University of Pennsylvania
Department of Electrical and System Engineering
Computer Organization

ESE680-002, Spring 2007 Final Exercise Wednesday, April 4

Due: Friday, May 4, 5:00PM

1 Setup

You are a recent hire at Variable Bitwidth Design (VBD). VBD has been building customized programmable components in the (c,w)-design space discussed in ESE680-002 (Day 9, Chapters 8 and 9 of [1]). To date, they have a product catalog of 100 different components corresponding to power of two c and w values from 1 to 1024.

With the rising costs of masks, it is getting difficult to maintain 100 different designs. Further, some customers complain that their applications actually have a mixed of throughput (Path Length, l_{path}) and SIMD datapath width (w_{app}) requirements and are poorly served by any of VBD's products.

Consequently, you have been asked to assess the opportunity for VBD to design a new Robust Arbitrary Bitwidth (RAB) array that might subsume all the of the existing designs.

You will recall from the Day 9 lecture, that if a single variable is matched in the design (*e.g.* $w_{arch} = w_{des}$ or $c = l_{path}$ it is possible to pick a robust design point for the other variable (c or w_{arch}, respectively) such that the architecture is over 50% efficient across all designs (all values of l_{path} or w_{arch}). So, you know it would be possible to reduce VBD's 100 design points to 10 designs that would be within a factor of 2 in area of the full 100 designs. However, that would not address the needs of customers with mixed design requirements. You will also recall that it was not clear how to pick such a robust design point across the entire (c,w) space.

Your boss suggests that the trick is to follow the robust design point allocation strategy, but find a way to make the instruction organization flexible. In particular, rather than hardwiring the width, w, provide configurable instruction distribution, allowing one bit operator to use its c *pinst*'s for itself, or combine its c pinsts with w' neighboring operators, such that they all execute the same pinst on each cycle, but together have an effective $c' = c \cdot w'$ pinsts. The RAB design should allow w' to take on any power of two between 1 and some architected maximum value (*e.g.* 1024). While discussing this over lunch, one colleague sketched out the configurable instruction distribution tree shown in Figure 1, and another sketched out the variant in Figure 2 and following.

The configuration of w' (the w''s) ultimately becomes an instruction distribution problem as well. Do we make this static configurable (at a different level than the cycle-by-cycle pinsts)? or do we broadcast w' configurations? Are all mixes of w''s sensible? As a starting

assumption, assume there is a single configuration for w''s, and they can differ across the array. The bitwidth configurations are loaded first, before the pinsts.

2 Assignment Goals

This assignment is intended to capture the major themes in the course, encourage you to integrate some of the pieces you've been studying as independent components, and give you an opportunity to spend some time working on a more open-ended problem in this domain. As such, the assignment is divided into three parts along this theme.

1. estimate implementation costs from a basic description
2. establish/quantify tradeoffs based on estimation
3. critique and optimize the design

This is deliberately an attempt to identify a "different" opportunity to explore (MATRIX [3] explored this theme, but not in as fine-grained a manner as suggested here). The starting design point you are given is based on a modest amount of thinking, but it may have some aspects which are nonsensical. The methodology in parts 1 and 2 should be clear from the course. For part 3, we [andre,nachiket] definitely don't know the answers; however, we believe there are rich things to spend some time thinking about using the intellectual tools from the course.

This is an individual assignment. Work alone. Consult the TA or instructor as you need clarification.

3 Design Point

The base VBD parameterized array is as follows:

- Compute operator is a 4-LUT with cascaded 5th input which runs down the columns of the array. The setup is similar to the cascaded 4-LUT in assignment 4. However, they use the parallel-prefix LUT cascade option discussed on Day 12 so that the gate delays in an n-bit cascade scale as $\log(n)$.
- Interconnect is a MoT interconnect (Day 18, [2]) with $p = 0.5$, $C = 8$, arity=2. The input selection portion of the C-box is densely encoded.
- Retiming is HSRA input style (Day 20, [4]) with a depth of 4. The retiming depth selection is densely encoded.

In the original VBD design, when an array is built for a particular w, w adjacent elements along a column are grouped together; an instruction bus distributes the same pinst to all w bit operations from a single instruction store of depth c.

VBD builds subarrays up to size 1024×1024, then tiles multiple subarrays onto a chip. Each subarray has its own controller (*e.g.* FSM) which feeds array-wide addresses into the subarray. Cascades are never longer than 1024.

For your design, the suggestion is to keep SIMD grouping along columns as in the original design. However, instruction storage will be distributed with the bit operators so it can be used individually in the $w' = 1$ case.

