A Fault Tolerant Cache Architecture for Sub 500mV Operation: Resizable Data Composer Cache (RDC-Cache)

Avesta Sasan (Mohammad A. Makhzan), Houman Homayoun, Ahmed Eltawil, Fadi Kurdahi
Electrical Engineering and Computer Science Department
University of California Irvine
{mmakhzan, hhomayou, aeltawil,kurdahi}@uci.edu

ABSTRACT
In this paper we introduce Resizable Data Composer-Cache (RDC-Cache). This novel cache architecture operates correctly at sub 500 mV in 65 nm technology tolerating large number of Manufacturing Process Variation induced defects. Based on a smart relocation methodology, RDC-Cache decomposes the data that is targeted for a defective cache way and relocates one or few word to a new location avoiding a write to defective bits. Upon a read request, the requested data is recomposed through an inverse operation. For the purpose of fault tolerance at low voltages the cache size is reduced, however, in this architecture the final cache size is considerably higher compared to previously suggested resizable cache organizations [2][3]. The following three features a) compaction of relocated words, b)ability to use defective words for fault tolerance and c) “linking” (relocating the defective word to any row in the next bank), allows this architecture to achieve far larger fault tolerance in comparison to [2][3]. In high voltage mode, the fault tolerant mechanism of RDC-Cache is turned-off with minimal (0.91%) latency overhead compared to a traditional cache.

Categories and Subject Descriptors
B.3.1 [Semiconductor Memories]: Static Memory (SRAM)

B.3.2 [Design Styles] Cache Memories,

B.1.3 [Control Structure Reliability, Testing and Fault-Tolerance]: Error Checking, Redundant Design.

General Terms
Algorithm, Design, Reliability, Theory

Keywords
Remapping Cache, Variation Aware Cache, Low Power Cache, low power memory organization, low power design, Fault Tolerance, VFS, Memory organization.

1. INTRODUCTION
With migration of fabrication technology to nanoscale transistor dimensions, CMOS circuits suffer from performance and power yield losses due to short channel effects that exacerbate process variation effects [1]. Due to the random nature of local process variation, resulting defects exhibit a random and uniform distribution [1] that adversely affect the expected system yield. This in turns leads to higher defect rates especially in memory intensive devices that are sensitive to changes in operation parameters including temperature, voltage and frequency. Furthermore, voltage scaling exponentially increases the impact of process variation on memory cell reliability, resulting in an exponential increase in the fault rate [4-8]. This introduces a tradeoff between cache yield and minimum achievable voltage Vcc[3]. In order to improve the cache yield and/or lower the minimum achievable voltage scaling bound many fault tolerant mechanisms are suggested. By tolerating a number of defects, a fault tolerant mechanism allows operation in lower voltages and/or improves the production yield. Having an error tolerant mechanism in place, usually require spending some extra power for supporting logic, introduces a certain area overhead, and might result in changing some system parameters. In case of caches and SRAMs this could be a change in the effective cache size, cache cycle time and/or its latency. At the same time each fault tolerance mechanism is capable of tolerating a certain defect rate. In this paper we address the process variation defects. As it is shown later in this paper, the number of these defects grows exponentially as the voltage is reduced. Since our target is achieving the lowest possible Vcc (sub 500mV range) we should be able to tolerate a very large number of defects. The larger the number of tolerated defects, the lower the achievable bound of Vcc is. We propose a fault tolerant architecture for caches that detect and correct the memory defects via resizing the cache. Due to its ability to compress defective locations, the proposed architecture, shows much higher fault coverage when compared to previous fault tolerant resizable caches reported in the literature [2][3]. While the compression of defect locations tends to slow down the process of cache resizing, the effective cache size is higher (or equal in worse case) as compared to prior work [2][3].

2. Related & Prior Work
The simplest solutions for providing moderate fault tolerance against process variation, is changing the SRAM basic cell size and design. Increasing the size of the transistor within the memory cell reduces the effects of gate width and length variation and reduces impact of random Dopant fluctuation. This results in a narrower distribution of access and write time in different voltages which in turn lowers the defect rate at each voltage. In addition using 8-T, 10-T and Schmidt Trigger ST-Cells [9] could also reduce the impact of process variation on the cell behavior. However such changes to the cell sizing and design result in a fast increase in the memory area. Kulkari et al [9] compared the 6T,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CASES’09, October 11–16, 2009, Grenoble, France.
Copyright 2009 ACM 978-1-60558-626-7/09/10 ...$10.00.

