Design Considerations for Databus Charge Recovery

Benjamin Bishop Victor Lyuboslavsky N. Vijaykrishnan Mary Jane Irwin
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA
{bishop,lyubosla,vijay,mji}@cse.psu.edu

Abstract— **The charge recovery databus is a scheme which reduces energy consumption through the application of adiabatic circuit techniques. Previous work [2] gives a solid theoretical analysis of this scheme, including quantitative data assuming random bus values.**

We extend this earlier work by presenting a quantitative analysis of the charge recovery databus using 15 benchmarks and 4 high-level bus coding schemes. We show that a very simple implementation of the charge recovery databus is capable of reducing average energy consumption by 28% beyond traditional high-level bus encoding techniques. In addition, we examine delay and energy consumption in the added hardware.

I. Introduction

Charge recovery is a technique that can be used to decrease the total energy consumed on a databus. Since the current state and the desired next state of the databus are known, charge from the falling bitlines can be intelligently used to precharge the rising bitlines. This allows for a reduction in the energy required to drive the bus. Khoo et al. [2] report a theoretical energy savings of 47% and 59% for a 32 bit databus, using 1 and 4 charge transfers per bus access respectively. The theoretical maximum energy savings on a 32 bit databus is shown to be 72%. This work assumes random bus data.

In this paper, we extend previous work by considering a number of issues related to practical implementation of the charge recovery databus. We simulate the energy consumed by the charge recovery databus using bus data from a number of realistic benchmarks in conjunction with multiple bus coding techniques. Since we are concerned with practical implementation, where speed is likely to be important, we consider only the simple case of one charge transfer per bus access. In other words, we simply short every transitioning bus line prior to driving the bus.

II. Implementation

As described in [2], the single charge transfer requires only minimal control overhead. The sending hardware becomes slightly more complicated with the insertion of the charge transfer circuitry. Additionally, both the sender and receiver must tri-state the bus during the charge transfer phase.

Figure 1 shows the sending circuit for the single charge transfer 4 bit bus. The tri-state hardware is not included for simplicity.

Benjamin Bishop is now with the University of Georgia
Victor Lyuboslavsky is now with National Instruments

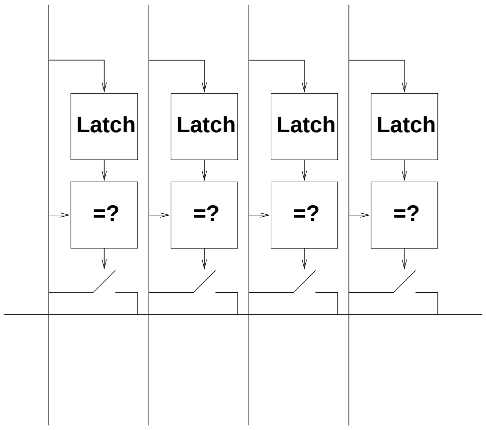

Fig. 1. Single charge transfer bus (4bit)

III. Simulation and Model

We can describe the energy consumption of the single transfer charge recovery databus as:

$$E = RCV\Delta V$$

where R is the number of rising bitlines, C is the bitline capacitance, V is the supply voltage, and ΔV is the voltage change on the bitlines after the charge transfer. For the purposes of simulation, it is convenient to express ΔV in terms of rising (R) and falling (F) bitlines. If we assume equal bitline capacitance, we have:

$$E = RCV\Big[V - V * \frac{F}{(R+F)}\Big]$$

$$E = CV^2\Big[\frac{R^2}{(R+F)}\Big]$$

For those interested in more information, [1] gives a thorough discussion about charging and discharging capacitance.

A. Benchmarks

A total of 15 benchmarks were used to generate bus data traces for our simulator. These benchmarks are a subset of the Mediabench benchmark set [3]. The benchmark subset was chosen[1] to simplify simulation. The traces were collected by porting Mediabench to the simplescalar architecture [6], and using a modified version of the simplescalar

[1] Benchmark selection occurred prior to simulation

simulator to dump memory access data traces. Due to the nature of the simplescalar architecture, only a 32 bit bus was considered. A short description of the benchmarks follows (for more information, see [3]):

- adpcm encode/decode: Adaptive differential pulse code modulation
- epic 1/2: Experimental image compression
- g721 encode/decode: CCITT based voice compression
- ghostscript: Postscript interpreter
- gsm encode/decode: European standard for speech transcoding
- jpeg encode: Lossy image compression
- mesa mip: OpenGL clone, fast texture mapping
- mesa os: OpenGL clone, standard rendering pipeline
- mesa tex: OpenGL clone, texture mapped Utah teapot
- mpeg decode: MPEG2 video compression
- pegwit: Elliptical curve public key encryption

B. Coding Techniques

We consider 4 coding techniques to reduce bit-switching on the charge recovery databus. Note that we are not modeling the energy consumed by encoding the data. A short description of these techniques follows (for more information, see [5]):

- 2's complement (2c): Standard 2's complement representation.
- Gray code (gc): Representations of X and X+1 differ by only one bit.
- Signed magnitude (sm): Data is represented as a sign bit and an n-1 bit unsigned data component.
- bus invert (bi): Extra bitline is introduced. It is set if it is less expensive (in terms of energy) to send the inverse of the bus data.

