RESEARCH ARTICLE OPEN ACCESS

Modelling Of Variation Trained Drowsy Cache

V. Padmaja*, D. Kishore **
*(Department of Electronics & Communication Engineering, SRI SAI ADITYA institute of science and technology, Surampalem)
** (Department of Electronics & Communication Engineering, SRI SAI ADITYA institute of science and technology, Surampalem)

ABSTRACT:
On-chip caches represent a sizable fraction of the total power consumption of microprocessors. Although large caches can significantly improve performance, they have the potential to increase power consumption. As feature sizes shrink, the dominant component of this power loss will be leakage. However, during a fixed period of time the activity in a cache is only centered on a small subset of the lines. This behavior can be exploited to cut the leakage power of large caches by putting the cold cache lines into a state preserving, low-power drowsy mode. Moving lines into and out of drowsy state incurs a slight performance loss. In this paper we investigate policies and circuit techniques for implementing drowsy caches. We show that with simple architectural techniques, about 80%-90% of the cache lines can be maintained in a drowsy state without affecting performance by more than 1%. According to our projections, in a 0.07um CMOS process, drowsy caches will be able to reduce the total energy (static and dynamic) consumed in the caches by 50%-75%. We also argue that the use of drowsy caches can simplify the design and control of low leakage caches, and avoid the need to completely turn off selected cache lines and lose their state.
Index Terms— Cache, drowsy cache, static random access memory (SRAM), AXI protocol

I. INTRODUCTION

In this paper, we propose a drowsy cache architecture that is aware of process variability within the structure. By using a simple and low cost distributed supply voltage management unit (one for each cache way) our pro-posed architecture allows a majority of the memory cells to operate at a reduced voltage

The gap between processor and memory speed continues to widen, cache performance becomes increasingly critical to the overall system performance. Researchers have observed that different program regions may have different access patterns and reuse types, and it is possible to fine -tune the cache management mechanism to exploit the reuse behaviors and achieve a better memory performance. cache schemes proposed previously, the cache selection and the bypass decision are made by the hardware.

To use cache mapping to improve program memory performance, we must have a way to estimate the dynamic memory access behavior of the program. Typical applications which run on the Strong ARM and the XScale processors perform multimedia tasks, encryption, and compression, among others

The gap between processor and memory speed continues to widen, cache performance becomes increasingly critical to the overall system performance. Researchers have observed that different program regions may have different access patterns and reuse types, and it is possible to fine -tune the cache management mechanism to exploit the reuse behaviors and achieve a better memory performance. cache schemes proposed previously, the cache selection and the bypass decision are made by the hardware.

To use cache mapping to improve program memory performance, we must have a way to estimate the dynamic memory access behaviour of the program. Typical applications which run on the Strong ARM and the XScale processors perform multimedia tasks, encryption, and compression, among others.

II. PROPOSED ARCHITECTURE

To counter the effect of process variations, typically designers overdrive the entire memory array. This leads to extra power consumption as both leakage and dynamic power increase. We are proposing to use a modified wordline driver peripheral circuit to allow selective wordline overdriving utilizing a small one step charge pump. The wordline peripheral circuit will drive the wordline in two phases. In the first phase, using the supplied Vdd the wordline is driven to Vdd. In the second phase the charge pump will overdrive the wordline voltage increasing the Vgs above the supply Vdd. Increase in Vgs improves both access and write time to the cell as will be described in the following sections DRV for cold lines which are determined by cache access pattern and are managed via the proposed architecture. Cache ways that are supplied with this voltage are referred to as “Cold Ways”. The remaining two voltage levels are used in cache ways located within the Cache Window of

Execution (CWoE) V_{dd}^{LOW} is supplied if cache way Could operate correctly in that voltage, otherwise cache way is supplied with V_{dd}^{HIGH}.

A 6T SRAM memory, however, can be manufactured in a standard logic process and continues to offer high-performance at a reasonable density and power dissipation. SRAMs will likely remain the dominant on-die memory technology, as SRAM cells have already been demonstrated down to the 32nm technology node. A 6T SRAM cell uses a pair of cross-coupled inverters as its bi-stable storage element with two additional NMOS devices for read and write access (Figure 1).

