Exploration with Upgradeable Models Using Statistical Methods for Physical Model Emulation

Bailey Miller
Dept. of Computer Science and Engineering
University of California, Riverside
bmiller@cs.ucr.edu

Frank Vahid
Dept. of Computer Science and Engineering
University of California, Riverside
Also with CECS, UC Irvine
vahid@cs.ucr.edu

Tony Givargis
Center for Embedded Computer Systems (CECS)
University of California, Irvine
givargis@uci.edu

ABSTRACT

Physical models capture environmental phenomena such as biochemical reactions, a beating heart, or neuron synapses, using mathematical equations. Previous work has shown that physical models can execute orders of magnitude faster on FPGAs (Field-Programmable Gate Arrays) compared to desktop PCs. Different models of the same physical phenomenon may vary, with "upgraded" models being more accurate but using more FPGA area and having slower performance. We propose that design space exploration considering upgradable models can dramatically increase the useful design space. We present an analysis of the solution space for utilizing networks of processing-elements (PEs) on FPGAs to emulate physical models, implement a web-based frontend to a compiler and cycle-accurate simulator of PE networks to estimate solution metrics, and utilize design-of-experiments (DOE) statistical methods to identify Pareto points. By considering upgradeable models during the design space exploration of a human lung physical model, the solution space of possible speedup, area, and accuracy is increased by 6X, 7.3X, and 1.5X, respectively, compared to evaluating a single model.

Categories and Subject Descriptors

B.7.0 [**Integrated Circuits**]: General

General Terms

Design, Performance

Keywords

Design space exploration, FPGA, cyber-physical systems

1. INTRODUCTION

Fast physical model emulation is important in various domains for research and testing purposes. Iterative step-solvers can be used to compute the state of the physical model at real-time or faster than real-time speeds. Fast physical model emulation is especially important in the cyber-physical domain for testing purposes, since physical models can be used as a replacement for environments that are dangerous, expensive, or difficult to recreate.

Previous work has shown that emulating physical models on networks of processing-elements on FPGAs can provide orders of magnitude speedup over desktop processors and graphical processing units [2][3], due largely to parallel execution on tens/hundreds of processing elements. Such parallel execution is enabled by physical models consisting of independent and locally communicating equations. Previous work applied traditional design space exploration, partitioning equations among different types and numbers of processing elements to achieve area and performance tradeoffs.

However, physical systems provide a rather unique additional solution option. The same physical system can be modeled with different equations. Each model may have tradeoffs in terms of the number of equations, ease of computation, and accuracy. We denote sets of models that are functionally similar as *upgradeable* models, since a designer may 'upgrade' to a more accurate model at the expense of area and performance. For example, Figure 1(a) shows three models that capture the behavior of the same physical system of lung airway mechanics. A simple RC model can coarsely capture the behavior using a single ordinary differential equation (ODE). For higher accuracy, a binary-tree shaped Weibel model [13] with variable levels of complexity can be used at the expense of higher computational costs. Accuracy also depends on the step size and type of equation solver. A smaller step size yields higher accuracy but slower performance. Likewise, more accurate solvers yield slower performance. Figure 1(b) illustrates the accuracy of each model, where dashed lines represent some deviations in accuracy due to different step solvers or step sizes.

Upgradeable models substantially increase the solution space that must be explored, not only via expanding area and performance ranges, but also by adding the design metric of accuracy. Figure 2 shows various tradeoffs in terms of speedup, area, and accuracy for a set of upgradeable models.

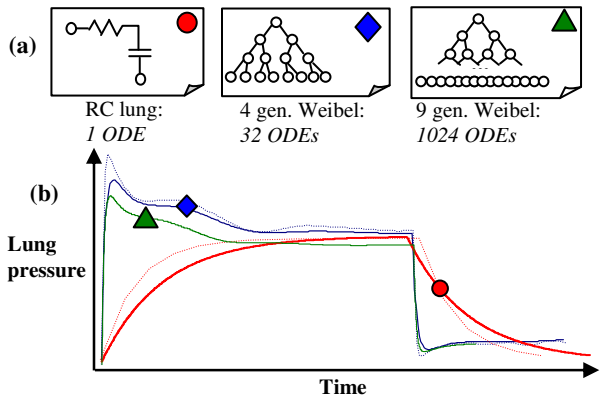

Figure 1: (a) A set of upgradeable models that implement similar lung airway mechanics behavior. (b) Relative accuracy of each model. Dashed lines show variations in accuracy when different solvers or step sizes are used.