8T and 10T cell with their proposed ST 10T-cell and showed better low voltage reliability compared to other designs, however using ST-Cell incurs a 100% increase in the memory array area. In addition improving the reliability by changing the cell design reduces the statistical chances of failure and improves yield. A drawback of this approach is that after production, the system will not be able to tolerate new defects due to temperature variation, aging, and etc. Usually such pre-layout designs for reliability improvement should be coupled with an additional architectural detection and correction mechanism to increase life time reliability in addition to the yield.

Traditionally, a more general approach is the use of row and column redundancy [10][11], which is widely practiced. Redundancy is a good mechanism with low performance and area overhead for tolerating few manufacturing defects. With migration to nanometer regimes and the resulting exponential rise in the process variation induced defects, row and column redundancy fall short of tolerating this large number of defects. At lower voltages, where every cache row is likely to contain defects, the row and column redundancy are practically useless.

On a system level approach, a wide range of Error Detection Code (EDC) and Error Correcting codes (ECC) could be used. ECC is proven as an effective mechanism for handling soft errors. However using ECC alone for tolerating process variation induced defects has several major limitations: First is the increased vulnerability against soft errors. Any row that utilizes the ECC mechanism for detection and correction of a process variation induced defect is vulnerable and defenseless against soft error occurrence. This encourages using multi bit Error Detection and Correction codes. Secondly, using ECC codes incurs a high overhead in terms of storage for the correction code, large latency, slow and complex decoding [12].

The fault tolerant issue is also addressed from an organization stand point of view. In [13] the authors present the concept of using a victim cache, referenced to as the Inquisitive Defect Cache (IDC), as a small direct or associative cache that works in parallel with L1 cache and provides a defect free view of the cache for the processor in the current window of execution. However, in this work the basic assumption is that the data, if lost, could be recovered from lower level cache or memory and thus could only work for hierarchical structures. The concept of RDC-Cache (this work) however is applicable to any memory structure. A recent paper from Intel's microprocessor technology lab [3] suggested the use of fault tolerant mechanisms trading off the cache capacity and associatively for fault tolerance. The proposed approaches (assuming similar Probability of cell failure in 65nm and 130nm and using 130 nm probability of failure curve) allow scaling the voltage from a nominal 0.9 v down to 500mV in a 65nm technology. The cache size is reduced to 75% or 50% depending on the mechanism that is used. When compared to our proposed architecture, the RDC-Cache fault tolerance is considerably higher. This is due to the fact that the relocated defective words are saved in the RDC-cache in a compressed form. In addition the cache size is reduced just enough to provide the necessary fault coverage and therefore for all configurations, the RDC-Cache experience larger effective cache size in compare to that suggested in [3]. In fact the lower bound of cache size, in the worse case in RDC-Cache is equal to that offered in [3]. The work in [2] suggested resizable caches. In this technique it is assumed that in a cache layout, two or more blocks are laid in one row, therefore the column decoders are altered to choose another block in the same row if the original block is defective. Not only is the effective cache size in this case quickly reduced, the limit of fault tolerance is much lower than that achievable by RDC-Cache. In addition, this method interferes with temporal locality of the data.

In this paper we introduce a resizable fault tolerant cache organization that provide larger effective cache size in lower voltages compared to previously suggested organizations[2][3].

3. Voltage Scaling & Memory Failures

3.1 Classification of Memory Errors

Classically, failures in embedded memory cells are categorized as either of a transient nature, dependent on operating conditions, or of a fixed nature due to manufacturing errors. Symptoms of these failures are expressed as either: (1) an increase in cell access time, or (2) unstable read/write operations. In process technologies greater than 100nm, fixed errors are predominant, with a minority of the errors introduced due to transient effects. As technology scaling progresses, due to the random nature of the fluctuation of Dopant atom distributions and variation in gate length, this model cannot be sustained. In fact, in sub 100nm design, Random Dopant Fluctuation (RDF) has a dominant impact on the transistors' strength mismatch and is the most noticeable type of intra-die variation that can lead to cell instability and failure in embedded memories [12]. This Manufacturing Induced Process Variation (MPV) results in mismatch in the intrinsic threshold voltage (Vth) of neighboring transistors. When applied to memory cells, due to the analog nature of memory operation, the Vth variation results in large variation in access and write time to the memory cells. Dependence of Vth to the temperature makes the write/access time sensitive to the die temperature. In addition since the transistor's speed is a strong non-linear function of the separation between Vth and Vdd, the access/read time is strongly and non-linearly dependent on the supply voltage.