4 Technology Parameters

A_{bit}	SRAM bit for Instruction Store	1,000 λ^2
A_{retime}	one retiming register with mux	10,000 λ^2
A_{cmtx}	Compose Mux Transform for cascade	12,000 λ^2
$A_{2:1}$	2:1 mux	5,000 λ^2
$A_{16:1}$	16:1 mux	20,000 λ^2
A_{switch}	bidirectional switch (no configuration mem)	2,000 λ^2
A_{or2}	2-input OR gate	2,000 λ^2
A_{and2}	2-input AND gate	2,000 λ^2
A_{inv}	inverter	1,000 λ^2

5 Questions

As always, state necessary assumptions. Be clear about the design you are evaluting. There are certainly multiple interpretations of the sketch given in Section 3.

1. Build area models for the base design. [**30pts total**]

   (a) [**10**] Write an equation for the area of the bit operator in the base design excluding instruction memory. This should use symbolic constants for technology parameters and design parameters (this will be the same for any (c,w) design point). Definitely be clear about the design assumptions you are making in order to write this equation.

   (b) [**2**] Using the design and technology parameters given above, calculate the area of the bit operator above.

   (c) [**5**] Identify the pinst for the bit operator (again, same for all design points); give a table showing logic fields and bits.

   (d) [**1**] Identify the area required to store one pinst.

   (e) [**5**] Identify the 10 robust design points for the original VBD design (*i.e.* give robust point for $w_{des} = w_{app} = \{1, 2, 4, ...1024\}$.

(f) **[5]** Write an equation for the area per bit operator for the proposed instruction distribution scheme, including the memory matching. This should use symbolic constants for technology parameters. Definitely be clear about the design assumptions you are making in order to write this equation; the figures in the assignment refer to multiple design points (and each of those is, potentially, incomplete).

(g) **[2]** Using the technology parameters given above, calculate the area per bit operator of the instruction-distribution scheme.

2. Identify the costs and benefits of the RAB design. [**30pts total**]

(a) **[6]** Develop an equation for the efficiency of the RAB design point across homogeneous application space. The reference design at each point will be the optimally matched (c,w) VBD design without flexible instruction distribution.

(b) **[4]** Plot this efficiency across the original VBD design space (powers of two 1 to 1024 in each c and w).

(c) Consider an application where 50% of the design requires $w_{app} = 1$, $L_{path} = 1$, and 50% requires $w_{app} = 64$, $L_{path} = 1024$.

i. **[6]** Develop an equation for the efficiency of an original, homogeneous VBD designs (in terms of c and w) versus an optimal, heterogeneous design (*i.e.* half bitops have $c = 1$, $w = 1$ to match half of application, other half have $c = 1024$, $w = 64$ to match other half).

ii. **[4]** Plot the efficiency across (c,w) space.

iii. **[2]** Identify and report the design point and efficiency achieved for the homogeneous design point which is most efficient for this mixed application.

iv. **[8]** Calculate the efficiency of the RAB design for this application. Reference remains the optimal heterogeneous design.

3. Critique and Optimize the design. [**40pts total**]
Pick one of the open-ended design questions below and analyze. Quantitative analysis with equations, areas, and perhaps delays is encouraged.

- Propose and analyze a better (more efficient) scheme for flexible instruction distribution. Timing analysis, in addition to area analysis, will be a must.
- The tree instruction distribution scheme supports widths which are powers of two. Develop and analyze a scheme that supports arbitrary w'. Timing analysis, in addition to area analysis, will be a must.
- Given where the MoT bits land within the bit operators, what happens to the column MoT dimension in SIMD mode? What does this do to communication within a column? How can we redesign the column interconnect (or instruction coding sharing) to make this more reasonable?
- Can we mix different width configurations on a cycle-by-cycle basis? Explain architectural modification (or clarifications) necessary (if any) and usage rules. Give examples.

- Propose and analyze a better retiming design for the RAB design. Analysis here should consider robustness to various retiming profiles.
- Identify an application kernel that would exploit this flexibility and work out a mapping; report any lessons from the application mapping that might guide further improvements of the architecture.
- Identify something else that is problematic with the baseline architecture used in this flexible instruction distribution manner, and propose and analyze a better solution. [If you choose this: please run the idea by us no later than April 27th so we can give you feedback/guidance.]

You may answer more than one question, and the additional answers will be considered for extra credit. **However, answering one question in depth is better than giving mediocre answers to any number of questions**; so, do not use this option to give shallow answers to questions. The primary goal of question 3 is for you to put substantial thought and analysis into a focused technical issue.

References

[1] André DeHon. Reconfigurable Architectures for General-Purpose Computing. AI Technical Report 1586, MIT Artificial Intelligence Laboratory, 545 Technology Sq., Cambridge, MA 02139, October 1996.