IV. Results

In this section, we discuss the energy consumption of the charge recovery bus in the simulation environment described above. Figures 2 and 3 show the energy consumption according to benchmark and bus coding strategy. An average energy savings of 28%, on gray code, signed magnitude and bus invert coding strategies, is shown at the rightmost column in figure 3. Please note that "regular" refers to a typical bus, and "enhanced" refers to the charge recovery bus.

It is interesting to consider how the bus data patterns affect the energy savings given by the charge recovery bus. Prior to passing data to the bus, it is helpful to balance the number of high-to-low transitions with the number of low-to-high transitions, in addition to minimizing the number of bit transitions. Table 1 illustrates this:

In this example, we consider the bus traces A (0001->1110->0001) and B (0011->1100->0011) on a 4 bit bus. Notice that the number of bit transitions in both traces is the same, but trace A has less balance in the number of low-to-high and high-to-low transitions. Table 1 shows the bitline voltages from which we can derive the energy consumed. We assume unit capacitance and a 2.5v supply. Step 1 is the initialization of the bus. In step 2, all bitlines

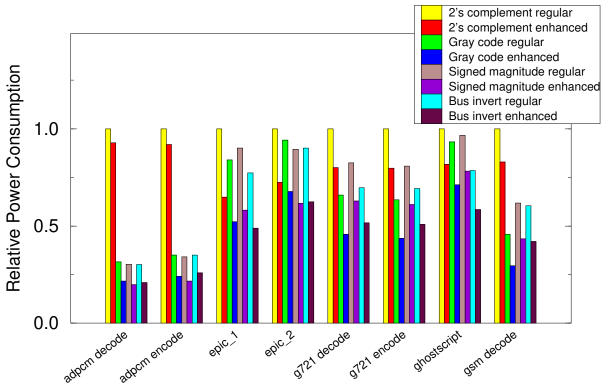

Fig. 2. Energy Consumption by Benchmark

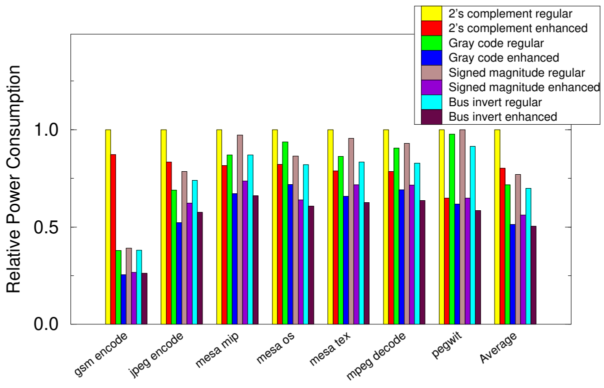

Fig. 3. Energy Consumption by Benchmark (continued)

are shorted since they are all making a transition. Using the first equation for energy given in section III, we find that the trace A consumes $3*1*2.5*(2.5-0.625) = 14.0625$ units of energy and that trace B consumes $2*1*2.5*(2.5-1.25) = 6.25$ units of energy in step 3. In step 4, all bitlines are again shorted. In step 5, trace A consumes $1*1*2.5*(2.5-1.875) = 1.5625$ units of energy while trace B consumes $2*1*2.5*(2.5-1.25) = 6.25$ units of energy. Not including initialization, trace A consumes a total of 15.625 units of energy and trace B consumes 12.5 units of energy.

In order to capture this notion of transition balance, we introduce a measure, which we call bias. It is defined as follows:

$$b = \Big[\sum_{all\ transitions} |R - F| \Big] * \frac{1}{total\ transitions}$$

Step #	A3	A2	A1	A0	B3	B2	B1	B0
1 (charge)	0.0	0.0	0.0	2.5	0.0	0.0	2.5	2.5
2 (short)	0.6	0.6	0.6	0.6	1.2	1.2	1.2	1.2
3 (charge)	2.5	2.5	2.5	0.0	2.5	2.5	0.0	0.0
4 (short)	1.9	1.9	1.9	1.9	1.2	1.2	1.2	1.2
5 (charge)	0.0	0.0	0.0	2.5	0.0	0.0	2.5	2.5

TABLE I
Charge recovery process (all units in Volts)

where R and F are the number of rising and falling bitlines on a given access, respectively. *total transitions* represents the total number of bit transitions over all bus accesses. Bias can be a useful measure for understanding energy reduction in the charge recovery databus. For example, Epic 1 (gray code) shows the largest reduction of energy consumption due to the charge recovery databus, this large reduction is due to the low bias of this benchmark. In a separate simulation over all possible transitions, we have found that b^{-1} is linearly related to the average energy savings of the charge recovery bus over a conventional bus. Figures 4 and 5 show bit transitions and bias respectively.