Table 1: Change in the mean and access time with Vdd

Voltage	Mean(ps)	Standard Deviation (ps)
0.9	43.77	7.504
0.8	65.75	13.873
0.7	91.6	19.987
0.6	136.9	26.35
0.5	197.54	37.038

3.2 Memory Access Time & Process Variation

To model the access/write time distribution as a result of MPV, a simulation was setup where MPV effects are lumped into an independent Gaussian distribution characterizing the Vth fluctuations of each transistor [14]. The circuit under test is a standard six transistor SRAM memory bit cell. The SPICE models used for the simulation were obtained from the Predictive Technology Model (PTM) [14] website in 32nm.

Due to the random and uniform distribution of Vth , it is expected and verified by Monte-Carlo simulation that the access/write time to the cache follows a "Gaussian like" distribution. However as the supply voltage to the memory changes, the characteristic of access/write distribution is expected to change. We repeated 10K Monte-Carlo Simulation for each voltage point. By fitting the

obtained iteration points at each voltage to the closest Gaussian distribution, we obtained the associated mean and standard deviation. The data is presented in Table 1. As illustrated, at lower voltages not only does the mean access time change, but also the standard deviation from mean widens

3.3 Defining System Access Time, Safety Margin (SM) and its Implications on Memory Cell Failure Rate

The conventional model for defining an access time at a voltage point when the access distribution is known is to choose an access time large enough that the probability of the distribution function tail that exceed the defined access time is very small. This is illustrated in Figure 1. This probability is determined by the designer depending on expected yield of the structure and the amount of dedicated redundancy available. The gap between mean access/write time and defined cycle time is referred to as 'Safety Margin' (SM). Choosing a large SM will result in higher yield however it adversely degrades the system's performance. In addition as the SM is increased, the power consumption of the memory device is also increased. A larger safety margin implies always expecting a larger differential voltage between the word-lines and therefore a larger waste in dynamic power (design based on worse case and for smallest differential voltage between bitlines).

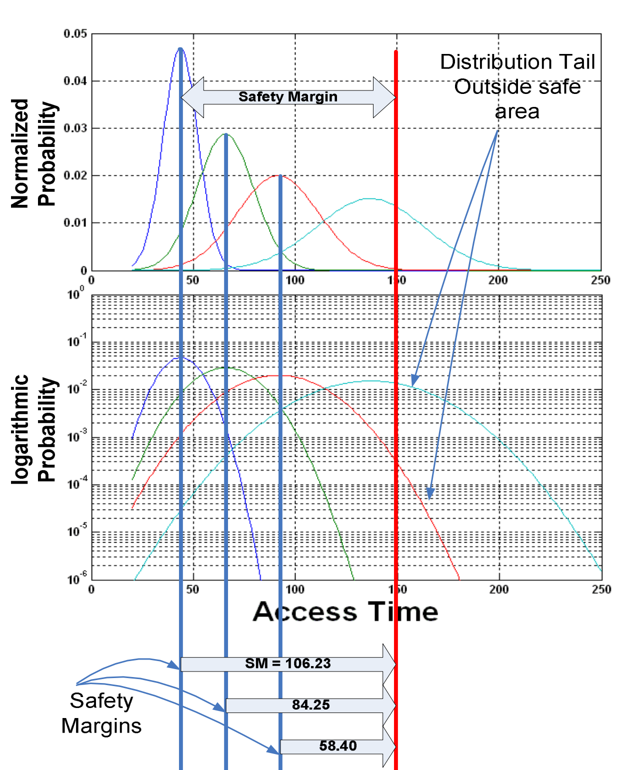

Figure 1: Voltage scaling and change in the mean, and standard deviation of access time distribution and change in safety margin for an FFVS policy with cycle time of 150ps

As the supply voltage is scaled, the mean (μ) and standard deviation (σ) of the access/write time grow larger. It is however still possible to define the cycle time at each voltage such that the probability of failure stays constant, but moving to lower voltages increases the mean and standard deviation quickly and trying to maintain the same probability of failure results in extremely long access time and extremely poor performance. On the other hand if the cycle time is chosen such that failure probability is traded for the performance, the failure rate will increase thus necessitating a fault tolerant architecture.

In order to obtain a mapping for the probability of failure to voltage, we chose an architecture that controls the sense amplifier activation via a delay buffer unit. The SM is practically added to the delay model by choosing a delay unit that ideally (i.e. no process variations are present) activates the bitlines whenever a differential voltage of 26mV is developed given that the sense amplifier is capable of detecting the signal with only 14mV differential voltage between its word-lines. At lower voltages the extra propagation delay of the control signal through the delay unit extends and defines the SM. Monte Carlo Simulation results of 10^8 iterations were used for obtaining the failure probability across different voltages. Using this methodology for defining SM at each voltage level results in the failure curve illustrated in Figure 2. This curve represents the combined rates of memory read, write and access failures.