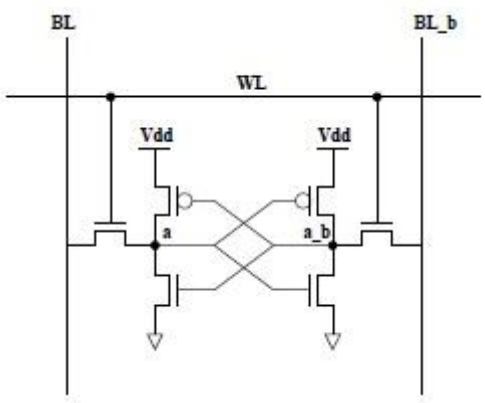

Fig 1. 6T SRAM cell

The cells are aggregated into cell arrays to share the decoding and I/O logic. On a read, the decoder raises the wordline (*WL*) of the desired word. The bitlines (*BL* and *BL b*) have been precharged to a reference voltage, and the cell drives a differential current onto the bitlines according to the stored value. The cell current is relatively weak for the bitline capacitance, so to speed the read operation, a sense amplifier in the I/O logic amplifies the bitline differential voltage to produce a full swing logic value.

Figure 2 & 3 shows how a read transaction uses the read address and read data channels and how a write transaction uses the write address, write data, and write response channels.

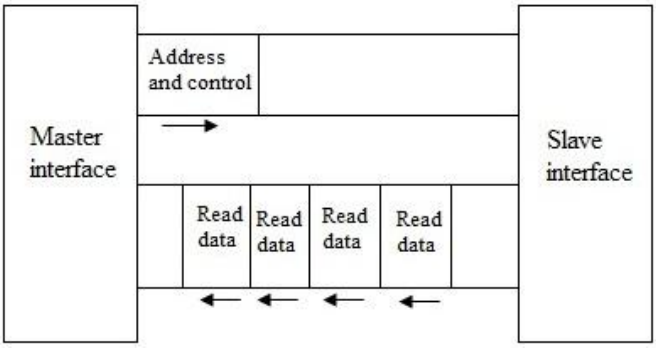

Figure 2. Channel architecture of reads

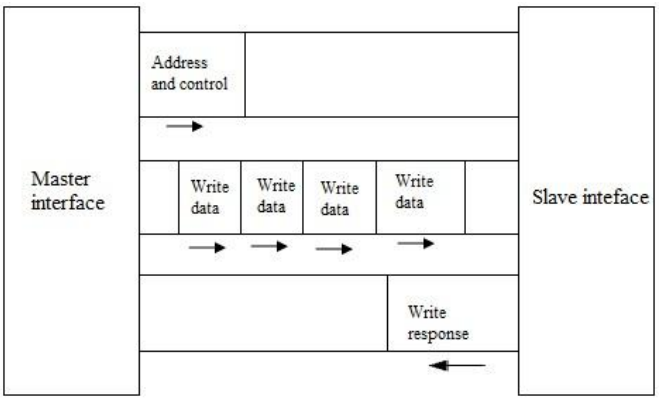

Figure 3. Channel architecture of writes

A. Channel definition*:*

Each of the five independent channels consists of a set of information signals and uses a two-way VALID and READY handshake mechanism. The information source uses the VALID signal to show when valid data or control information is available on the channel. The destination uses the READY signal to show when it can accept the data. Both the read data channel and the write data channel also include a LAST signal to indicate when the transfer of the final data item within a transaction takes place.

I. Read and write address channels:

Read and write transactions each have their own address channel. The appropriate address channel carries all of the required address and control information for a transaction.

The AXI protocol supports the following mechanisms:

- Variable-length bursts, from 1 to 16 data transfers per burst
- bursts with a transfer size of 8-1024 bits
- Wrapping, incrementing, and non-incrementing bursts
- Atomic operations, using exclusive or locked accesses
- System-level caching and buffering control
- Secure and privileged access.

1.Read data channel*:*

The read data channel conveys both the read data and any read response information from the slave back to the master. The read data channel includes:

- The data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide
- A read response indicating the completion status of the read transaction.

