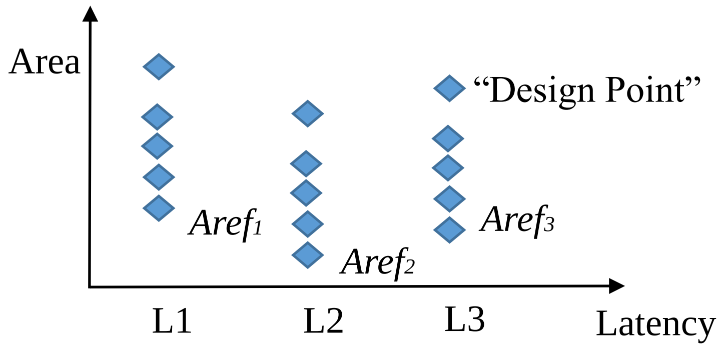
- Design flow overview
  - Stage 1: HLS exploration
  - **Stage 2: Pruning and Logic Synthesis**
    - **A. Pruning: Sorting + Windowing**
    - **B. Learning Model of Classification & Decision Making**

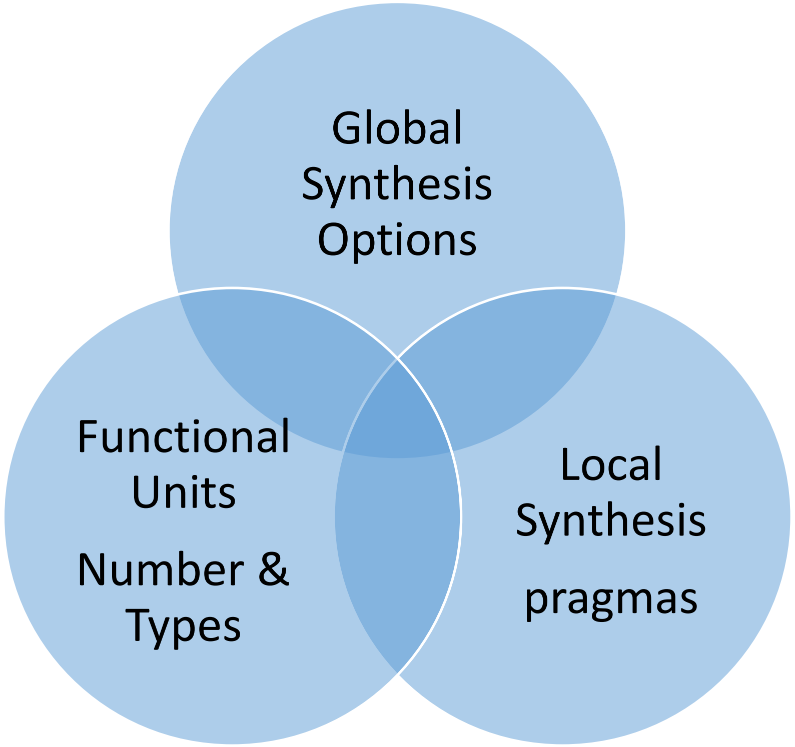
Main Contribution



Proposed Design Space Explorer

- Stage 1: HLS exploration
  - Use any existing heuristic (SA, GA, ACO)
  - Objectives: Store all the designs generated in this stage, to be used at the next stage