DAC '13, May 29 – June 07 2013, Austin, TX, USA.

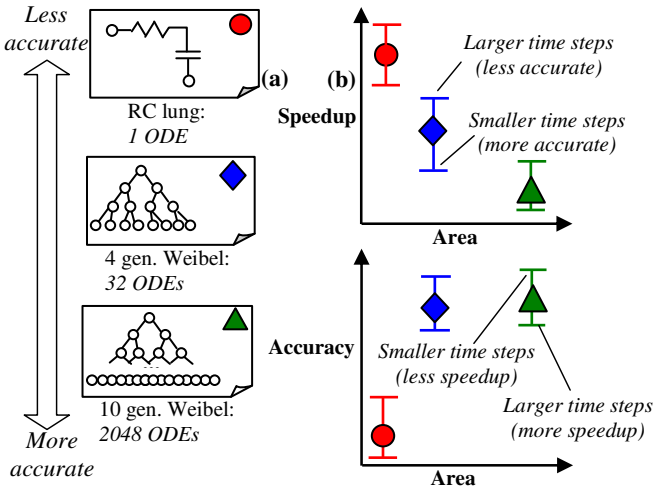

Figure 2: Impacts of various physical models on metrics. (a) Set of upgradeable models, and (b) comparisons of each model's speedup, area, and accuracy. The vertical scales show how solver time step affects speedup or accuracy.

Upgradable models introduce numerous additional parameters that influence and tremendously increase the design space. The influence on design metrics of those parameters can be complex and interdependent. To deal with these new parameters, we apply the statistical method known as design-of-experiments (DOE), which efficiently determines the impacts and dependencies of parameters, to enable efficient search of the design space. We introduce an approach that expands design space exploration to consider upgradeable models. To search the large design space, we utilize DOE statistical method to generate the Pareto points of the design. By generating the Pareto points, the design space can be pruned to enable a feasible exploration of solutions. We present a web-based tool that uses a processing-element (PE) network compiler and cycle-accurate simulator to automatically generate a PE network from an MML-language [7] based input model specification and evaluate the relevant size, performance, and accuracy metrics of PE network implementations. The web-based tool also supports automatic exploration of the design space using DOE techniques to aid in finding and appropriate model to use from an upgradeable set of models and an appropriate underlying PE network implementation that meets given constraints.

2. RELATED WORK

Previous research has demonstrated a compiler for translating MML-language based input specifications of physical models to VHDL descriptions of networks of PEs, which provides orders of magnitude speedup over standard desktop PCs [2]. Model equations are partitioned to PEs using various heuristics, scheduled to exploit parallelism, and connected via custom point-to-point networks. Various types of PE networks have been investigated, including homogeneous networks of programmable ALU-based general PEs, homogeneous networks of PEs with custom datapaths to solve specific equations, and heterogeneous networks consisting of both general and custom PEs [3]. Implementations of physical systems models on FPGAs exist elsewhere in literature [9], however all of these solutions are ad-hoc application specific solutions, whereas the PE network approach is a general CAD flow for any physical system model.

Considering how to select the appropriate physical model to emulate, to the best of our knowledge, has not been investigated elsewhere in literature. The problem however has analogs to other domains in the CAD literature. The work on algorithm selection is a close corollary [10], in which a suitable algorithm from a set of functionally equivalent algorithms must be selected in order to optimize a given goal such as throughput, cost, or power.

Design space exploration is a well-known and often addressed issue [1][11]. The generation of Pareto points or Pareto curves has long been of interest as a method for reducing the amount of system configurations that need to be considered. Givargis introduced Platune [1], a tool for automatically exploring the design space of System-on-Chip (SoC) architectures. Platune implements an automated algorithm to explore the design space of a given SoC, and provides estimations of the resulting solution. Sheldon improved on the automated exploration by using statistical methods to calculate platform independent parameter interdependencies [12]. Our work is similar to Platune, except that we specifically target PE network implementations.