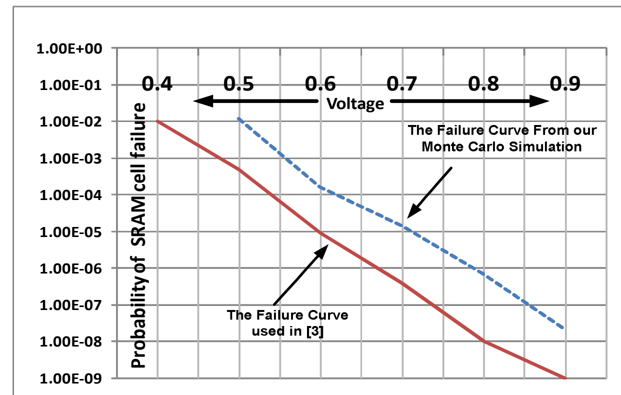

Figure 2: Probability of cell failure

We note here that our results are merely a "case study" to capture the trend of change in failure rate as supply voltage is scaled. The relationship between failure probability and voltage is affected significantly by several design variables such as technology size, cell design, cell size, bitline and wordline organization, sense amplifier specification and/or design, to name but a few. Even when considering the effect of all these "degrees of freedom", the scaling of the supply voltage will always result in an exponential change in the number of defective cells. In order to compare "apples-to-apples" the effectiveness of our architecture against some previously proposed architecture [2,3] we use the failure rate trend reported in [3] for 65 nm technology. This curve is also reproduced in Figure 2. It is interesting to note that the failure rate in 32nm is almost two orders of magnitude higher than that in 65nm, however the trend of failure response to voltage scaling is similar.

4. Proposed Architecture: RDC-Cache

The RDC-Cache is designed to provide tolerance for defects rates in sub 500mV voltage range. The proposed RDC-Cache could be turned on or off based on operating voltage level. When at higher voltages turning off the Fault Tolerance Mechanism (FTM) of RDC-Cache lowers both access time and power consumption. The delay associated with gating mechanism for switching the FTM to be active/gated is studied and as it will be discussed is very small compared to the overall cache latency (around 0.28%).

4.1 RDC-Cache Concept & Organization

In the RDC approach, banks are arranged in a circular chain, with each bank providing fault tolerance for the previous bank in the chain and the first bank providing tolerance for the last bank. This is illustrated in Figure 3.

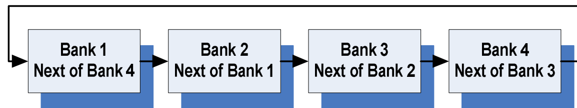

Figure 3: Banks are organized in a circular chain; for each bank, its next-bank will provide fault tolerance

RDC-Cache provides a word level fault tolerance. It generates and keeps a special defect map that has the defect information at a word level granularity. In RDC-Cache the last cache way in each row is used for fault tolerance (FT-way). If a cache way contain a defective word, the information that are mapped to that defective word are relocated and saved in FT-way in its next bank in the circular chain. RDC-Cache uses a mechanism that allows saving the relocated words of two or more ways in one or more rows in a single FT-Way. A FT-way that all its words are used as a destination for relocated words is called "saturated". Similar to [3], the proposed defect handling mechanism (DHM) provides tolerance for defects in data, and not for tag bits. Tag bits at lower voltages could be protected by a combination of upsizing the tag cells and using ST-Cells [9]. In addition, the proposed DHM could be combined with ECC to achieve even lower limits of voltage scaling and tolerance against soft errors.

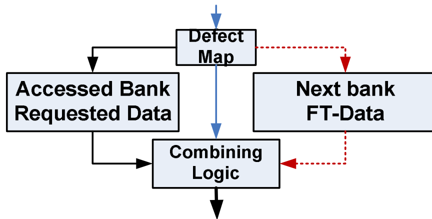

Figure 4: access to a RDC-Cache in low voltage mode

When choosing the destination FT-Way for relocated words, the RDC-Cache first uses unsaturated FT-ways that contain defective word(s). Then it uses the unsaturated previously used FT-ways that are not yet saturated and finally the defect free FT-ways. This allows us to keep the maximum possible number of defect free FT-ways. Finally, if these FT-ways are not used they are released and used as ordinary ways in the cache. This increases the final RDC-cache size compare to previously suggested resizable caches [2][3]. The process of associating a FT-Way in the next bank to a defective cache way is referred to as "linking". The proposed structure allows linking of any defective way to any FT-way in its next bank.