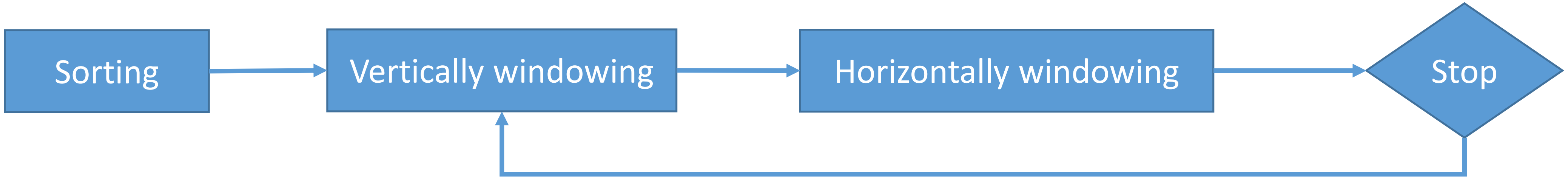


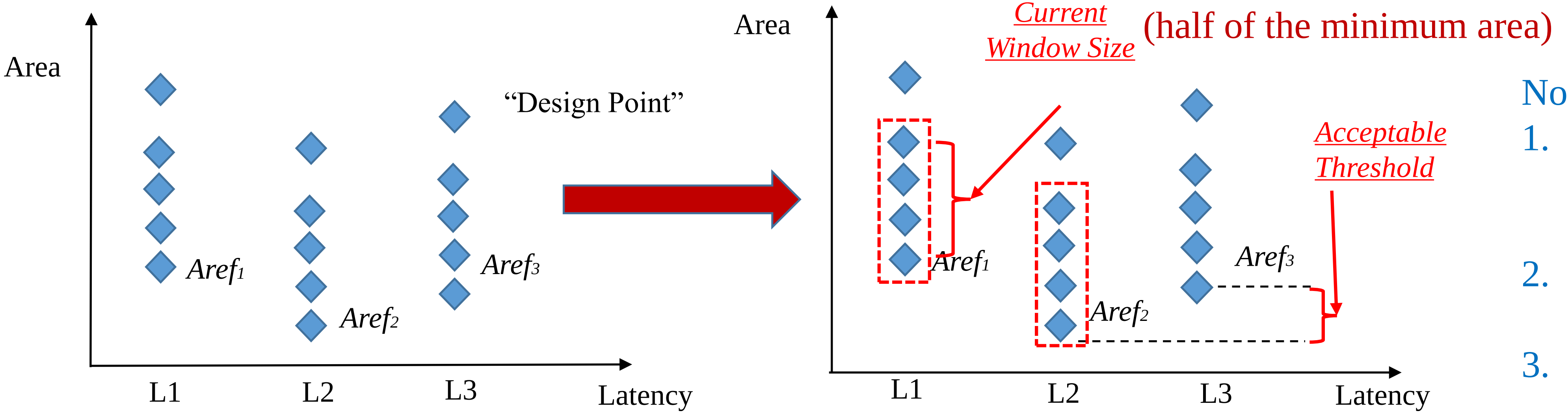


Global	Frequency	1000MHz, 2000MHz…
	Scheduling mode	Manual, automatic, automatic pipeline
FU	Type	adder, multiplexer, subtractor...
	Number	0 to 100
Pragmas	Array	RAM, ROM, EXPAND, LOGIC, REG
	Loop	unroll_times, folding
	Function	inline, goto

Proposed Design Space Explorer

- Stage 2A: Pruning: Sorting with Windowing
  - Algorithm Description



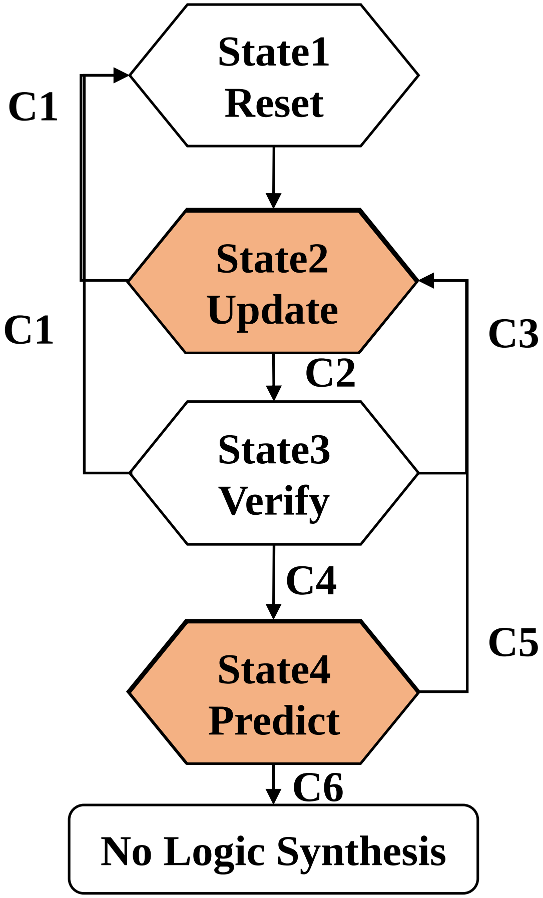


Notes:

1. The window size determine the size of training set.
2. Best training case: 3 designs
3. Worst training case: all designs

Proposed Design Space Explorer

- Stage 2B: Learning Model of Classification & Decision Making
  - State Transition Diagram of Learning Model



STATE	S1	Reset the score sheet and renew the design with smallest area of Synth. Rept.
	S2	Update the score sheet
	S3	Verify the score sheet
	S4	Predict the detection to perform logic synthesis
CONDITION	**C1**	If smallest (Area) design can be found
	C2	If smallest (Area) design cannot be found
	C3	If score sheet fail to make decision (Verify fail)
	C4	If score sheet success to make decision (Verify done)
	C5	If score sheet decide to perform logic synthesis
	C6	If score sheet decide not to execute logic synthesis

Proposed Design Space Explorer

- Before introducing model, predictors is shown
- Predictor values taken from HLS report