2.Write data channel:

The write data channel conveys the write data from the master to the slave and includes:

- The data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide

- One byte lane strobe for every eight data bits, indicating which bytes of the data bus are valid.

Write data channel information is always treated as buffered, so that the master can perform write transactions without slave acknowledgement of previous write transactions.

B. Basic Transactions:

This section gives examples of basic AXI protocol transactions. Each example shows the VALID and READY handshake mechanism. Transfer of either address information or data occurs when both the VALID and READY signals are HIGH. The examples are provided in:

1. *Read burst example:*

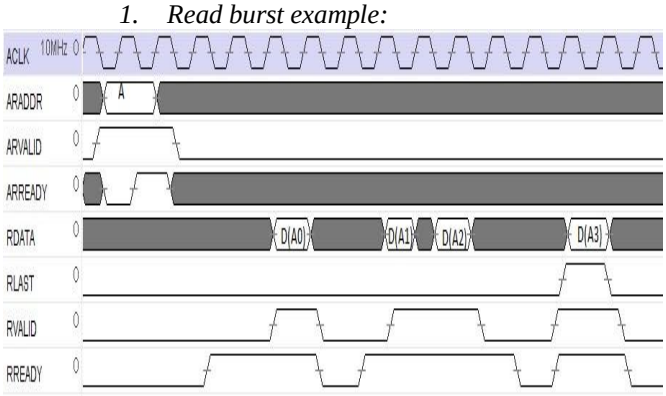

Figure 4. Read burst

Figure 4 shows a read burst of four transfers. In this example, the master drives the address, and the slave accepts it one cycle later.

The master also drives a set of control signals showing the length and type of the burst, but these signals are omitted from the figure for clarity. After the address appears on the address bus, the data transfer occurs on the read data channel. The slave keeps the VALID signal LOW until the read data is available. For the final data transfer of the burst, the slave asserts the RLAST signal to show that the last data item is being transferred.

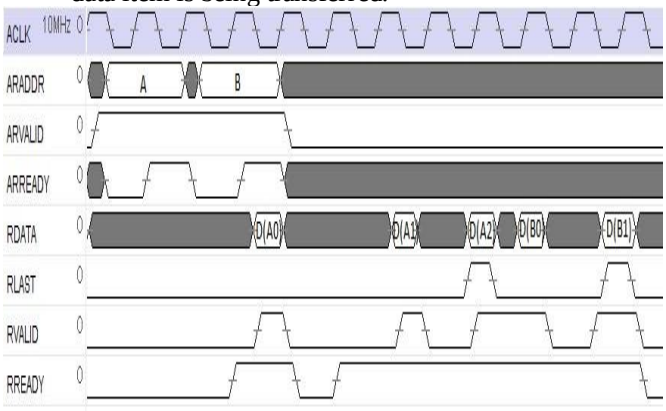

Figure 5. Overlapping read bursts

Figure 5 shows how a master can drive another burst address after the slave accepts the first address. This enables a slave to begin processing data for the second burst in parallel with the completion of the first burst.

1.Write burst example:

The process starts when the master sends an address and control information on the write address channel. The master then sends each item of write data over the write data channel. When the master sends the last data item, the WLAST signal goes HIGH. When the slave has accepted all the data items, it drives a write response back to the master to indicate that the write transaction is complete. As shown in fig 6.

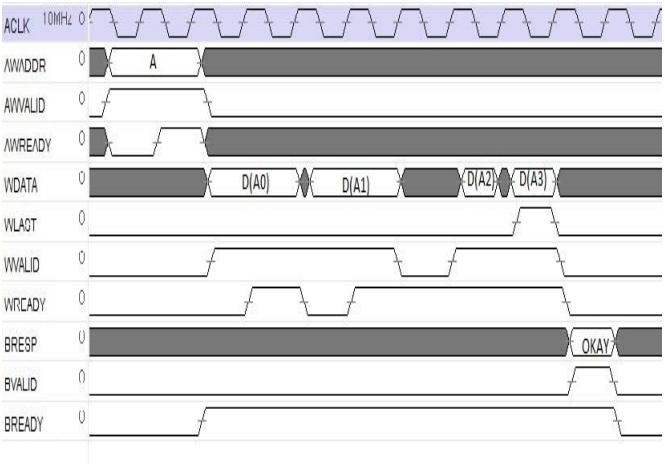

Figure 6. Write burst

III. AXI PROTOCOL

AXI is part of ARM AMBA, a family of micro controller buses first introduced in 1996. The first version of AXI was first included in AMBA 3.0, released in 2003. AMBA 4.0, released in 2010, includes the second version of AXI, AXI4.

