International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-5, June 2013

# Hardware Modeling of VTD- Cache for fine Grain Voltage Scaling

**Y.Ch Sekhar, P.S.Srinivas Babu**

***Abstract— This proposed title of this work allows us to design Variation Trained Drowsy Cache for significant saving of power consumption. When addressing reliability issues. The novel and modular architecture of the VTD – Cache and its associated controller makes it easy to implemented in memory compilers with a small area and power overhead. With proper selection of scaled voltage levels and hort training period the proposed architecture allows micro tuning of the cache and also this architecture variation of supply voltage settings. We model this scheme on FPGA core.***

***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.

In this design, the existence of a cell that is severely affected by process variation does not dictate a larger voltage to the entire cache but rather the higher volt-age requirement is only mandated to the cache way(s) that contains the affected cell(s). This allows a majority of the cells in the cache to operate at a lower supply voltage while addressing the reliability concern raised by the "weaker" cells by shifting their statistics to a more favorable operating region via higher supply voltage. The proposed cache architecture not only guarantees safe and correct operation at lower voltages within the window of benchmark execution in the cache.

## II. PROPOSED ARCHITECTURE

The VTD-Cache conceptually operates by allowing a fine grain control over the voltage of each cache way. Among available voltage levels the supplied voltage is chosen by a simple voltage selector that is implemented at each cache way. In VTD-Cache each cache way can be supplied from one of three possible voltages. The lowest voltage level supplies 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).

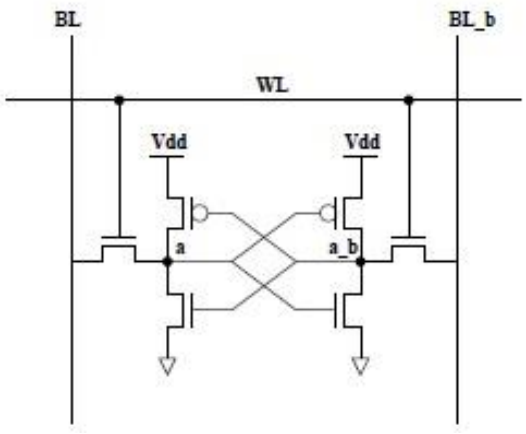


Fig 1. 6T SRAM cell

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.

On a write, the write driver in the I/O logic places the write data onto the bitlines as full-rail signals. The decoder again raises the wordline of the accessed word, and the cells store the new data values. A number of cell columns can share I/O circuitry via the column multiplexor. There are a number of ways in which designers have optimized the architecture and circuit design of SRAMs to improve the performance and energy efficiency. In this chapter, we will review a few of the most prevalent and significant optimizations.

The AXI protocol is burst-based. Every transaction has address and control information on the address channel that describes the nature of the data to be transferred.

**Manuscript published on 30 June 2013.**
* Correspondence Author (s)
**Y.Ch Sekhar**, Department of Electronics & Communication Engineering, K L University, Vijayawada, India.
**P. S. Srinivas Babu**, Department of Electronics & Communication Engineering, K L University, Vijayawada, India.

The data is transferred between master and slave using a write data channel to the slave or a read data channel to the master. In write transactions, in which all the data flows from the master to the slave, the AXI protocol has an additional write response channel to allow the slave to signal to the master the completion of the write transaction.
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.

