# On the Asymptotic Costs of Multiplexer-based Reconfigurability

Johnathan York
The University of Texas at Austin
PO Box 8029 - F0252
Austin, TX 78713
jayork@mail.utexas.edu

Derek Chiou
The University of Texas at Austin
1 University Station C0803
Austin, TX 78712
derek@ece.utexas.edu

## ABSTRACT

Existing literature documents a number of techniques for combining a set of independent datapath designs into a single datapath that is run-time configurable to the functionality of any datapath in the set. This paper explores how delay, energy and area overhead attributable to reconfigurability scales with the number of configurable functionalities, independent of the design of specific datapaths. Distinct design space regions are identified based upon common scaling properties, with implications on the design and feasible efficiency bounds of reconfigurable devices.

## Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling Techniques

## General Terms

Design,Theory

## Keywords

Reconfigurable Logic, Datapath Merging

## 1. INTRODUCTION

Reconfigurable devices, such as Field Programmable Gate Arrays (FPGAs) and Digital Signal Processors (DSPs), are known to have substantial overhead compared to devices that cannot be reconfigured. Contemporary literature estimates that commercially-available FPGAs are 8-88X worse in area, 2-14X in delay, and 12-500X in power relative to even a standard-cell ASIC design[18]. Often worse are fetch-execute processors, which can be orders of magnitude less efficient in energy and delay than FPGA implementations[19, 24].

A middle ground that introduces flexibility into ASICs without incurring the full overhead of FPGA or processors would offer the ability to implement a limited set of applications at much higher efficiencies. Techniques to approach this middle ground have been developed for a number of application areas, under a variety of labels, including: the Datapath merging (DPM) problem, multi-mode synthesis, application-specific accelerator synthesis, Virtual Reconfigurable Architectures (VRAs), and configurable ASICs.

While extensive prior work exists on a middle ground between ASICs and general-purpose logic, most work is focused on specific applications, design problems, and/or optimization strategies. There is a literature gap at the highest levels of abstraction most useful for system-level architects. That is, there is little documented guidance on what exactly the overall design space might look like. High-level questions remain unaddressed, including: how do delay, energy, and area costs scale as the number of functionalies merged by DPM increase? How dependent are results on the specific topologies and similarities of circuits being merged? Which optimization strategies are most appropriate for a given DPM problem? This paper examines the general characteristics of the design space resulting from solving the DPM problem at a high level of abstraction, with particular attention to how overhead scales with the number of required datapath configurations.

## 2. BACKGROUND

As noted previously, techniques to approach a middle ground between inflexible ASICS and general-purpose programmable devices have been developed for a number of application areas, under a variety of labels. The Datapath Merging (DPM) subproblem of high-level synthesis accepts as input any number of DFGs, and produces as output a "single reconfigurable datapath", with the goal being to "design a reconfigurable datapath which incorporates all the [...] datapaths and has [the fewest] functional units and interconnections as possible" [22]. Experimental work on this subject has primarily focused on using unscheduled DFGs obtained from intermediate compiler representations of software implementations, although manual examples of the technique exist in the context of FPGA run-time reconfiguration [27]. The DPM problem has a substantial body of literature addressing algorithmic complexity [28], heuristics [2, 17], and optimization algorithms [23]. A simple example of DPM is illustrated graphically in Figure 1. Some DPM solutions may yield cyclic "false" timing paths, and these may be handled with techniques documented by Malik [21].

More recent work has applied a classic high-level synthesis argument that scheduling and binding are best jointly-optimized to the DPM problem [4, 3]. Specifically Chavet et al. [4] argue that among prior work, "four distinct ap-

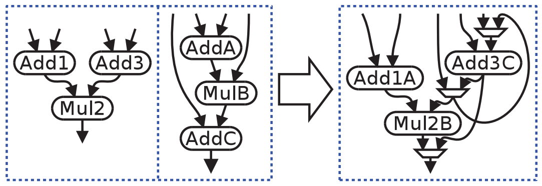

**Figure 1: An simple example of the Datapath Merging problem. Here two datapaths on the left are combined into a single datapath that is configurable with the functionality of either datapath. Note the introduction of additional wires and multiplexers to support configurability.**

proaches can be identified" based upon which of the steps in a conceptual high-level synthesis design are modified to be merge-aware. Chavet et al. further argue the need for a distinct "multi-mode" synthesis design flow that co-optimizes scheduling and binding, and suggest that new "scheduling, binding and register merging algorithms have to be proposed" [4]. Chiou, Bhunia and Roy [5] presented a multimode synthesis flow based upon a SPA-tially Chained Transformation (SPACT) in which the input DFGs are scheduled individually with estimated resource constraints, concatenated, bound, and then synthesized into HDL code.