There are three types of AXI4 interfaces:

- AXI4-for high-performance memory-mapped requirements.
- AXI4-Lite–for simple, low-throughput memory-mapped communication (for example, to and from control and status registers).
- AXI4-Stream—for high-speed streaming data.

C. AXI4:

The AXI4 protocol is an update to AXI3 which is designed to enhance the performance and utilization of the interconnect when used by multiple masters.

D. AXI4-Lite:

AXI4-Lite is a subset of the AXI4 protocol intended for communication with simpler, smaller control register-style interfaces in components. The AXI4-Lite IP Interface is a part of the Xilinx family of Advanced RISC Machine (ARM) Advanced Microcontroller Bus Architecture (AMBA) Advanced

extensible Interface (AXI) control interface compatible products. It provides a point-to-point bidirectional interface between a user Intellectual property (IP) core and the AXI interconnect. This version of the AXI4-Lite IP interface has been optimized for slave operation on the AXI. It does not provide support for Direct Memory Access (DMA) and IP Master Services.

Features:

- Supports 32-bit slave configuration.
- Supports read and write data transfers of 32-bit width.
- Supports multiple address ranges.
- Read has the higher priority over write.
- Read to the holes in the address space returns 0*00000000.
- Writes to the holes in the address space after the register map are ignored and responded with an OKAY response.
- Both AXI and IP Interconnect are little ending.

E. AXI4-Stream Protocol:

The AXI4-Stream protocol is used for applications that typically focus on a data-centric and data-flow paradigm where the concept of an address is not present or not required. Each AXI4-Stream acts as a single unidirectional channel for a handshake data flow. At this lower level of operation (compared to the memory mapped AXI protocol types), the mechanism to move data between IP is defined and efficient, but there is no unifying address context between IP. The AXI4-Stream IP can be better optimized for performance in data flow applications, but also tends to be more specialized around a given application space. The AXI4-Stream protocol is designed for unidirectional data transfers from master to slave with greatly reduced signal routing.

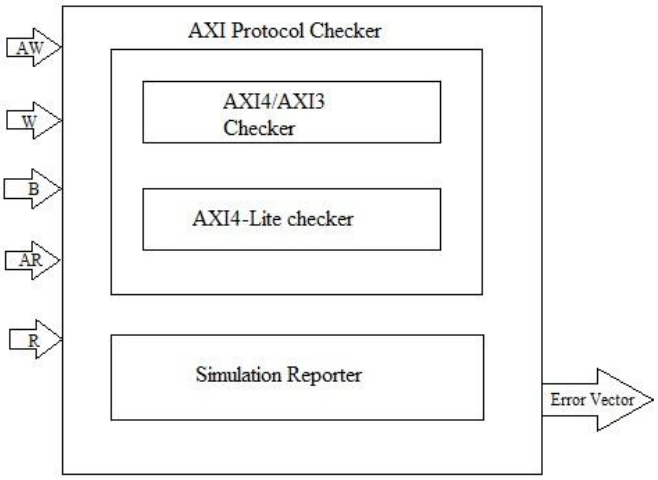

Fig 7. AXI Protocol CIRCUIT DIAGRAM

F. Additional features:

The AXI protocol also supports the following additional features:

1. **Burst types*:***

The AXI protocol supports three different burst types that are suitable for:

- normal memory accesses
- wrapping cache line bursts
- Streaming data to peripheral FIFO locations.

2. System cache support:

The cache-support signal of the AXI protocol enables a master to provide to a system-level cache the buffer able, cacheable, and allocate attributes of a transaction.

3. Protection unit support:

To enable both privileged and secure accesses, the AXI protocol provides three levels of protection unit support.