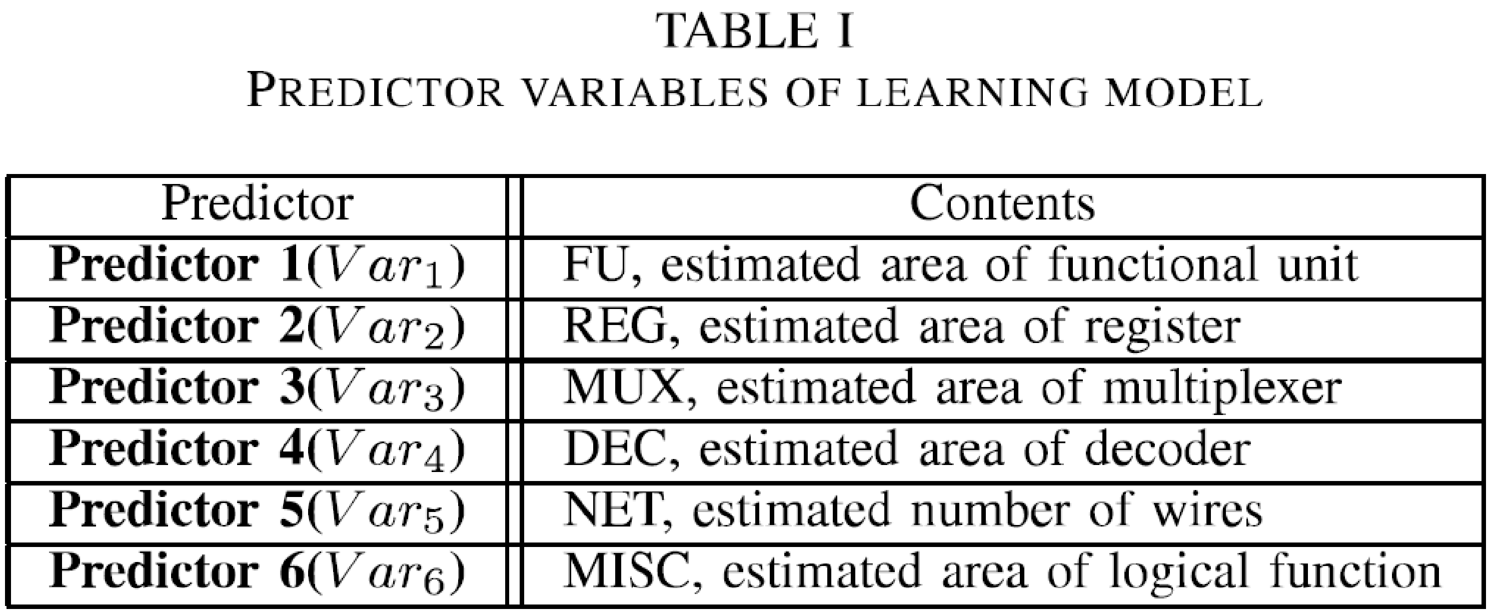


TABLE I
PREDICTOR VARIABLES OF LEARNING MODEL

Predictor	Contents
Predictor 1(Var_1)	FU, estimated area of functional unit
Predictor 2(Var_2)	REG, estimated area of register
Predictor 3(Var_3)	MUX, estimated area of multiplexer
Predictor 4(Var_4)	DEC, estimated area of decoder
Predictor 5(Var_5)	NET, estimated number of wires
Predictor 6(Var_6)	MISC, estimated area of logical function

Proposed Design Space Explorer

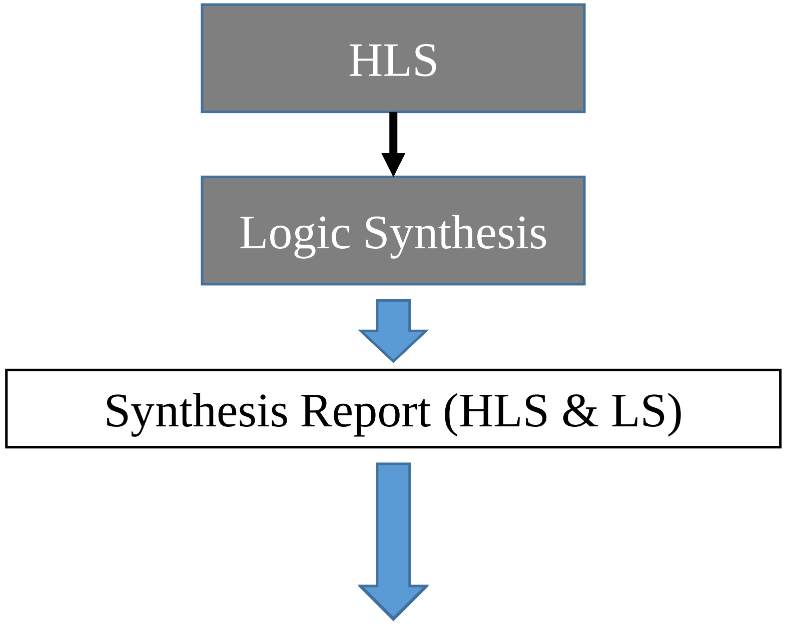
- Stage 2B – Updating Score Sheet State
  - RPCL model: Score sheet

$$Score(i)_{update} = \begin{cases} Score(i)+1, & \text{if trend not changed} \\ Score(i)-1, & \text{if trend changed} \end{cases}$$