3. UPGRADEABLE MODELS

Upgradeable models in the context of physical systems emulation refers to having multiple underlying sets of equations that are each able to emulate the same physical model, with tradeoffs among accuracy versus area and performance. In this section, we define how to determine if relative models are part of the same upgradeable set, and discuss size-scalable upgradeable models.

3.1 Functional equivalence

We consider different models to be a part of the same upgradeable set if they meet the following requirements:

1. The models contain the base input/output interface required to support the physical system behavior.
2. The models are functionally similar, i.e. they produce similar output for all possible inputs.

The first requirement ensures that all models can operate on the same inputs and can provide the same outputs. Physical model emulations are usually a part of a larger design, often for testing purposes, thus ensuring that all models in an upgradeable set have a similar interface ensures smooth transitions and reduces the potential to introduce new errors. Some small differences in interfaces may be acceptable, as long as a correct transformation is available. For example, a lung model may require either an air flow input or an air pressure input. Flow can be easily converted into air pressure, and vice versa, thus we may still consider the models to be functionally equivalent. Models may also provide a supplemental input/output interface, in addition to the required base interface. For example, the Weibel lung model of Figure 2(a) provides output pressures at each of the leaf nodes, whereas the RC lung model provides only a single output pressure node below the capacitor. The supplemental interface is not required, but may improve model accuracy or provide additional information about the internal model state to the designer.

The second requirement demands that all models in an upgradable set produce similar outputs for given inputs. This requirement ensures that the physical model being emulated is similar in functionality, despite any differences in the underlying equations that are computed. Similarity can be determined by both qualitative and quantitative methods. Figure 1(b) shows the output of three various lung models; the models are considered interchangeable because they all produce an output of the same physical system, yet they have different quantitative and qualitative measures of accuracy. A designer can either determine that models are close enough to be functionally interchangeable, or a distance measure could be automatically calculated.

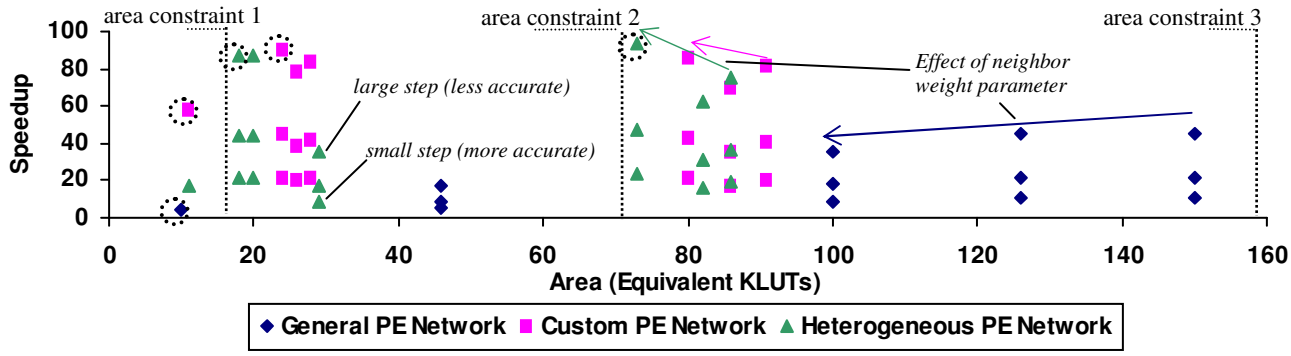

Figure 3: Possible PE networks for lung model. The three vertical lines indicate area constraints. Each solution is shown with a step size of 1e-2, 0.5e-2, and 0.25e-2 milliseconds. Arrows show the effect of the compiler neighbor weight parameter. Dashed circles show possible area/performance metric Pareto points.

3.2 Scaling model size

Many physical models have a common, repeating pattern or structure. The previously introduced Weibel model has a binary tree structure, which resembles the 23 bifurcating branches of a human lung. Neuron or cell models can consist of hundreds or thousands of individual elements that are connected to neighboring elements in mesh or grid structures. Previous work has shown that physical model structure can even be utilized to aid placement of PE networks on FPGA fabrics [6]. Physical models with regular structures can be considered upgradeable if the physical model can be scaled in size by adding new elements into the structure.