The related application-specific accelerator synthesis problem [15, 1] has been studied to implement Application-Specific instruction set extensions for otherwise conventional microprocessors. For instance, Zuluaga and Topham [30] propose a technique that considers latency constraints during the merging process between multiple instruction set extensions. At a system level, Huang and Malik [17] discuss the DPM problem as a component of a methodology to minimize runtime reconfiguration overhead in Systems-on-a-Chip(SoC).

Addressing the problem from an angle applicable to existing FPGAs, Rullmann and Merker have developed a technique for development of virtual architectures on top of FPGAs using datapath merging [26]. In another paper, the datapath merging technique (including a novel Ant Colony Optimization algorithm) is used to generate placement constraints to force the FPGA synthesis tool to place similar logic in similar placement between multiple designs, thereby maximizing redundant configuration bits between DFGs[25].

The Totem Project at The University of Washington has the stated goal of providing an "automatic path for the creation of custom reconfigurable hardware, targeted for use in Systems-on-a-Chip (SoCs)" [16]. This ambitious project is intended to span from high-level architecture generation, through layout of the programmable chip, ultimately including CAD suites customized for each generated architecture.

Building upon the RaPiD framework [10], Compton and Hauck [8] developed a two stage algorithm for combining multiple RaPiD netlists into an application-specific RaPiD-like structure. This paper considered area optimization as a sole metric and demonstrated that custom architectures can achieve area efficiencies of only 1.5 times a lower bound based on the minimum number of functional units able to implement each of the input netlists. These concepts were further elaborated upon in Compton's Ph.D [7], which introduced the term "configurable ASIC" for the generated architectures. Among the contributions of the dissertation is a comparison of sample configurable ASIC designs against a traditional FPGA implementation. The comparisons with traditional FPGAs were limited to area-efficiency, but were quite favorable, with improvements ranging from 4-12X.

## 3. APPROACH

When examined relative to any of the input (i.e. fixed-function) datapaths, solutions of the DPM commonly rely upon the addition of multiplexers and connectivity (e.g. wires) within the datapath to introduce the required configurability. These added multiplexers and wires introduce overhead relative to the fixed-function datapath. While some of this overhead is incurred each time the device is reconfigured, the focus of this exercise is solely on overhead that is incurred post-configuration, during the operation of the datapath. That is, the focus is on overhead incurred while datapath remains a single configuration in exchange for the capability to reconfigure the datapath for other computations at a later time. To constrain the scope, we make several simplifying assumptions:

- that the configurable datapaths resulting from solving the DPM problem consist of opaque computing components present in the input datapath set, multiplexers inserted to allow configurability, and additional wires inserted to support the additional required connectivity,
- that the opaque computing components are homogenous (e.g. FPGA LUTs)[1],
- that the overhead of interest can be attributed to either multiplexer costs or the costs of the added wires,
- that no rescheduling is permissible (i.e. DPM is restricted to choosing a binding, with scheduling fixed by the datapath designs, as is common in RTL synthesis),
- that each of the input designs requires the same number of computing components,
- that the input designs have connectivity described by Rent's rule and have the same intrinsic Rent exponent, and
- that no subgraph isomorphisms are exploited in the DPM (i.e. a worst-case solution for typical DPM algorithms).

To provide insight on wiring costs independent of the peculiarities of any specific datapath, we adopt a parametrized model for wiring. The specific approach, based upon Rent's rule [20], is well known in the EDA community [6] and has proved useful in quantifying circuit characteristics in order to estimate features including wirelength distributions[14, 11] and average wirelengths [13, 29]. These results have in turn been used to estimate critical-path lengths, dynamic-power dissipation, and die areas [12]. Among the parameters used are:

- $C$ - the number of components in a circuit (or subset thereof),

[1]Preliminary work suggests that a similar analysis applies to datapaths with balanced ratios of heterogenous components.