State: Updating

Score(1)	Score(2)	Score(3)	Score(4)	Score(5)	Score(6)
-1	*0*	*-1*	*0*	*0*	*0*

Design Count: 2



HLS

Logic Synthesis

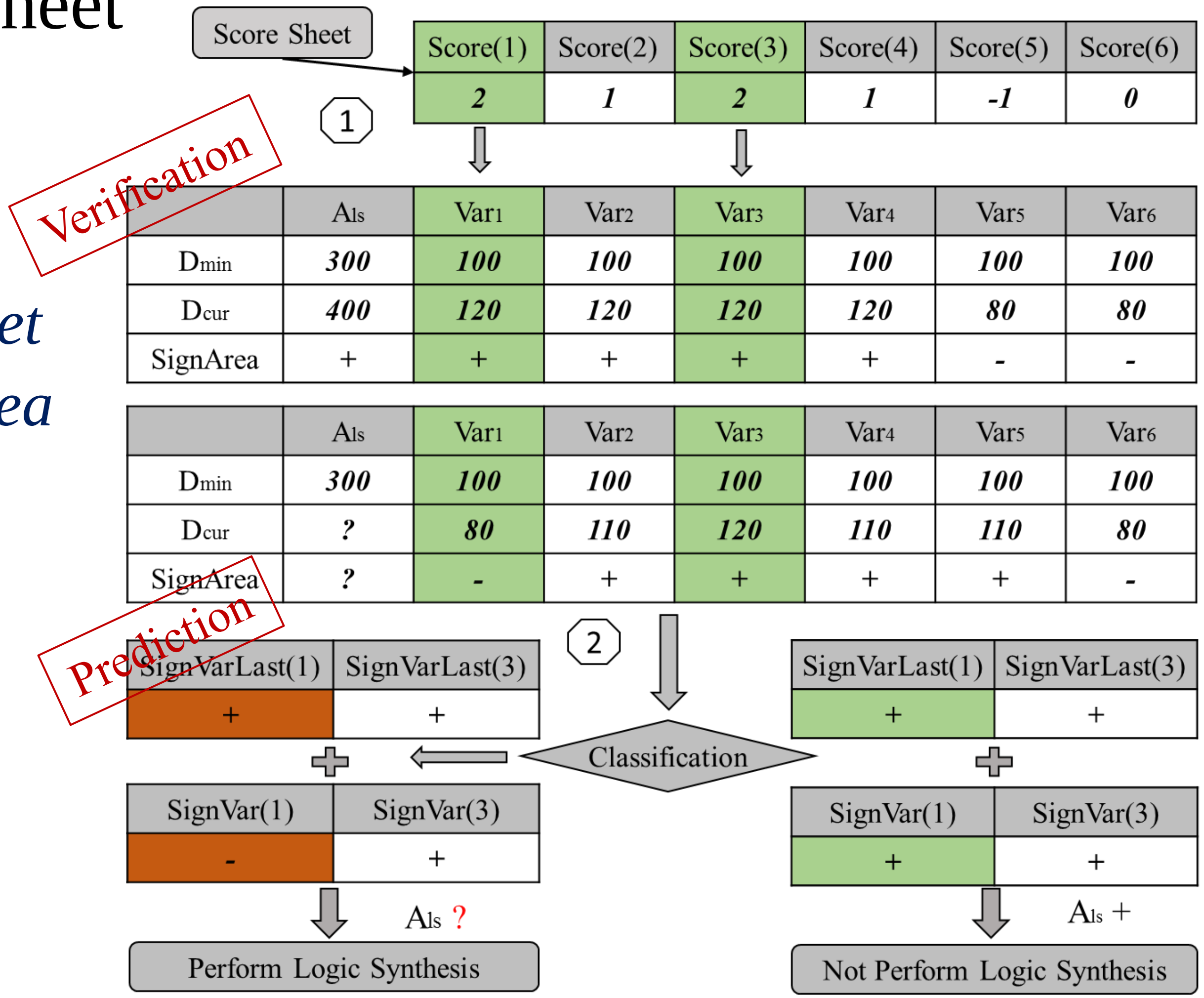
Synthesis Report (HLS & LS)

	A_{ls}	Var_1	Var_2	Var_3	Var_4	Var_5	Var_6
D_{min}	*300*	*100*	*100*	*100*	*100*	*100*	*100*
D_{cur}	*350*	*180*	*120*	*190*	*120*	*80*	*85*
SignArea	+	-	+	-	+	-	-

Proposed Design Space Explorer

- Stage 2B – Prediction State with Score Sheet
  - Schematic Diagram of Prediction State in Learning Model
  - *Step 1: Select variable in terms of score sheet*
  - *Step 2: Calculate the alteration of actual area*
  - *Step 3: Classify the design candidates*
  - *Step 4: Make the decision of performing the Logic Synthesis*

Note: the difference between verification state and Prediction State is the order between performing LS and using score sheet to do prediction



Score(1)	Score(2)	Score(3)	Score(4)	Score(5)	Score(6)
2	*1*	*2*	*1*	*-1*	*0*

	A_{ls}	Var_1	Var_2	Var_3	Var_4	Var_5	Var_6
D_{min}	*300*	*100*	*100*	*100*	*100*	*100*	*100*
D_{cur}	*400*	*120*	*120*	*120*	*120*	*80*	*80*
SignArea	+	+	+	+	+	-	-

	A_{ls}	Var_1	Var_2	Var_3	Var_4	Var_5	Var_6
D_{min}	*300*	*100*	*100*	*100*	*100*	*100*	*100*
D_{cur}	*?*	*80*	*110*	*120*	*110*	*110*	*80*
SignArea	*?*	-	+	+	+	+	-

SignVarLast(1)	SignVarLast(3)
+	+

SignVar(1)	SignVar(3)
-	+

A_{ls} ?