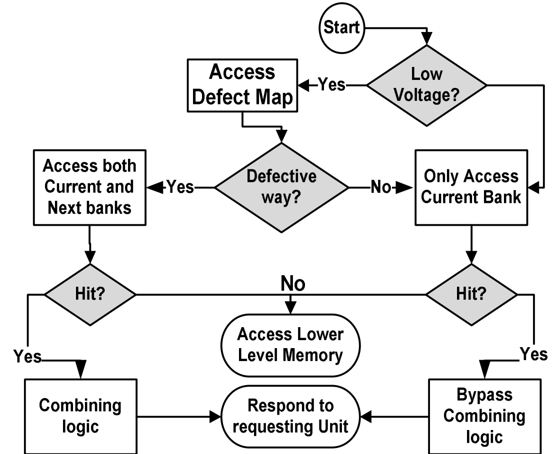

Figure 5: access flowchart to a RDC-Cache in low voltage mode

At low voltage, when reading from a defective cache way, as it is illustrated in Figure 4, the RDC-Cache first identifies the location of relocated words from defect map, and then accesses both banks (addressed bank and one containing relocated words) at the same time. Then through logical operations (combining logic), based on the defect map of an accessed cache way, FT-way etc., it combines the information in both cache ways and generates the defect free fetch group which is sent back to the requesting unit. The usage of another memory bank for remapping of defective words is a means to avoid designing multi-port caches to improve area and delay of the cache. The access scenario to a RDC-Cache is illustrated in the flowchart in Figure 5.

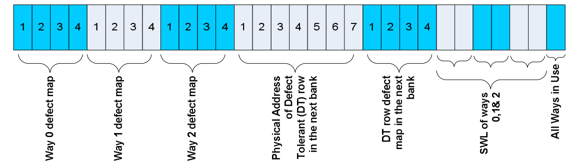

Figure 6: different fields in each row of RADM

4.2 Generating a Relocation Aware Defect Map

A raw defect map is generated at boot time. During the boot time, using the memory Built In Self Test (BIST) unit, the L1 and L2 cache(s) are tested under low voltage conditions. The output of the BIST is a raw defect map containing one bit per each word in the cache. If there are multiple operating points for different combination of Voltage, Temperature and Frequency, the BIST operation is repeated for each of these settings. The obtained defect map is then modified and processed to be usable with RDC-Cache. Processing the defect map is done at full voltage and through a complied assembly program that realizes the pseudo code in Figure 8 and is explained next.

Each entry in Relocation Aware Defect Map (RADM) of a cache with associativity of 4 that uses one of its ways (last) for fault tolerance contains the fields shown in Figure 6. From the following discussion it will be trivial to extend the RADM to

cover caches of any associativity and any number of FT-ways per row.

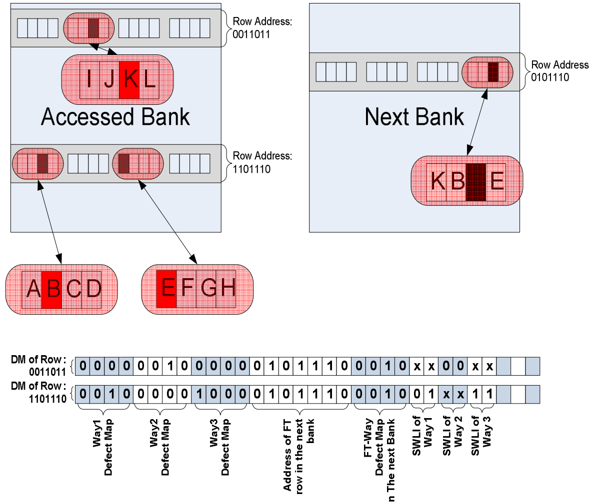

Figure 7: an example illustrating how RADM of two cache lines that are mapped to the same FT-Way in the next bank looks like.