- $p$ - the Rent exponent,
- $k$ - the average number of terminals per component,
- $\alpha$ - the fraction of terminals that are inputs
- $n_{cp}$ - the number of components in the critical path.

From these parameters, we assume wiring of the datapath designs to be merged are well-characterized by the analysis of Donath [13] and Davis et al [11]. That is, the expected total number of wires for a circuit $W$ is given by

$$W = \alpha k C(1 - C^{p-1}). \tag{1}$$

Moreover, the average wire length $\overline{R}$ is related to the *Rent exponent* $p$ and the number of circuit components $C$, and scales as follows

$$\overline{R} \sim \begin{cases} C^{p-0.5} & \text{for } p > 0.5 \\ \log C & \text{for } p = 0.5 \\ f(p) & \text{for } p < 0.5, \end{cases} \tag{2}$$

where $f(p)$ is an unspecified function of only $p$. Additionally, several parameters are defined specific to this effort:

- $N$ - the number of datapaths (i.e. functionalities) being merged.
- $\beta$ - the fraction of component input ports that need multiplexers inserted to maintain functional correctness. In the case of topological similarities, multiplexers may not be required in certain resource sharing manipulations. For a worst case bound, $\beta = 1$.
- $\gamma$ - the instance-dependent fraction of dynamic energy dissipated as a result of efficiencies gained from operand isolation [9] techniques. For a worst-case bound, $\gamma = 1$.

The assumptions above suggest a bookkeeping structure with three separate sources of costs in programmable designs: the computing components themselves, multiplexers inserted to allow reconfigurability, and the wires added to connecting those multiplexers for reconfigurability. To provide insight in a manner independent of any fabrication technology, we choose to normalize the multiplexer and wiring costs relative to cost of the computing components. Therefore, with regard to any particular cost metric (e.g. area, delay, energy), one can speak of a solution to a DPM problem instance falling somewhere on the 2D plane shown in Figure 2. Towards the top and right of the space, the cost from the multiplexers and wires added for configurability dominates the cost of the opaque computing components, respectively. Similarly, towards the bottom left, the cost of computing components is dominant over the costs incurred for reconfigurability.

Based upon this accounting, we decompose the design-space into three asymptotic regions based upon the dominant source of costs. The regions are referred to as asymptotic in that their properties hold true only so far as the associated source of costs dominates the others. This decomposition offers the advantage that within an asymptotic region, costs scale distinctly with $N$, the number of explicit reconfiguration options. Within this framework, the remainder of this paper addresses two key questions:

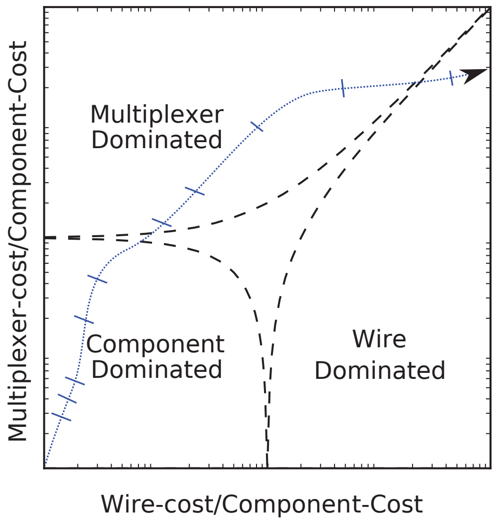

**Figure 2: Depiction of the solution space implied by multiplexer-based solutions to the datapath merging (DPM) problem. A single function datapath has no configurability overhead, and would be represented by a point on the extreme bottom left. As the number of desired functionalities increase ($N$), the configurability overhead grows and the point traverses towards the upper right. This is illustrated graphically by the arbitrarily drawn path.**

1. how do the delay, energy, and area overheads due to reconfigurability scale as the number of configurable functionalities ($N$) increase?
2. where do the boundaries between regions exist in terms of the number of functionalities($N$) (i.e. when does the overhead due to reconfigurability become dominant)?

## 4. SCALING COSTS

We now address the question of how costs scale with the number of functionalities within each of the asymptotic regions. We consider the cost metrics of critical path delay, energy per operation, and area independently, using "big-O" Bachmann-Landau asymptotic notation for conciseness.