Perform Logic Synthesis

SignVarLast(1)	SignVarLast(3)
+	+

SignVar(1)	SignVar(3)
+	+

A_{ls} +

Not Perform Logic Synthesis

Experiment Results

- Experiment detail
  - Benchmarks from S2CBench (www.s2cbench.org)

fir	adpcm	kasumi	snow3G	decimation	md5C

  - Three methods

HLS + LS	HLS + LS opt	Proposed DSE
LS for each designs	LS for only optimal design of HLS	Proposed method in this paper

  - Experiment Setup

Simulation Computer	HLS tool and LS tools	Target FPGA
Intel Xeon2 processor running at 2.4GHz with 16G RAM running Linux Fedora Core 20	NEC CyberWorkBench v.5.5 Xilinx ISE v14.3	Xilinx Virtex 5 FPGA XCVFS100T

* www.s2cbench.org

Experiment Results

- Criteria for measuring the **quality** of experiment results

Indicators	***Definition***	***Evaluation***
Average Distance from Reference Set (ADRS)	How close a Pareto-front is to the reference front	The lower ADRS, the better
Pareto Dominance (Dom)	The ratio between the total number of designs in the Pareto set being evaluated	The higher Dom, the better
Cardinality (Card)	The number of dominating designs found by each method, indicate the number of design to chose from	The high Card, the better

- Criteria for measuring the **quantity** of experiment results
  - Running Time

Experiment Results

- Detailed results (quality)

Accurate Method

Fast Method

	HLS + LS				HLS + LS opt				Proposed DSE			
Bench	ADRS	Dom	Card	Run[s]	ADRS	Dom	Card	Run[s]	ADRS	Dom	Card	Run[s]
fir	0	1	2	5,428	0.2	0.5	1	770	0	1	**2**	780
adpcm	0	1	5	6,829	0.31	0.6	4	914	0.18	0.8	**5**	4,458
kasumi	0	1	4	35,028	0.17	0.75	3	1,415	0.06	0.75	**4**	3,944
snow3G	0	1	3	94,600	0.36	0	2	2,243	0.03	0.67	**3**	13,234
decimation	0	1	10	469,972	0.15	0.6	9	7,801	0	1	**10**	39,617
md5c	0	1	12	401,128	0.43	0.75	10	22,900	0.37	0.92	**12**	41,811
Avg	0	1	6	-	0.27	0.53	4.83	-	0.1	0.86	6	-
Geomean	-	-	-	53,387.93	-		-	2,713.32	-	-	-	8,184.76