A raw defect map has to be processed and converted to RADM format. Figure 7 illustrate an example of how RADM fields are generated. This figure illustrates the RADM or two rows that use the same row in the next bank for the fault tolerance. The first row [R#: "0011011"] contain a defect in the third word of its second associative way. The second row [R#: "1101110"] contain two defective ways one at the first way and the second one at its third way. The FT-way that is chosen in row [R#:0101110] of the next bank also has one defect in its FT-way. However the total number of available words is equal to that needed for tolerance of defects in rows "0011011" & "1101110". Figure 7 also shows the RADM for each of these rows. Each RADM entry includes the defect map of the first 3 associative ways, the address of the row containing the FT-way in the next bank, the defect map of the FT-way in the next bank followed by three 2-bit Starting Word Location Indexi (SWLIi) fields. Each SWLIi index points to the location of the first relocated word in the cache way "i". Equation (1) generalizes the size of RADM entry based on cache configuration. In Equation (1) A: is the associatively of the cache, W: is the number of words in each cache way, and R is the number of rows in each memory bank.

$$size_{RADM} = A \times W + Log_2^R + (A-1) \times \log_2^W + 1 \quad (1)$$

The ability to use a defective FT-Way for fault tolerance of defective ways in the main bank allows us to preserve the non defective FT-ways to be used only if no other defective FT-way was available. Therefore, if after RADM generation some of the FT-ways were unused they could be released increasing the cache associativity in that row by one and its capacity. In order to build RADM from a plain defect map the algorithm in Figure 8 is used. The input to this algorithm is the raw defect map and the output is the RADM. After BIST has generated a Raw Defect Map the core runs the assembly realization of pseudo code algorithm in Figure 8, processing the RADM. If there are multiple operating points (sets of Voltage, Temperature and Frequency) for each set, a separate RADM is generated. The resulting defect map is saved in a non-volatile memory and is loaded to cache's defect map when voltage is lowered to that associated to the generated RADM. The RADM is to be generated once, however if a new failure or defect is detected the RADM could be quickly updated. In order to enable quick updates to RADM, along with RADM a Fault Tolerant List (FTL) which is the list of unused FT-ways is also saved.

```
Structure LnkLE* { row,  index }
//  link list also contain remove and add operations

RDM[][][]=March()*;
// raw defect map

Create_FTL(RDM[][][]){
Wy = total number of cache way in each row.
  For B = 0 To size(B)
    For R = 0 to size(R)
      count = defect count in RDM[B][R][Wy];
      LnkLE LL; LL.row = R; LL.index = 0;    // link List
      FTL[B][count].add(LL);

Generage_RADM( RDM[ ][ ][ ] ){
FTL[ ][ ][ ] = create_FTL( RDM[ ][ ][ ]);  // fault tolerant
locations
cnt = 0; Wy = total number of cache way in each row.
For B = 0 To size(B)
  For R = 0 to size(R)
    For  W = 0 to [size(W) – 1]
      RADM[B][R][W] = RDM [B][R][W];
      count[W] = number of defective words in way W;
      cnt = cnt + count[W];
    LL = Find_FT_Row( FTL[(B+!)%size(B)][],cnt);
    ft_row = LL.row
    RADM[B][R][wy-1] = ft_row;
    RADM[B][R][wy]= RDM [B+1][R][Wy];
    swli = index of the first non defective bit in FT-way;
    For  W = 0 to [size(W) – 1]
      field_index = wy+1+W;
      RADM[B][R][field_index] = swli;
      swli = swli + count[W] + number of defects in
        FT way from bit swli to bit
swli+count[W]+LL.index ;
 For B = 0 To size(B)
   for LL=0 size of link list in FTL[B][wy]
   LL = FTL[B][Wy].remove();
  RADM[B][LL.row][2*wy+2] = 1;

Find_FT_Row( FTL[B][ ], cnt){
      unused_space = 0;
       while (FTL[B][cnt+unused_space] is an empty link
list)
           unused_space ++;
      LnkLE  LL = FTL[B][cnt].remove();
      if unused_space != 0; {
           LL.index = cnt-unused_space;
           FTL[B][unused_space].add(LL);
      return LL
```

Figure 8: The RADM generation algorithm

When scaling the cache voltage, we avoid scaling the RADM voltage. RADM sits on the critical path of the read and writes operation in lower voltages. The size of the RADM is fairly small compared to the cache (about 3.5%) however since its voltage is kept at high voltage, its contribution to the power consumption at lower voltages increases relative to the overall cache power. Having the RADM at higher voltage, requires designing dual voltage rails and/or voltage islands, which is standard practice in today's reference design flow [18]. Alternatively, the RADM power overhead could be reduced, at the cost of extra area and latency penalty if the RADM is realized via ST-Cells [4] allowing a single Vccmin across the chip.

4.3 Reading from RDC-Cache

Reading a way containing defective words from RDC-Cache involves reading the addressed bank, reading the FT-way from the next bank, and then passing the data through a Combining Logic Unit (CLU). The CLU also needs the defect map of the accessed cache way, and the FT-way. With this information provided, CLU will process and combine the words in the defective way with those obtained from FT-Way and produce the final defect free group of words to be sent back to the requesting unit. A simple realization of the combining logic for a 4 way associative cache is illustrated in Figure 9.