Scaling a physical model may or may not affect the accuracy of the model, but certainly impacts the resulting area and performance of the implementation. The Weibel model can be scaled in size to have more or less tree generations - having more generations implies a higher level of accuracy because the number of branches is closer to actual lung physiology. However, doubling the number of cells in a cell tissue model does not necessarily imply that the equations of each individual cell are more accurate. Even so, a designer may want to know how many cells can be included, given some area or performance constraints.

4. PE NETWORK PARAMETERS AND METRICS

For a given physical model of sufficient size and complexity, the solution space for a PE network that emulates the model is extremely large. This is due mostly to the parameters available during PE network synthesis, and partly to the non-deterministic heuristics used during equation partitioning. The considered key parameters are the model specification itself, PE network type, equation partitioning neighbor function weightings, the given resource constraints, step size, and solver type. The key solution metrics are FPGA area (LUTs, memory, and DSP usage), performance (speedup over real-time), and accuracy (closeness to exact solution).

Figure 3 depicts a chart of possible PE network solutions for a neuron model with 300 equations. Three sections are shown which depict solutions yielded by using different area constraints. For each area constraint, the PE network type, neighbor weight, and step size parameters are varied. Area/speedup metric Pareto points are circled; the accuracy metric is not measured explicitly in the figure, though points towards the bottom typically use smaller time steps and thus would be more accurate.

The model itself is an important parameter. The PE network solutions that are generated depend highly on how many equations are in the model, the complexity of each equation, and the data dependencies between the equations. The user also must specify a coefficient that quantitatively captures the quality of the model compared to the others in the set. For example, the 9 generation Weibel lung model may be considered to be the most accurate, and have a coefficient of 1. The RC model may be considered to have a coefficient of 0.4, because it only coarsely captures realistic behavior. The coefficients are used when comparing relative accuracies of different models.

PE type is a critical parameter that determines the type of PE network that is generated to emulate the model. There are three options: homogeneous general PE network, homogeneous custom PE network, and heterogeneous PE network. A general PE is a flexible, programmable, ALU-based processor that can solve any equation. A custom PE uses a pipelined datapath to solve a single specific equation much faster than a general PE, but may use more FPGA resources and incur higher routing congestion cost. Heterogeneous PE networks combine general and custom PEs to create a network with balanced performance and area metrics.

Neighbor weight refers to an option within the PE network compiler that controls whether the equation partitioning favors size or performance. This option is a sliding scale that can be set from 1 (favor size) to 10 (favor performance).

Area constraints detail the available LUTs, DSPs, and BRAMs on the target platform. The PE compiler will not allocate more than the available resources. The area constraint may have a very large area or performance impact on sufficiently complex models. An area constraint which is too small for the given model will not allow enough PEs to be allocated and reduce the possible speedup. Small models are not affected by the area constraint.

Solver type selects the iterative step solver to use. The currently supported solvers are Euler and Runge-Kutta4. Euler is the most simple solver, but can be inaccurate or diverge with medium to large time steps. The Runge-Kutta solver type is much more accurate, but may require up to 4X more computation time than the Euler solver.

Step size determines the amount of time between iterative solutions of the model equations. Decreasing the step size requires more computations per second, which reduces the performance of the model, but allows the solvers to be more accurate.