IV. RESULTS

Cache Enable:

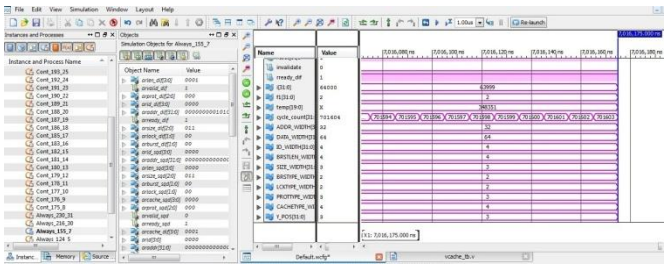

Fig 8. Cache Enable

Cache Disable:

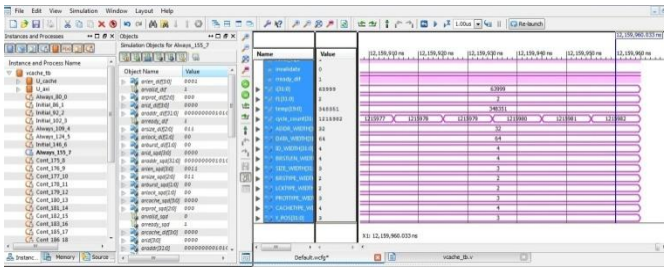

Fig 9. Cache Disable

V. CONCLUSION

In this paper we presented a novel architecture for low power and high yielding memory arrays. Proposed approach utilizes a charge pump wordline driver and selectively overdrives the wordlines containing weak cells. This architecture achieves power savings of more that 50% in dynamic, and 60% in leakage power as compared to the traditional architecture running at nominal voltage. Alternatively, when operated at the same voltage as traditional memory it provides an improvement in memory array yield. By using AXI protocol we are sending the data in cache memory. When the data is transferred in to the memory the enable and disable conditions is same.

REFERENCES

[1] A. Sasan, H. Homayoun, A. Eltawil, and F. J. Kurdahi, "Process variation aware SRAM/cache for aggressive voltage-

frequency scaling," in *Proc. DATE*, pp. 911–916.

[2] A. Sasan (Mohammad AMakhzan), A. Khajeh, A. Eltawil, and F. Kurdahi, "Limits of voltage scaling for caches utilizing fault tolerant techniques," in *Proc. ICCD*, pp. 488–495.

[3] M. A. Lucente, C. H. Harris, and R. M. Muir, "Memory system reliability improvement through associative cache redundancy," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1990, pp. 19.6/1–19.6/4.

[4] Z. Bo, D. Blaauw, D. Sylvester, and K. Flautner, "The limit of dynamic voltage scaling and insomniac dynamic voltage scaling," *IEEE Trans.Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 11, pp. 1239–1252,Nov. 2005.

[5] K. Flautner, K. Nam Sung, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: Simple techniques for reducing leakage power," in *Proc. 29th Annu. Int. Symp. Comput. Arch.* , 2002, pp. 148–157.

[6] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte, "Adaptive mode-control: A static-power-efficient cache design," in *Proc. Int. Conf. Parallel Arch. Compilation Techn.*, 2001, pp. 61–70.

[7] A. Agarwal, B. C. Paul, S. Mukho padhyay, and K. Roy, "Process variation in embedded memories: Failure analysis and variation aware architecture," *IEEE Trans. Solid-State Circuits*, vol. 40, no. 9, pp. 1804–1814, Sep. 2005.

VELUVALI PADMAJA was born on 2ND DECEMBER 1980 at Kakinada, India. She received her, B.Tech in Electronics & Communication Engineering from JNTU KAKINADA, AP, India and Pursuing M.Tech in VLSI at SRI SAI ADITYA institute of science and technology, Surampalem , Peddapuram AP, India.

D.KISHORE received his B. Tech in Electronics & Communication Engineering from JNTU KAKINADA IN 2004 and M. Tech degree in COMMUNICATION AND SIGNAL PROCESSING from BAPATLA ENGINEERING COLLEGE,GUNTUR, ACHARYA NAGARJUNA UNIVERSITY, in 2009 and also working as ASSOCIATE PROFESSOR in SRI SAI ADITYA institute of science and technology, Dept of Electronics & Communication Engineering, Surampalem, Peddapuram AP, India.