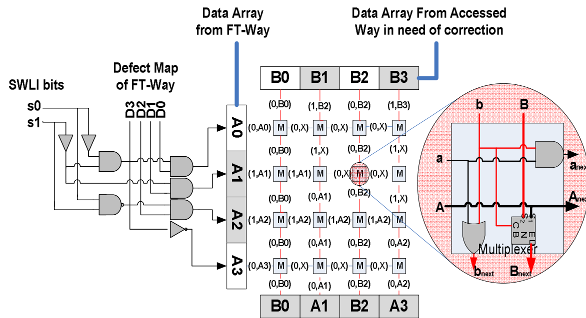

Figure 9: Combinational Logic Unit (CLU)

As explained previously, the relocated words to the FT-way are saved in a compact form. This means that one FT-way might be used to store defect-free copies of defective words located in more than one row in the previous bank. The first 2bits of the SWLI field in the defect map are used to realize the starting location (offset) of the first relocated word. It is possible that more than one defective word is in a cache way, however, we know that all these words regardless of their location in the original cache way are compacted next to each other. For the example, in Figure 9 the words B1 and B3 are defective and they are compacted and saved in locations A1 and A2 in a FT-way. In this case SWLI index is "01" meaning the first word is either defective or used for fault tolerance of another cache way. The combination of (s0,s1) bits and defect map of the FT-way could be used to generate an array of bits (a0,a1,a2,a3) that indicate the locations of relocated words in the FT-way. A simple realization of such a circuit is provided in Figure 9. This array of indexes along with defect map of the currently accessed defective cache way (b0,b1,b2,b3), its data (B0,B1,B2,B3) and finally the data of the FT-way (A0,A1,A2,A3) is the input to the Combining Mesh Grid (CMG). CMG is a matrix of M boxes. The functionality of each M box is very simple; M boxes help with routing the data words such that the relocated data words in the FT-way would find the proper location in the final fetch group. The logic of each M-box is fairly simple as following:

$$a_{next} = ab, b_{next} = a + b, A_{next} = A, B_{next} = bB + \bar{b}A \quad (2)$$

The defect information and SWLI indexes are available much earlier than the data in the accessed way and FT-way are available; therefore the effective delay that the entire combining logic introduces (in case of 4 way associative caches) is only 4 levels of MUX propagation delays. This delay linearly increases as the cache ways increases

4.4 Writing to RDC-Cache

Writing to a defective cache way in the RDC-Cache involves regrouping and compacting the words mapped to defective locations in the accessed way to their corresponding location in their associated FT-way. Before writing the information in the FT-way we should identify in which cache-way in the accessed bank, will the data be saved. Writing to the FT-way involves compacting the defective words together, shifting the compacted words to the appropriate starting word suggested by SLWI index in the defect map, and then going through a muxing stage to make sure data will not be saved in the defective locations in the FT-way. This process is simply achieved by a Decomposition Logic Unit (DLU) similar to that used for combining. Note that in this case writing to the FT-way is on the critical path of the write operation. Furthermore, writing to the FT-way cannot start until data has propagated through the decomposition matrix (in case of a 4 way associative cache, it is propagation delay of 4 multiplexer). Normally, the cache is designed so that, the write time is shorter than the read time. Thus, although writing to the FT-way extends the delay of write critical path, the write time is still expected to be much lower than the read time

Table2: System operating parameters for different voltages.

	High Voltage	Low Voltage
Processor Frequency	3GHz	500MHz
Memory Latency	300 Cycles	50 Cycles
Voltage	1.3 V	500mV

4.5 Access Delay Analysis

When the voltage is lowered, both analog (word-line fire to sense amplifier detection) and digital sections (decoding, comparison, hit signal generation, buffering and propagating through inverter chain in the output driver) of the cache operation take longer time. In addition to that in the proposed architecture every access to the cache is extended by the latency introduced by reading the defect map as well as the propagation delay through the combining logic. As a tradeoff, this increase in the access latency enables the cache to tolerate a higher defect rate. Viewing the same problem from another perspective, by tolerating a higher defect rate, the proposed cache architecture achieves the same cache yield at lower voltages.