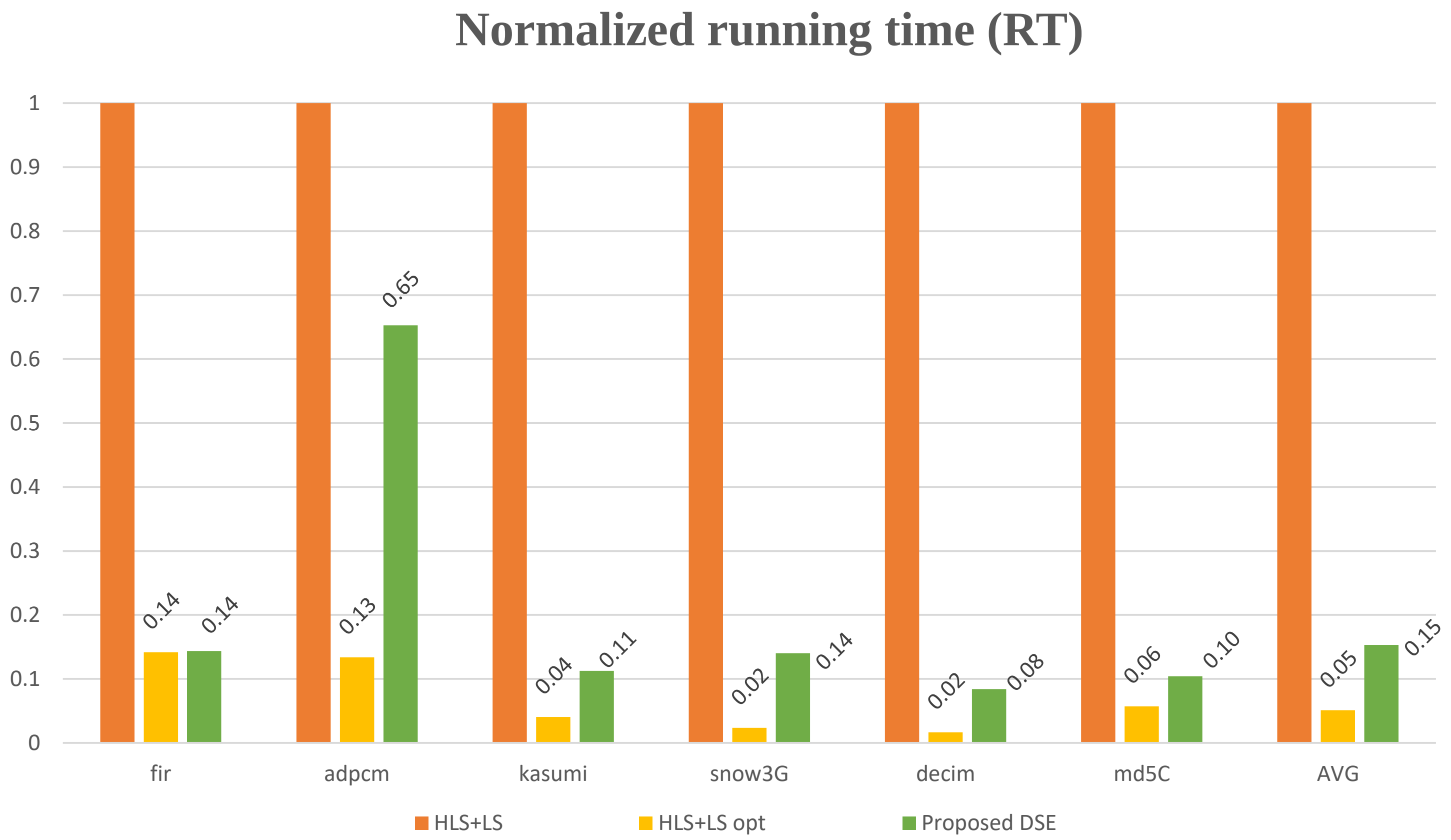
lower

higher

higher

Experiment Results

- Running times comparison (quantity) Acceptable



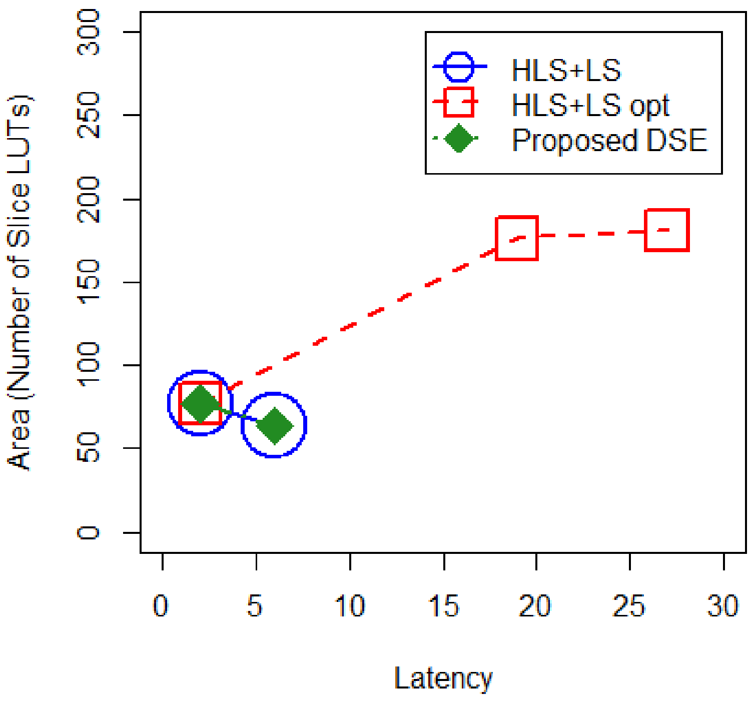
- Average Running Time Speedup

Ref.	Proposed DSE
HLS + LS	6.5 X faster
HLS + LS opt	3.0 X slower

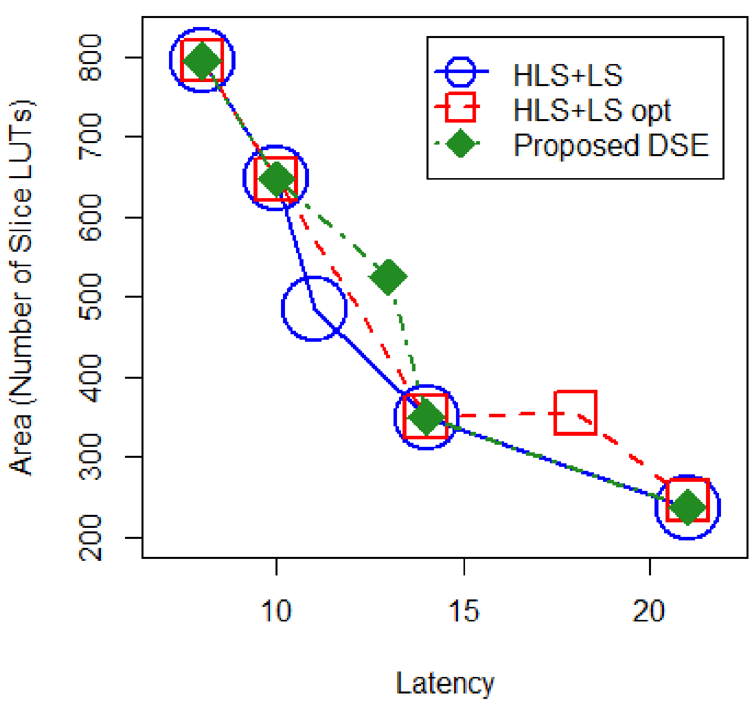
Experiment Results

- Detail of Pareto-sets (1)