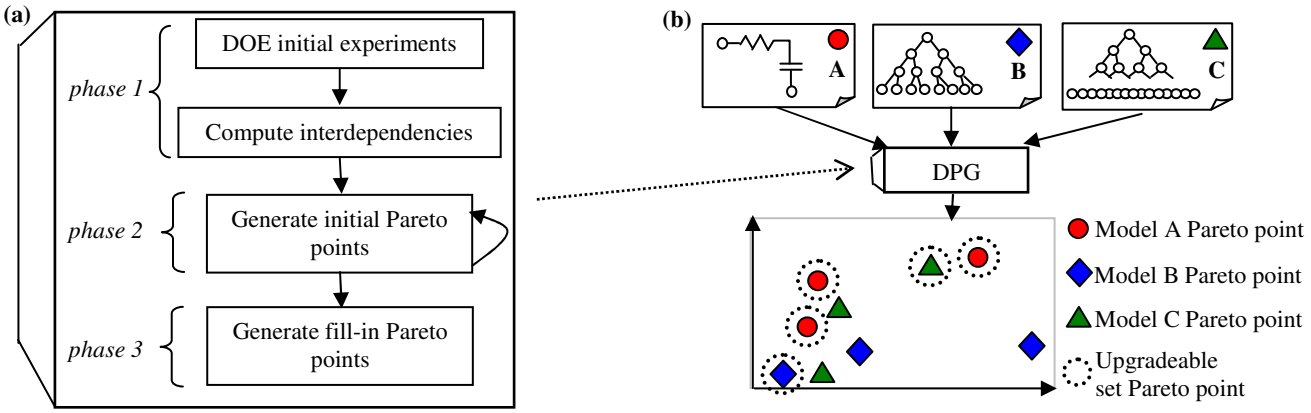

Figure 4: (a) The DPG algorithm flow. (b) Finding the Pareto points for a set of upgradeable models by applying DPG.

5. METHODOLOGY AND FRAMEWORK

To explore the space of PE network solutions for a set of upgradeable physical models, we have developed a visual web-based frontend coupled with a design-of-experiments statistical approach to identifying Pareto points that span an upgradeable set of physical models. In the following section we describe briefly what DOE is, and how our tool utilizes DOE.

5.1 DOE-based exploration of PE networks

Design-of-experiments is a statistical technique that identifies a minimal set of experiments that provide maximal cover of the possible solution space. Originally, DOE was developed for use in agriculture, but has since been developed into a powerful statistical technique used in many fields. DOE automatically identifies each parameter's magnitude of influence on the solution, since the differences between physical models (complexity, connectivity of equations, etc.) can impact how much a specific parameter like PE network type or neighbor weight matters. For example, a model with few equations will not be sensitive to area constraints.

5.1.1 *The DPG algorithm*

The DOE-based Pareto-point Generation (DPG) algorithm [12] can be used to apply DOE to and identify PE network solution Pareto points. By applying DPG to the upgradeable models, such that the models themselves are a parameter to the algorithm, the Pareto points that span across the set can be easily located. A basic flow chart of DPG is given in Figure 4. DPG consists of three phases: running initial experiments to identify parameter interdependencies, generating initial Pareto points, and filling in gaps in the Pareto curve. DOE uses either two or three-level parameters. Since PE networks have some continuous parameters, such as step size, we always select the minimum and maximum and midpoint values for continuous parameters to ensure we cover the space well enough. Phase 3 fills of DPG fills in the gaps of the solution space left by this discretization.

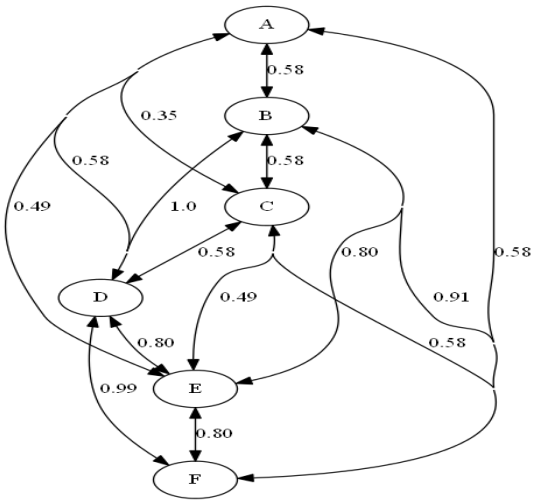

Figure 5: (a) Weighted parameter interdependency graph generated by DPG for a lung model.

Phase 1 of DPG runs an initial Plackett-Burman [8] set of experiments to automatically generate a *weighted parameter interdependency graph*. This graph details the relationship between parameters for each metric in a single description. DPG generates the graph by first estimating the solution metrics for every pair of parameters in the system, and then running the experiment. The amount of error between the estimated and actual value suggests the amount of interdependency between the parameters. Figure 5 shows the interdependency graph for the speedup metric for a RC lung, 6 gen. Weibel lung, and 9 gen. Weibel set of upgradeable models. Each node represents a parameter: *A* is the model from the set, *B* is the area constraint, *C* is the step size, *D* is the PE network type, *E* is the solver, and *F* is the neighbor weights. Higher edge weights represent higher levels of interdependency; for example, the 0.99 weight between *D* and *F* indicates that the effects of the PE type and neighbor weight options on the solution depend on one another, which is observable in by examining the changes in speedup due to different neighbor weights.