By definition, within the component-dominated asymptotic region the cost of the opaque computing components dominate the costs of multiplexers and wires added to introduce configurability. As a result, within the component dominated-region, delay, energy, and area costs are independent of the number of functionalities, and therefore scale as $O(1)$.

Within the multiplexer-dominated region, the cost of the multiplexers inserted to support configurability dominates by definition, and thus overall costs scale as do the costs of the added multiplexers. In the worst-case, it is sufficient to solve the DPM problem by inserting a $N$-input multiplexer at the input of every opaque computing component. That is, the $N$ inputs of each added multiplexer are connected so as to provide the connectivity required for each of the $N$ functionalities. Assuming a worst-case recursive implementation of 2-input multiplexers, for each component this re-

| Asymptotic Region | Delay | Energy | Area |
|---|---|---|---|
| Component | $O(1)$ | $O(1)$ | $O(1)$ |
| Multiplexer | $O(log(N))$ | $O(N)$ | $O(N)$ |
| Wire | $O(N)$ | $O(N^{\frac{3}{2}})$ | Omitted |

**Table 1: Cost scaling with number of functionalities ($N$)**

sults in the addition of $O(N)$ multiplexers in a configuration $O(log(N))$ deep. Therefore the critical path delay added by these multiplexers scales as $O(log(N))$. Again assuming a worst-case implementation with no operand isolation or data gating, the energy dissipated by the multiplexers switching scales with the number of multiplexers, or $O(N)$. The area required also scales with $O(N)$

Within the wire-dominated region, the cost of wires added to support configurability dominate by definition and thus overall costs scale asymptotically as do the costs of the added wires. As noted above, in the worst-case adding $N$-input multiplexers for each input of each computing component is sufficient to solve the DPM problem. Each of these $N$ inputs requires a corresponding wire in a circuit to provide the needed connectivity. In the worst case, the overall data path area scales as $O(N)$ due to the added $O(N)$ multiplexers, such that wirelength distributions would tend to scale as $O(\sqrt{N})$. If one assumes these wires are unbuffered within the datapath, critical path delay will scale then as the square of the wirelength or as $O(N)$. As the number of wires ($W$) scales as $O(N)$, and the wirelength distribution scales as $O(\sqrt{N})$, the dynamic energy required to charge/discharge this wire network therefore scales as $O(N^{\frac{3}{2}})$. A similar analysis for area is omitted as a straight-forward analysis is complicated by the existence of distinct resources (e.g. metal layers) for wiring that are scalable somewhat independently of the area of active resources.

These scaling derivations are summarized in Table 1. A casual inspection suggests a number of important features of the design-space implied by the DPM problem:

1. Within the component dominated region, the marginal cost of adding new functionalities to a given design is asymptotically zero. The extent of this region is of key importance and is discussed in the following section.
2. In the multiplexer and wire dominated regions, delay scales much better than energy with added functionalities. This provides a simple explanation for reports of much higher energy overheads relative to delay overheads in general purpose programmable devices (e.g. FPGAs [18]).
3. Wire-dominated reconfigurable datapaths scale more poorly than do multiplexer dominated designs. This suggests that as wiring costs become more costly relative to switching (i.e. multiplexer) costs in process technologies, rich configurability will tend to become an even more expensive design option.

## 5. REGION BOUNDARIES

While the prior section outlined the scaling properties within each asymptotic region, it does not necessarily follow that any particular region has a non-trivial extent. We now address the existence and extent of these asymptotic regions. The component-dominated region contains the single-functionality limit case, and therefore contains at least one trivial design point. We now attempt to predict the extent of the component dominated region in terms of the number of desired functionalities ($N$) by identifying when the cost of multiplexers and wires dominate the cost of the opaque computing components.