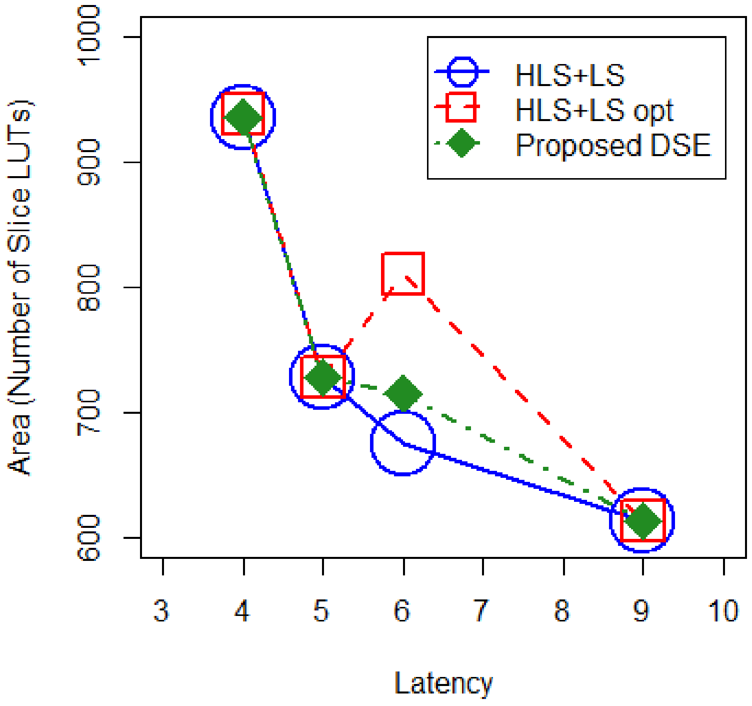
fir



adpcm



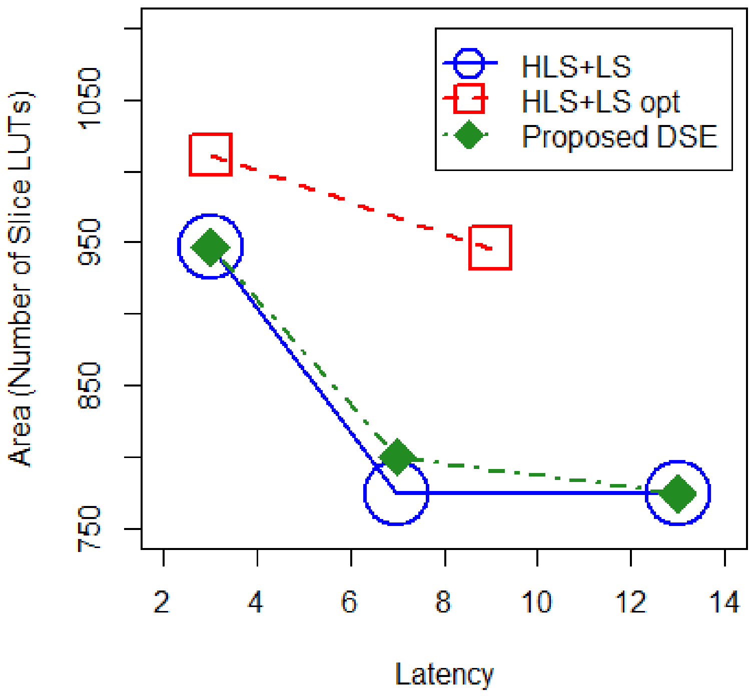
kasumi



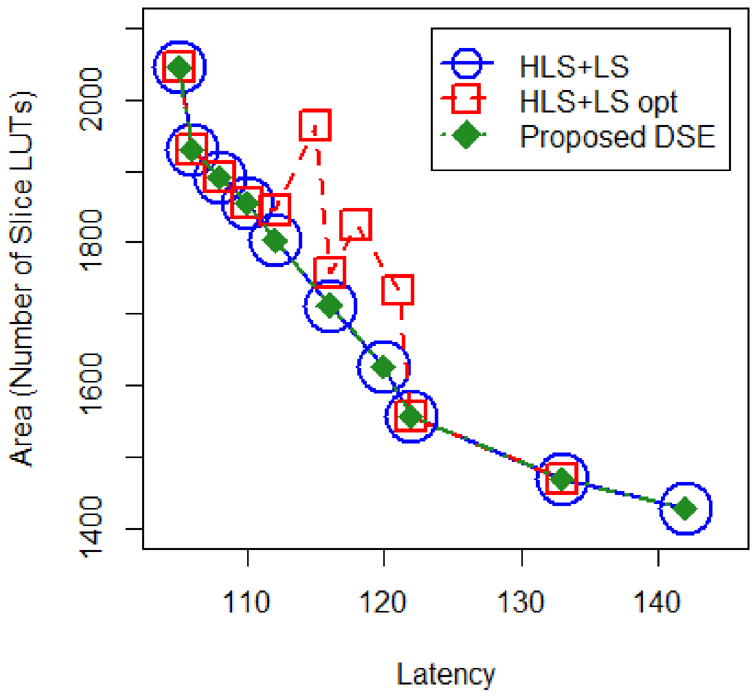
Experiment Results

- Detail of Pareto-sets (2)