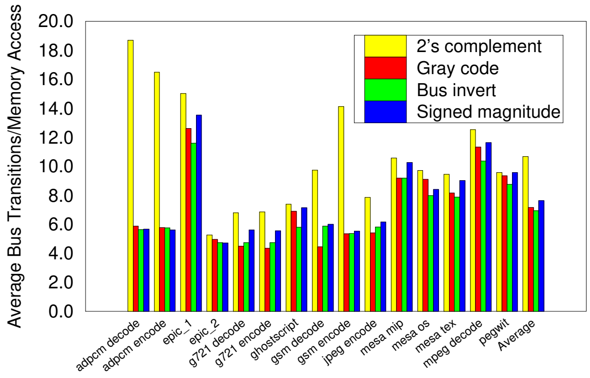

Fig. 4. Average bit transitions per memory access

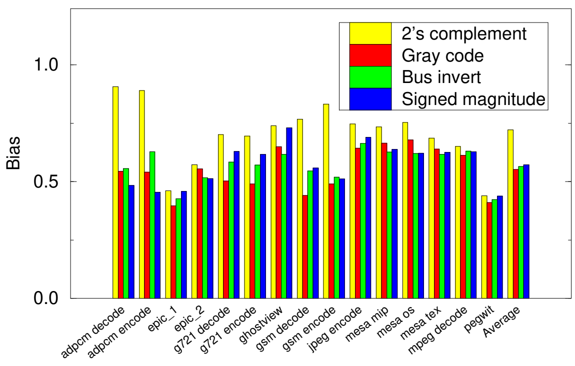

Fig. 5. Bias by benchmark

A. Circuit Simulation

To be of practical value, the charge recovery bus must have an efficient implementation. We present simulation results that predict the energy overhead of the charge recovery hardware as well as the additional delay introduced.

An 8-bit charge recovery bus and an 8-bit simple bus were simulated. A non-overlapping 3 phase clock was required for the charge recovery circuit. This clock can be derived locally from the global clock. Both circuits were designed with 0.25μm, 3.3V technology using the Micro Magic toolset [7]. The simulations were conducted using HSPICE on a set of input vectors based on the MediaBench benchmarks.

Table II shows the effects of varying the charge transfer time. For example, on the 20pF bus we can achieve 90% of the savings possible through charge recovery with a 10ns shorting time. It is clear from these results that charge recovery does require a significant amount of time. However, this technique is applicable to designs that can afford to trade speed for energy as well as buses that are not on the critical path.

Additionally, it was found that the charge transfer logic has an energy overhead of 3.6%, meaning that the charge recovery bus consumes 3.6% more energy than the simple bus for the 0pF case. This result demonstrates that the charge recovery hardware energy consumption is insignificant.

20pF bus	5ns	10ns
% Savings	57.6%	90.1%
10pF bus	3ns	6ns
% Savings	58.1%	94.1%
5pF bus	2ns	4ns
% Savings	61.0%	95.7%

TABLE II

Fraction of potential charge recovery savings for different shorting times

V. Conclusion

We present further information about energy savings through the use of a charge recovery databus. Our simulations use realistic benchmarks and bus coding techniques. Assuming one charge transfer per bus access for high speed and easy implementation, we show that an average energy savings of 28% is possible, although the benchmark with the greatest reduction saved 38%. These savings can be achieved in addition to those from bus coding and low-swing techniques [4]. The additional hardware was found to consume an insignificant amount of power. However, it does introduce significant delay.

References

[1] G. Yeap, *Practical Low Power VLSI Design*, Kluwer Academic Publishers, Boston, 1998.

[2] K. Khoo, A. Willson, "Charge Recovery on a Databus", *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 185-189, 1995.

[3] C. Lee, M. Potkonjak, W. Mangione-Smith, "Mediabench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems", *Proceedings of the 30th Annual International Symposium on Microarchitecture*, pp. 330-335, 1997.

[4] Hui Zhang, Jan Rabaey, "Low-Swing Interconnect Interface Circuits", *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 161-166, 1998.

[5] A. Chandrakasan, R. Broderson, *Low Power Digital CMOS Design*, pp. 222-234, Kluwer Academic Publishers, Boston, 1995.

[6] T. Austin, D. Burger, "The SimpleScalar Architectural Research Tool Set, Version 2.0", http://www.cs.wisc.edu/~mscalar/simplescalar.html

[7] Micro Magic, Inc., http://www.micromagic.com January 30, 2000.