We begin by defining the delay of the configurable DPM solution as:

$$\begin{aligned} delay(programmable) =& delay(fixed_{components}) \\ &+ n_{cp}ceil(log_2(N))delay(MUX2) \\ &+ delay(fixed_{wires})\frac{\overline{R}_{programmable}}{\overline{R}_{fixed}}, \end{aligned} \tag{3}$$

where $delay(programmable)$, $delay(fixed_{components})$, and $delay(MUX2)$ are the critical path delays of the programmable circuit, the fixed function components, and a 2 element multiplexer, respectively. The latter two terms of the sum correspond to the delay introduced by the multiplexers and the delay introduced by the wires added to support configurability.Similarly, define the dynamic energy of the configurable DPM solution as:

$$\begin{aligned} &energy(programmable) = \\ &energy(fixed_{components}) + W_{programmable}energy(MUX2)\gamma \\ &+energy(fixed_{wires})\gamma\frac{W_{programmable}}{W_{fixed}}\frac{\overline{R}_{programmable}}{\overline{R}_{fixed}}, \end{aligned} \tag{4}$$

where $energy(programmable)$, $energy(fixed_{components})$, $energy(MUX2)$ are the total dynamic energy dissipated per operation in the programmable circuit, the fixed function components, and a 2 element multiplexer, respectively.

By definition, the boundary between the component-dominated asymptotic region and the multiplexer-dominated region exists when the cost of the opaque computing component cost equals the cost of the added multiplexers. We can compute the extent of the component dominated region for delay by solving the equation

$$delay(fixed_{components}) > n_{cp}ceil(log_2(N))delay(MUX2) \tag{5}$$

for $N$. Using the assumptions of equations 1 and 2, it can be shown that the DPM design solution lies in the component-dominated asymptotic region when

$$ceil(log_2(N)) < \frac{delay(fixed_{components})}{n_{cp}delay(MUX2)}. \tag{6}$$

Restated in prose, the total delay of the computing components is greater than the costs of the added multiplexors, provided the number of functionalities is less than two raised to the power of the delay of the average computing component expressed in units of the delay of a 2-input multiplexer. Similar derivations can be computed for energy and for the boundary with the wire dominated region. The end results of these derivations are summarized in Table 2.

Inspection of Table 2 reveals a number of interesting observations:

| Boundary | Comp/Mux | Comp/Wire |
|---|---|---|
| Delay | $2^{\frac{delay(comp)}{n_{cp} delay(MUX2)}}$ | $\left(\frac{delay(comp)}{delay(wires)}\right)^2$ |
| Energy | $\frac{energy(comp)}{C \cdot energy(MUX2)}$ | $\left(\frac{energy(comp)}{energy(wires)}\right)^{\frac{2}{3}}$ |
| Area | $\frac{area(comp)}{C \cdot area(MUX2)}$ | Omitted |

**Table 2: Each table entry is the number of functionalities ($N$) at the region boundary specificied in the first row. Thus the second column shows where component and multiplexer costs are equal and the lower row shows ($N$) where the component and wire costs are equal. Here $n_{cp}$ is number of components on critical path, $C$ is number of components in the circuit, and MUX2 is a 2-input multiplexer.**

| Boundary | Comp/Mux | Comp/Wire |
|---|---|---|
| Delay | $2^{67}$ | 10 |
| Energy | 3 | 6 |
| Area | 5 | - |

**Table 3: Each table entry is the number of functionalities ($N$) at the region boundary specificied in the first row for an example 90nm technology node with 100 components in each functionality, and each component having the costs of a single 32-bit adder. This represents the worst-case bounds. For typical bounds, see Table 4.**

1. As the cost of the computing components increase (relative to a 2-input multiplexer), the component-dominated asymptotic region grows larger. That is, the coarser-grained the reconfigurability, the more functionalities can be introduced without configurability dominating costs.
2. Considering only multiplexer delays, the number of functionalities ($N$) within component dominated region grows exponentially large with the delay of the computing components.

For better intuition, if one assumes typical values from 90nm standard cell technology, a computing element is a single 32-bit adder, $C = 100$ components in each merged functionality, and that average wirelength on the critical path scales as $\sqrt{area(adder) \cdot C}$, we can compute numeric values for the entries in Table 2. The resulting worst-case ($\beta = \gamma = 1$) values are shown in Table 3.