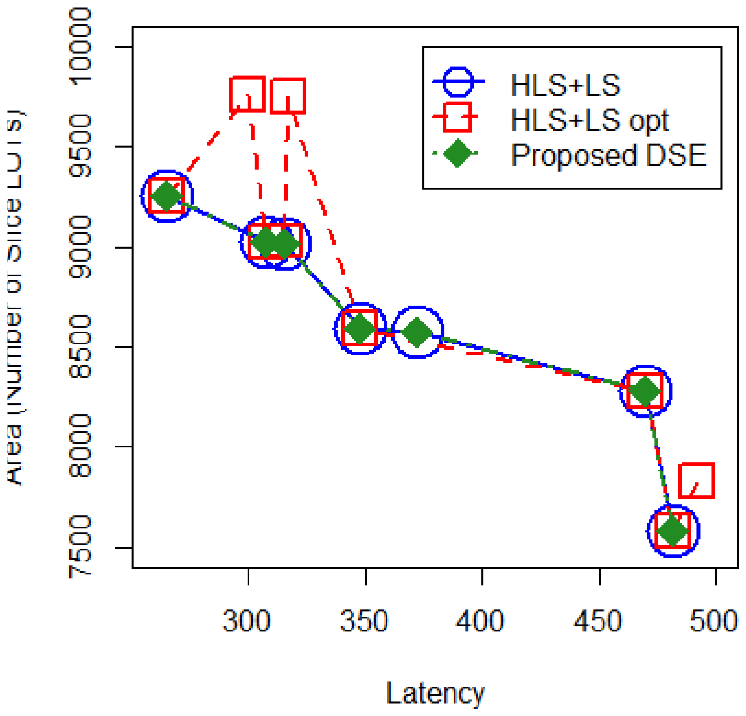
snow3G



decimation



md5c



Conclusion

- In this work, we have presented a HLS DSE for FPGA

1. Firstly, it is motivated that a dedicated explorer for FPGAs is needed in order to accurately predict if logic synthesis is required or not
2. A method based on RPCL learning model is introduced
3. Results show the proposed method is much better than just using the report from HLS tools.
4. Also, the proposed DSE can generate the trade-off curve of similar quality to the ones generated by performing LS for each designs, at a fraction of running time.

Thanks for Your Attention

Q & A

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (PolyU252000)

References

[1] Xilinx. Vivado HLS.

[2] Altera. Altera OpenCL SDK.

[3] V. Krishnan and S. Katkoori, “A genetic algorithm for the design space exploration of datapaths during high-level synthesis,” IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, pp. 213–229, June 2006.

[4] M. Holzer, B. Knerr, and M. Rupp, “Design space exploration with evolutionary multi-objective optimisation,” in Industrial Embedded Systems, 2007. SIES ’07. International Symposium on, July 2007, pp. 126–133.

[5] H.-Y. Liu and L. P. Carloni, “On learning-based methods for design space exploration with high-level synthesis,” in Proceedings of the 50thAnnual Design Automation Conference, ser. DAC ’13. New York, NY, USA: ACM, 2013, pp. 50:1–50:7.

[6] B. C. Schafer, “Probabilistic multiknob high-level synthesis design space exploration acceleration,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 3, pp. 394–406, March 2016.

[7] G. Zhong, V. Venkataramani, Y. Liang, T. Mitra, and S. Niar, “Design space exploration of multiple loops on fpgas using high level synthesis,” in Computer Design (ICCD), 2014 32nd IEEE International Conference on, Oct 2014, pp. 456–463.

[8] W. Sun, M. J. Wirthlin, and S. Neuendorffer, “Fpga pipeline synthesis design exploration using module selection and resource sharing,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no. 2, pp. 254–265, Feb 2007.

[9] B. Carrion Schafer and K. Wakabayashi, “Machine learning predictive modelling high-level synthesis design space exploration,” IET computers & digital techniques, vol. 6, no. 3, pp. 153–159, 2012.

[10] M. Kuhn and K. Johnson, Applied predictive modeling. Springer, 2013.

[11] L. Xu, A. Krzyzak, and E. Oja, “Rival penalized competitive learning for clustering analysis, rbf net, and curve detection,” IEEE Transactions on Neural Networks, vol. 4, no. 4, pp. 636–649, Jul 1993.

[12] B. Schafer and A. Mahapatra, “S2cbench: Synthesizable systemc benchmark suite for high-level synthesis,” Embedded Systems Letters, IEEE, vol. 6, no. 3, pp. 53–56, Sept 2014.

[13] NEC CyberWorkBench. (2015). [Online]. Available: www.cyberworkbench.com

[14] Xilinx: All Programmable. (2015). [Online]. Available: http://www.xilinx.com/products/design-tools/ise-design-suite.html

[15] C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler, “A tutorial onthe performance assessment of stochastic multiobjective optimizers,” in Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), vol. 216, 2005, p. 240.

Biography of presenter

Dong Liu received the B. Eng (Hons) in Electronic Engineering with First Class from the Hong Kong Polytechnic University, Hong Kong, in 2014. He is currently perusing the Ph. D degree in the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong.

His research interests now include, modeling of circuit and system, complex network application, Programmable hardware implementation