[2] André DeHon and Raphael Rubin. Design of FPGA Interconnect for Multilevel Metalization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(10):1038–1050, October 2004.

[3] Ethan Mirsky and André DeHon. MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources. In *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, April 1996.

[4] William Tsu, Kip Macy, Atul Joshi, Randy Huang, Norman Walker, Tony Tung, Omid Rowhani, Varghese George, John Wawrzynek, and André DeHon. HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, pages 125–134, February 1999.

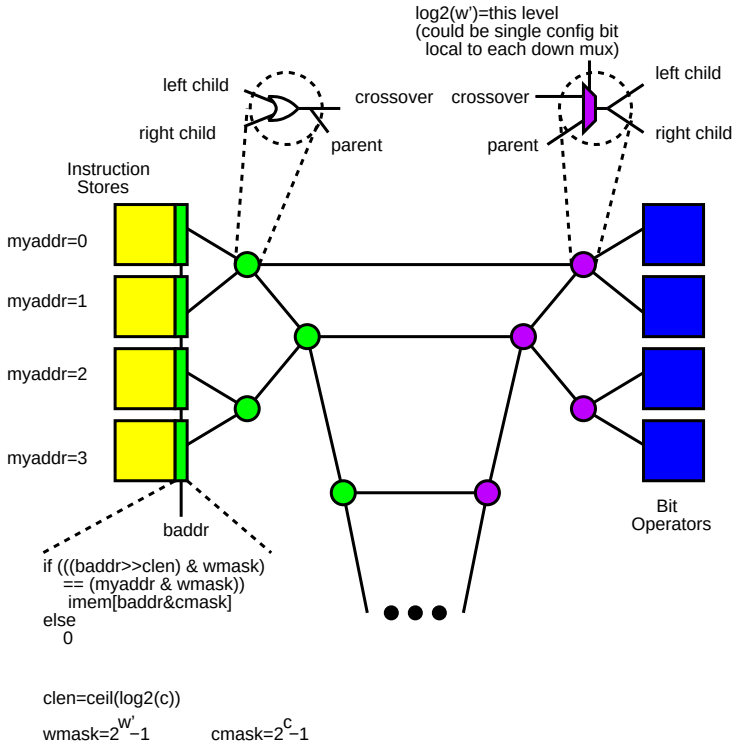

Figure 1: OR-based Configurable Instruction Distribution Tree

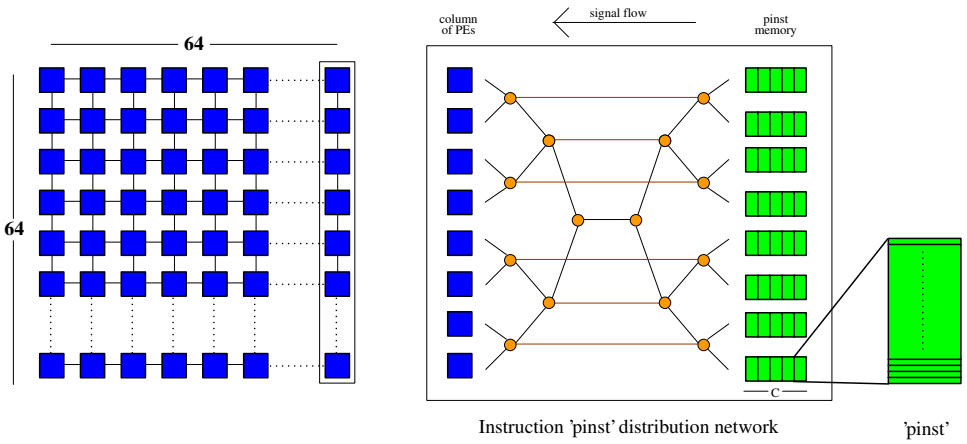

Figure 2: Mux-based Configurable Instruction Distribution Tree

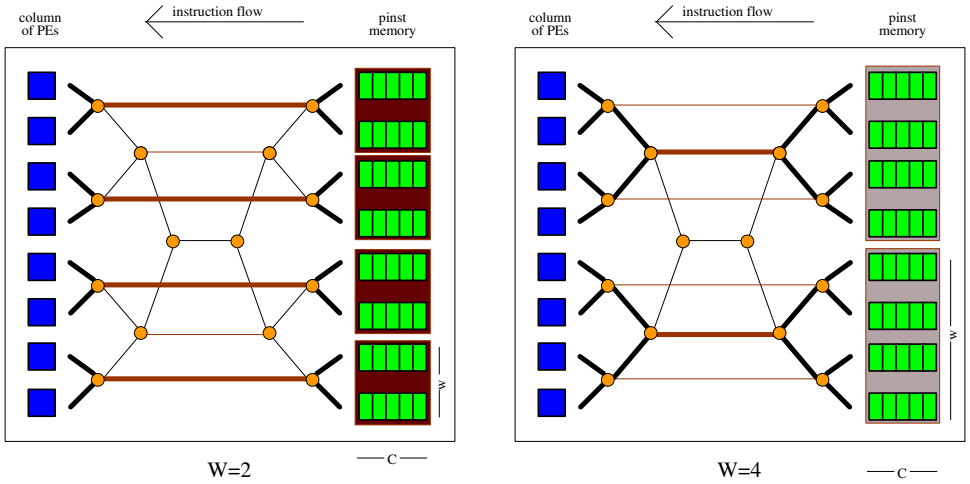

Figure 3: Example configuration for Mux-based Configurable Instruction Distribution Tree

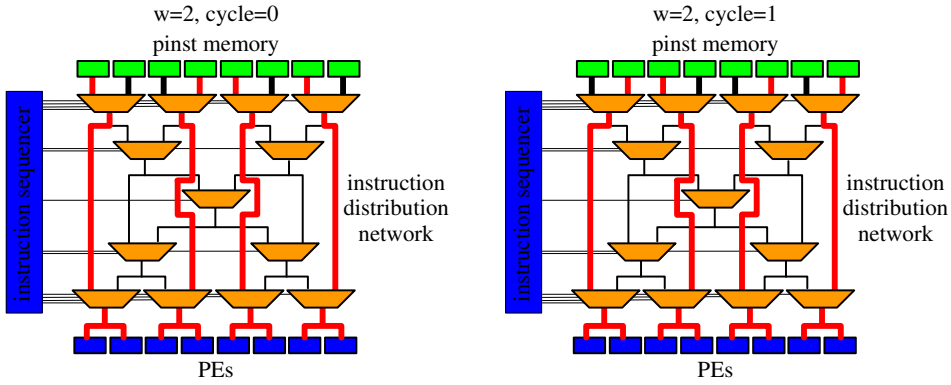

Figure 4: Cycle-by-cycle example of Mux-based Configurable Instruction Distribution Tree

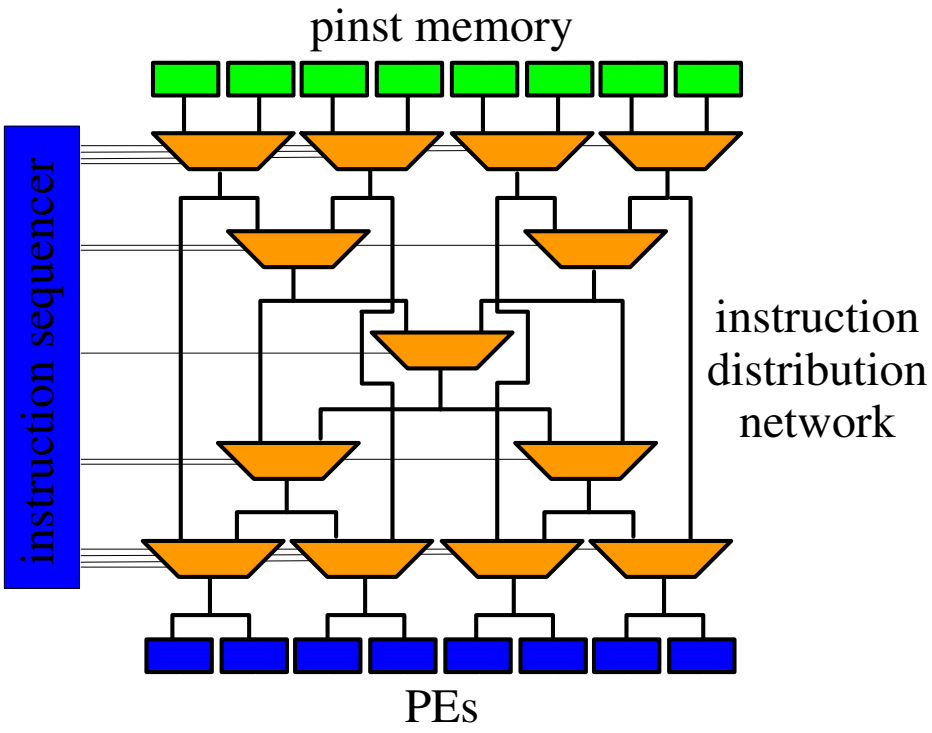

Figure 5: Control of Mux-based Configurable Instruction Distribution Tree