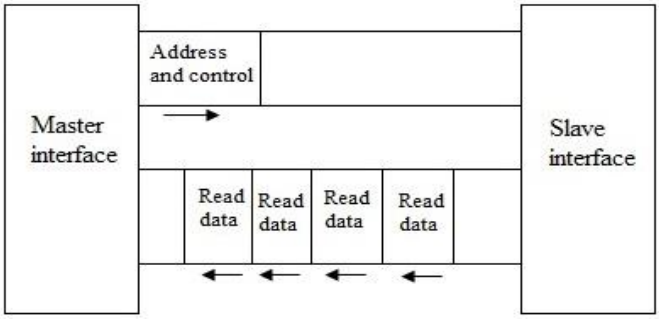


Figure 2. Channel architecture of reads

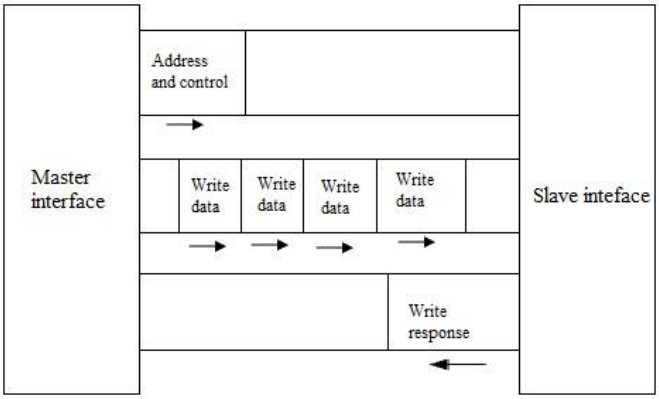


Figure 3. Channel architecture of writes

### 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.

#### *1. 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.

#### *2. 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.

#### *3. 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:*

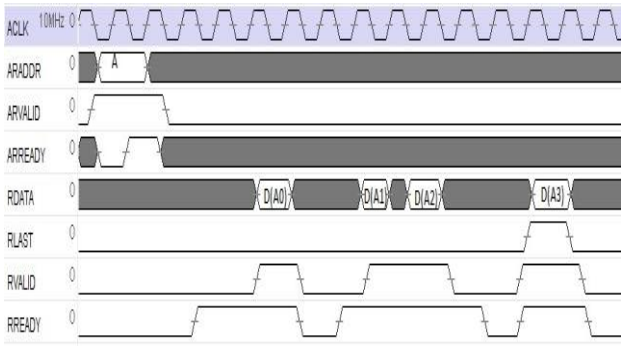


Figure 4. Read burst

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.

#### 2. *Overlapping read burst example :*


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

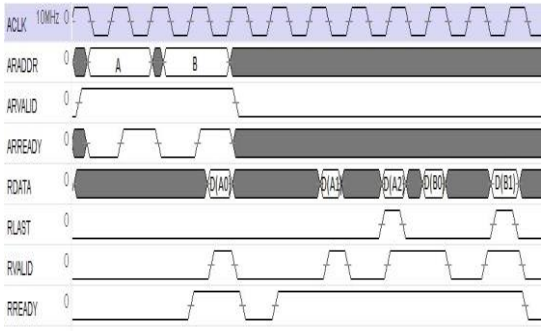


Figure 5. Overlapping read bursts

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.

***3. 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.

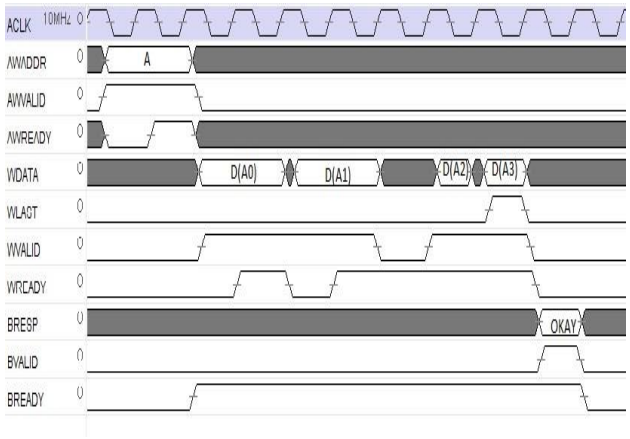


Figure 6. Write burst

## 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.

The AXI Protocol Checker is used to debug interface signals in systems using the AXI4, AXI3 or AXI4-Lite protocol. When placed in an AXI system, the connection of the AXI Protocol Checker monitors the traffic between the AXI Master and AXI Slave Core. The interface is checked against the rules outlined in the AXI Specification to determine if a violation has occurred. These violations are reported in a simulation log file message and as a debug net in the Vivado Logic Analyzer. In addition, the violations appear on the status vector output port from the core.

The AMBA AXI protocol is targeted at high-performance, high-frequency system designs and includes a number of features that make it suitable for a high-speed submicron interconnects.