Phase 2 of DPG generates initial Pareto points from the parameter interdependency graph. The algorithm starts by evaluating the edge with the highest error weighting, and exhaustively searching the possible ranges of the two associated parameters. DOE uses either two or three level parameter values, so there is a maximum of nine possible configurations to run. The solutions of the search are pruned to only the local Pareto points, and the two parameter nodes of the graph are merged. This continues until only one node remains which contains a set of Pareto points for the entire design.

Phase 3 of DPG identifies regions which were not explored, due to the reduction of continuous parameters into a discrete three-level parameter. Parameters which are constant around the region are locked, and a local search within the region takes place. New Pareto points are added to the set identified in phase two.

5.2 Tool

The tool to explore the PE network solution space consists of a web page frontend and server DPG backend, implemented in

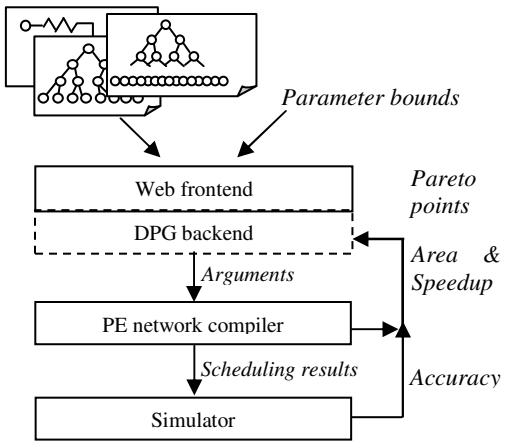

Figure 6: Architecture of the PE network exploration tool.

ASP.NET 4.0. A PE network compiler and simulator are implemented as .NET WCF web services. Figure 6 shows the architecture of the tool. The set of upgradeable models and parameter bounds are entered by the user. Parameter bounds include minimum and maximum area constraints, step sizes, etc. The DPG backend selects a set of experiments to run, and iteratively runs the compiler and simulator to generate area, speedup, and accuracy metrics. Pareto points are selected from the results by DPG and plotted visually on the web page.

5.2.1 PE network compiler and simulator

The PE network compiler accepts arguments generated by the DPG algorithm to partition the equations of a specific model across PEs. Once the equations are partitioned, each PE is scheduled. The partitioning and schedule information is enough to generate the area and speedup metrics. Area is reported by the compiler in terms of the number of LUTs, DSPs, and BRAM components used. The compiler automatically calculates the number of these components based on the type and frequency of each PE type. General PEs use one DSP and one BRAM each, while custom PEs may use arbitrary numbers of DSPs and typically a single BRAM. The final area metric is *equivalent LUTs* [5], which is a method for comparing resource usage for designs with various usage of logic cells and hard macros like DSPs. For a Xilinx Virtex6-240T, we use the following equation to calculate equivalent LUTs, where L_{EQ} is the equivalent LUTs, L is the number of LUTs, K_{DSP} is the equivalent LUTs per DSP (250), D is the number of DSPs, K_{BRAM} is the equivalent LUTs per BRAM (360), and B is the number of BRAMs:

$$L_{EQ} = L + K_{DSP} D + K_{BRAM} B$$

To calculate the speedup metric, the frequency of the resulting circuit must be estimated. The maximum frequency for a single PE is approximately 300 MHz when targeting a Virtex6, thus the maximum frequency that a larger PE network could achieve is also 300 MHz. As the number of PEs and connections between PEs grows larger, the place-and-route tools (Xilinx ISE 14.2) can not maintain the same timing due to congestion. We have created a regression model to estimate the frequency based on FPGA resource usage and the number of connections in the design:

$$Freq = K_0 - K_1 W - K_2 R_{DSP} + K_3 R_{BRAM} - K_4 R_{LUT}$$

Freq is the estimated frequency of the design, K_0, K_1, K_2, and K_3 are regression coefficients based on experimental data from PE networks targeting a Virtex6. W is the number of wires in the design (PE-to-PE connections), and R_{DSP}, R_{BRAM}, and R_{LUT} are resource usage ratios. This model is able to estimate frequencies to within 5% of their actual values, as shown in Figure 7.