To estimate more typical values rather than worst case, we have conducted DPM experiments on a set of Digital Signal Processing cores taken from a software defined radio (SDR) platform, including 1) coordinate rotation digital computer (CORDIC) sine/cosine generator, 2) complex-value heterodyne stage (hetero), 3) Cascaded Integrating Comb (CIC) decimating filter, 4) Finite Impulse Response (FIR) filter, and 5) a Fast Fourier Transform (FFT) butterfly. To avoid overly optimal effects from topological similarities, we used a random (i.e. unoptimized) binding in the DPM process, and estimated values for $\alpha$, $\beta$, $\gamma$, $k$, $C$. We found that

- only 51 of 202 possible (bus) multiplexers were inserted ($\beta = 0.25$),
- there was a 39% energy reduction via naive operand isolation ($\gamma = 0.61$),

| Boundary | Comp/Mux | Comp/Wire |
|---|---|---|
| Delay | $2^{80}$ | 208 |
| Energy | 40 | 20 |
| Area | 38 | - |

**Table 4: Each table entry is the number of functionalities ($N$) at the region boundary specified in the row header for an example 90nm technology node based on a case study conducted with Software Defined Radio datapaths.**

- designs were dominated by 2-input, 3-terminal components ($k = 3$,$\alpha = 0.67$),
- there were an average of 27 components per functionality ($C = 27$), and
- computing component costs are dominated by 16-bit multipliers, driving the $\frac{delay(fixed_{components})}{n_{cp}}$ term.

Using these parameter values, we find that the expected values for the extent of the component dominated region as are shown in Table 4. Note that even in the worst case, 20 functionalities can be merged before the costs of configurability begin to dominate the costs of the computing elements. Therefore the component-dominated region is perhaps usefully large, even without exploiting topological similarities between merged datapaths, as is commonly assumed for DPM.

## 6. CONCLUSIONS

We have shown that by careful construction of the problem, it is possible to predict characteristics of the design space implied by the DPM problem independent of the specific datapath topologies being merged. Moreover, we have established three asymptotic regions based upon the dominant cost component that form a framework convenient for performing analysis early in the design process. This decomposition has impacts on the further study of DPM optimization algorithms. Notably, any optimization strategy should focus on the dominant cost. Current DPM optimization has primarily focused on minimizing multiplexer insertion, which is reasonable for designs in the multiplexer-dominated region. However, such optimization may be misguided for designs known to reside in the other asymptotic regions. For instance, DPM binding algorithms reminiscent of recursive partitioning placement algorithms may be more suitable in the wire dominated region. This paper lays the ground work for system-level prediction such that a designer might predict a target region early in the design process.

We have further shown both bounds and typical values for the extent of these asymptotic regions. Notably, in a conservative analysis with datapath designs from an SDR application, we predict that tens of functionalities can be merged before wiring or multiplexer costs begin to dominate the cost of the computing components. This would suggest that, with suitable design tools, a useful degree of reconfigurability can be introduced into fixed-function designs without the cost of the reconfigurability becoming substantial.

## 7. REFERENCES

[1] K. Atasu, C. Ozturan, G. Dundar, O. Mencer, and W. Luk. CHIPS: Custom hardware instruction

processor synthesis. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 27(3):528, 2008.

[2] R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. Technical report, Algorithmica, 2001.

[3] L. Bertrand and E. Casseau. Automated multimode system design for high performance DSP applications. In *Proceedings of the 17th European Signal Processing Conference (EUSIPCO 2009)*, pages 1289–1293, 2009.

[4] C. Chavet, C. Andriamisaina, P. Coussy, E. Casseau, E. Juin, P. Urard, and E. Martin. A design flow dedicated to multi-mode architectures for DSP applications. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, pages 604–611. IEEE Press, 2007.

[5] L.-y. Chiou, S. Bhunia, and K. Roy. Synthesis of application-specific highly efficient multi-mode cores for embedded systems. *ACM Trans. Embed. Comput. Syst.*, 4(1):168–188, 2005.

[6] P. Christie and D. Stroobandt. The interpretation and application of Rent's rule. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(6):639 –648, Dec 2000.

[7] K. Compton. *Architecture Generation of Customized Reconfigurable Hardware*. PhD thesis, Northwestern University, 2003.

[8] K. Compton and S. Hauck. Totem: Custom reconfigurable array generation. *IEEE Symposium on FPGAs for Custom Computing Machines*, 2001.

[9] A. Correale, Jr. Overview of the power minimization techniques employed in the ibm powerpc 4xx embedded controllers. ISLPED '95, pages 75–80, 1995.