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

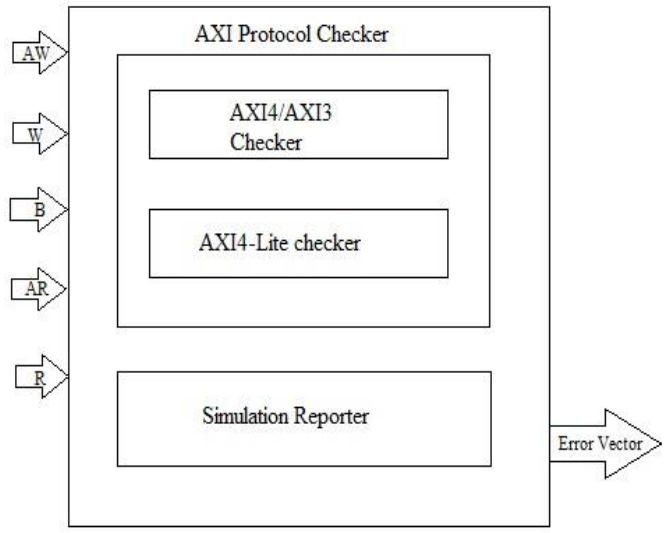


Fig 7. AXI Protocol CIRCUIT DIAGRAM

***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.

***4. Atomic operations:***

The AXI protocol defines mechanisms for both exclusive and locked accesses.

***5. Error support:***

The AXI protocol provides error support for both addresses decode errors and slave-generated errors.

***6. Unaligned address:***

To enhance the performance of the initial accesses within a burst, the AXI protocol supports unaligned burst start addresses.

## IV. RESULTS

***G.Cache Enable:***

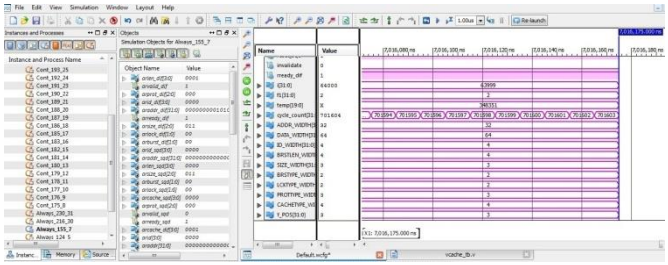

Fig 8. Cache Enable

***H. Cache Disable:***

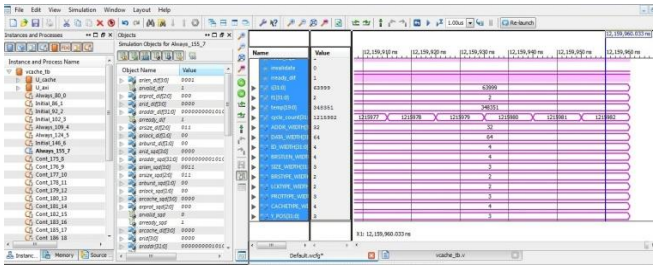

Fig 9. Cache Disable

## V. CONCLUSION

In this paper, we presented the Cache a novel solution for obtaining low power cache for high performance processors while addressing the reliability issues raised by process variability. We explored the design space of Cache architecture and its components. Our simulation results indicate a significant improvement in total energy consumption across simulated benchmarks. We consider Cache as a logical extension to drowsy cache, further improving its dynamic power consumption. 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.

[8] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol., no. 4, pp. 23–29, Jul.–Aug. 1999. 10. Y.-F. Tsai, D. Duarte, N. Vijaykrishnan, and M. J. Irwin, "Implications of technology scaling on leakage reduction techniques," in *Proc. Des.Autom. Conf.*, Jun. 2003, pp. 187–190 .

[9] K.Kuhn, "32nm SOC process, getting many things right at once, " Mar. 26,2010. [online]. Available: http://blogs.intel.com/technology/2010/03/32nm_soc_process_an_analogy_wi.php

**Y Ch Sekhar** was born on 17th Aug 1990 at Vijayawada, India. He received his, B.Tech in Electronics & Communication Engineering's from Lenora College of Engineering, Rampachodavaram, AP, India and Pursuing M.Tech in VLSI at K L University, Vijayawada, AP, India. His research interests in, Digital VLSI Design and Low Power VLSI Design.


*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

**Dr.P.S.Srinivas Babu** did his M.Tech in 2004 and PhD in 2009 both from College of Engineering, Andhra University, Visakhapatnam, India. Later he completed one year Post Doctoral Research with full financial support from National Research Council, Italy during 2011-2012. His area of interest includes Micro Electro Mechanical Systems (MEMS) for Biological & RF applications, Nano Electro Mechanical Systems (NEMS), Advanced Sensor System Technologies. He is the author for several National and International journal and conference publications. He visited Italy, Germany, Austria and Slovenia to attend various Conferences/Workshops. Currently, he is Professor of Electronics & Communication Engineering, K L University, Vaddeswaram, India.