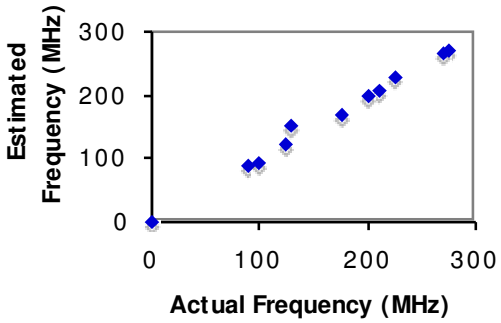

Figure 7: Regression model for estimating circuit frequency.

Once frequency has been estimated, total speedup is calculated:

$$Speedup = \frac{1}{\frac{1}{Freq} * C * S}$$

C is the number of cycles required to compute a iteration of the model. S is iterations per second, derived from the step size parameter. The factor 1/*Freq**C yields the amount of time to compute one iteration; multiplying by S yields the time to simulate 1 second. The inverse of the equation yields the speedup.

Accuracy is determined by simulating the PE network. A cycle-accurate simulator executes an iteration worth of PE instructions. The simulation is performed twice: once using the given solver and step size parameters, and once using a 'golden' set of parameters that consists of the most accurate configuration. For the golden parameters, we use RK4 solver and 0.01 ms step size.

After the simulations are complete, the time-series traces of each variable are compared. The simulator finds the variable in the user-defined simulation that is of a maximal distance from the golden standard simulation trace and returns the error. The error is then multiplied by the coefficient describing the model's relative accuracy in the upgradeable set, as described in section 4.

6. EXPERIMENT

We present an exploration of a set of upgradeable RC and Weibel models. We target a Xilinx Virtex6-240T FPGA, which consists of 150K LUTs, 716 DSPs, and 417 BRAMs. Table 1 enumerates the parameters and bounds that are input into the DPG algorithm for each set of models, since DOE uses 2 or 3-level parameters.

6.1 Lung models

The set of upgradeable lung models includes the RC, 6-generation and 9-generation Weibel models previously described. We use coefficients of 0.4, 0.9, and 1.0 to describe relative model accuracy, respectively. Figure 8 shows three plots comparing the area, speedup, and accuracy metrics of the 57 Pareto points found by the DPG algorithm. We consider the total design space size to be over 7,200 configurations, if the area constraint is discretized into just ten levels. Thus, the DPG algorithm prunes more than 99% of the design space, making exploration of the solutions more feasible. Filled circles indicate where groupings of Pareto points originate from the same model. For example, the left-most plot showing speedup vs. area has a group of Pareto points in the top-left corner that all are related to the RC model. Since the RC model is relatively simple, it has high speedup and low area

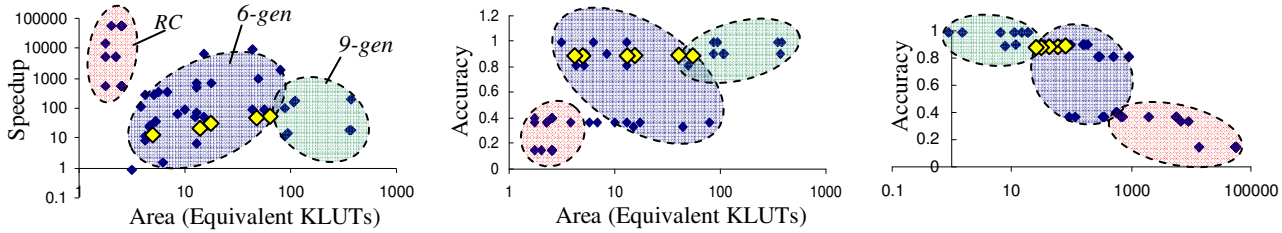

Figure 8: *3-dimensional Pareto plots projected onto 2-dimensions*. Yellow points show where accuracy parameters (model, step size, and solver type) are constant, emphasizing design space exploration of area and speedup without the accuracy metric. Considering upgradeable models during exploration tremendously expands the solution space.

Parameter	Low	Mid	High
LUTs	*10K*	*50K*	*150K*
DSPs	*20*	*200*	*716*
BRAMs	*20*	*200*	*417*
PE Type	*General*	*Custom*	*Hybrid*
Neighbor weight	*1*	*5*	*10*
Solver type	*Euler*	-	*RK4*
Step size (ms)	*0.01*	*0.1*	*1.0*

requirements; however, the other plots that include accuracy indicate the low fidelity of the solution.