[10] D. Cronquist, C. Fisher, M. Figueroa, P. Franklin, and C. Ebeling. Architecture design of reconfigurable pipelined datapaths. *20th Anniversary Conference on Advanced Research in VLSI, 1999.*, pages 23–40, 1999.

[11] J. Davis, V. De, and J. Meindl. A stochastic wire-length distribution for gigascale integration (GSI)-Part I: Derivation and validation. *IEEE Transactions on Electron Devices*, 45(3), 1998.

[12] J. Davis, V. De, and J. Meindl. A stochastic wire-length distribution for gigascale integration (GSI)-Part II: Applications to clock frequency, power dissipation, and chip size estimation. *IEEE Transactions on Electron Devices*, 45(3), 1998.

[13] W. Donath. Placement and average interconnection lengths of computer logic. *Circuits and Systems, IEEE Transactions on*, 26(4):272–277, Apr 1979.

[14] W. Donath. Wire length distribution for placements of computer logic. *IBM Journal of Research and Development*, 25(2-3):152–155, 1981.

[15] W. Geurts, F. Catthoor, S. Vernalde, and H. De Man. *Accelerator Data-Path Synthesis for High-Throughput Signal Processing Applications*. Kluwer Academic Pub, 1997.

[16] S. Hauck, K. Compton, K. Eguro, M. Holland, S. Phillips, and A. Sharma. Totem: Domain-Specific Reconfigurable Logic. *submitted to IEEE Transactions on VLSI*, 2008.

[17] Z. Huang and S. Malik. Managing dynamic reconfiguration overhead in systems-on-a-chip design using reconfigurable datapaths and optimized interconnection networks. *Design, Automation and Test in Europe Conference*, 0:0735, 2001.

[18] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. In *FPGA '06: Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, pages 21–30, New York, NY, USA, 2006. ACM Press.

[19] P. Kwan and C. T. Clarke. FPGAs for improved energy efficiency in processor based systems. *Advances in Computer Systems Architecture: 10th Asia-Pacific Conference, ACSAC 2005, Singapore, October 24-26, 2005: Proceedings*, 2005.

[20] B. Landman and R. Russo. On a pin versus block relationship for partitions of logic graphs. *IEEE Transactions on Computers*, C-20:1469–1479, December 1971.

[21] S. Malik. Analysis of cyclic combinational circuits. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 618 –625, Nov 1993.

[22] N. Moreano, G. Araujo, Z. Huang, and S. Malik. Datapath merging and interconnection sharing for reconfigurable architectures. In *ISSS '02: Proceedings of the 15th international symposium on System Synthesis*, pages 38–43, 2002.

[23] N. Moreano, E. Borin, C. D. Souza, and G. Araujo. Efficient datapath merging for partially reconfigurable architectures. In *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, pages 969–980, 2005.

[24] K. Parnell and R. Bryner. Comparing and contrasting FPGA and microprocessor system design and development. Technical report, Xilinx, 2004.

[25] M. Rullmann and R. Merker. Maximum edge matching for reconfigurable computing. In *Reconfigurable Architectures Workshop at 13th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006), Rhodes, Greece*. Citeseer, 2006.

[26] M. Rullmann, R. Merker, H. Hinkelmann, P. Zipf, and M. Glesner. An Integrated Tool Flow to Realize Runtime-Reconfigurable Applications on a New Class of Partial Multi-Context FPGAs. In *Proc. 19th Intl. Conf. on Field Programmable Logic and Appls.*, 2009.

[27] N. Shirazi, W. Luk, and P. Cheung. Automating production of run-time reconfigurable designs. *Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 0:147, 1998.

[28] C. C. d. Souza, A. M. Lima, G. Araujo, and N. B. Moreano. The datapath merging problem in reconfigurable systems: Complexity, dual bounds and heuristic evaluation. *J. Exp. Algorithmics*, 2005.

[29] D. Stroobandt. Improving Donath's technique for estimating the average interconnection length in computer logic. *ELIS Technical Report*, 1996.

[30] M. Zuluaga and N. Topham. Resource sharing in custom instruction set extensions. In *Proceedings of the 6th IEEE Symposium on Application Specific Processors.(Jun. 2008)*, 2008.