The lighter points of Figure 8 illustrate Pareto points that correspond to configurations using an RK4 solver, 0.01 ms step size, and the 9-generation model. The points represent the 'normal' design space exploration of a PE network that has configurable type, number of PEs, etc. Considering only the middle case of the 6-generation Weibel model yields a solution space with speedups between 8X and 9200X, area between 4KLUTs and 55 KLUTs, and accuracy between 0.33 and 0.89. By considering the RC and 9-generation Weibel models during exploration, the solution space expands to speedups between 0.86X and 55000X, area between 1.7KLUTs and 376KLUTs, and accuracy between 0.15 and 0.99. Overall, the solution space that can be considered has increased in size by 6X in terms of speedup, 7.3X in terms of area, and 1.5X in terms of accuracy.

7. CONCLUSION

Physical models implemented on FPGAs provide large speedups over other implementation methods. Physical models introduce the feature of upgradeable models into design space exploration. We demonstrated how to include upgradeable models into a search approach and demonstrated improvements in the solution space of 5X on average of the area, speedup, and accuracy metrics. We utilized a design-of-experiments approach to enable rapid finding of Pareto points. Upgradeable models are not limited to physical models, but in fact may also apply to domains like signal processing and video processing where different algorithms can be considered that tradeoff quality with size and performance.

8. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation (CNS1016792, CPS1136146), the Semiconductor Research Corporation (GRC 2143.001), and a U.S. Department of Education GAANN fellowship. Special thanks also to David Sheldon for his help with the DPG algorithm and data analysis.

9. REFERENCES

[1] Givargis, T., and Vahid, F. 2002. Platune: a tuning framework for system-on-a-chip platforms. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 21.11, 1317-1327.

[2] Huang, C., Vahid, F., and Givargis, T. A Custom FPGA Processor for Physical Model Ordinary Differential Equation Solving. *IEEE Embed. Syst. Lett.* 3, 4, Dec. 2011, 113-116.

[3] Huang, C., Miller, B., Vahid, F., and Givargis, T. 2012. Synthesis of custom networks of heterogeneous processing elements for complex physical system emulation. In Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES+ISSS '12). ACM, 215-224.

[4] Kahng, A.B., Li, B., Peh, L., Samadi, K. 2009. ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration. Proceedings of the Conference on Design, Automation and Test in Europe (DATE'09).423-428.

[5] Meyer, J., Kocan, F. 2007. Sharing of SRAM Tables Among NPN-Equivalent LUTs in SRAM-Based FPGAs, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol.15, no.2, pp.182-195, Feb. 2007.

[6] Miller, B., Vahid, F., and Givargis, T. 2013. Embedding-based placement of processing element networks on FPGAs for physical model simulation. In Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays (FPGA '13). ACM, 181-190.

[7] Miller, J. A., Nair, R. S., Zhang, Z., Zhao, H. 1997. JSIM: A JAVA-based simulation and animation environment. Simulation Symposium,Proceedings. 30th Annual, pp. 31-42.

[8] Petersen, R. 1985. Design and Analysis of Experiments. Mercel Dekker Inc. New York, New York, 1985.

[9] de Pimentel, J. C. G., Y. G., Tirat-Gefen. "Hardware Acceleration for Real Time Simulation of Physiological Systems". Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE (pp. 218-223). IEEE.

[10] Potkonjak, M., Rabaey, J. "Algorithm Selection: A Quantitative Computation-intensive Optimization Approach". Computer-Aided Design, 1994., IEEE/ACM International Conference on , vol., no., pp.90-95, 6-10 1994.

[11] J. M. Rabaey, C. Chu, P. Hoang, and M. Potkonjak. 1991. Fast Prototyping of Datapath-Intensive Architectures. *IEEE Des. Test* 8, 2 (April 1991), 40-51.

[12] Sheldon, D., Vahid, F. "Making good points: application-specific pareto-point generation for design space exploration using statistical methods. International Symposium on Field Programmable Gate Arrays, 2009, pp. 123-132. ACM.

[13] Weibel, E. R. "Morphometry of the human lung". Anesthesiology vol. 26, no., pp., 1965.