In order to determine the excess delay introduced by CLU processing and RADM lookup, we simulated the post-layout structure of a cache in 65nm technology. Using synthesis tools and considering 20FO4 [3] delay per cycle, we determined the excess delay introduced by RADM lookup and CLU. In case of our 32KB cache with access time of 3 cycles, the access time was extended by 0.92 cycles, effectively allowing access time to be done in 4 cycles. For a cache size of 2MB arranged in 8 banks and

8 ways per row, the access time was extended by 1.89 cycles. We repeated the simulation using 32nm PTM[16] model and achieved similar results (0.97 and 1.91 cycles for 32KB and 2MB caches respectively) allowing us to effectively simulate the 32KB and 2MB caches with 1 and 2 cycle overhead. In our simulation setup explained in section 5, in lower voltages the delay of the 32KB L1 and 2MB L2 caches are increased by this excess amount to account for CLU and RADM lookup delays

Table 3: SimpleScalar configuration

ROB size	256
Register File Size	256 FP, 256 INT
Fetch/schedule/retire/width	6/5/5
Scheduling Window Size	32FP, 32 Int, 32 Mem
Memory Disambiguation	Perfect
Load/Store Buffer Size	32/32
Branch Predictor	16KB
Cache Line Size	64 Byte
L1 Data and Inst Cache Size	32 KB, 8Way, 3 Cycles
L2 Unified Cache	2MB, 8Way, 20 Cycles
Execution Length	2B Fast Forward, 2B execution

5. Simulation Methodology

We used SimpleScalar [17] to simulate the SPEC2000 binaries when L1 and L2 cache access latencies are defined to those appropriate for RDC-Cache. Table 3 illustrates the setup of SimpleScalar for this simulation which is similar to that used in [3] and reflective of Intel® CoreTM 2 Duo processor on 65 nm technology [3]. The results are compared to an ideal cache capable of defect free operation in low voltage with no performance and delay penalty or capacity loss. This also enables us to compare the architecture to that given in [3]. Although our 32KB cache is able to work at 440mV and our L2 cache in 475mV with still passing the 999 in 1000 yield test, we used the supply voltage point to be 500mV so we could compare the architecture to that in [3].

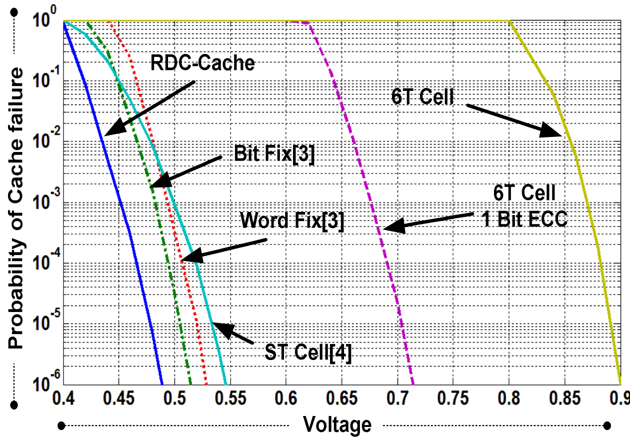

Figure 9: Probability of cache failure for different fault tolerant mechanisms

6. Results & Discussion

The probability of a 32KB RDC-Cache failure is illustrated in Figure 9. This figure also compares the Failure probability of other caches with the same size realized by different Fault tolerant Means. We adopt the definition for Vcc-min as the voltage at which 1 out of every 1000 cache instances is defective [3]. With no fault tolerant mechanism in place, a 32KB cache composed of 6T SRAMS, based of failure probability provided in Figure 2, have a Vcc-min of 0.87V. Introducing a 1 Bit ECC reduces the Vcc-min to 0.68V. On the other hand if memory array is realized via ST-Cells the Vcc-min is effectively reduced to 500mV. However an area penalty of 2X in the array size incurs. The Word-Fix Fault Tolerance mechanism suggested in [3] also reduces the Vcc-min lower but close to 500mV. The Bit-Fix mechanism in [3] further reduces the Vcc-min to 480mV. However the cache size is both of methods suggested in [3] is lower than that realized by RDC-Cache. Finally the RDC-Cache realizes the Vcc-min at only 450mV.

Figure 10 compares the RDC-Cache size to those suggested in [2] and [3] across different voltages (different failure rates). In the Word-Fix[3] and Bit-Fix[3] mechanism the cache size at lower voltages is constant. For our case study of a 8 way associative 32KB cache, Word-Fix scheme uses 2 out of the 8 ways for masking the defective words resulting in 25% loss in the cache size. The Bit Fix mechanism on the other hand incurs a 50% loss since it uses 4 out of every 8 ways for saving the needed defect tolerance information (patches and pointers). The Resizable Cache suggested in [2] have a higher cache size in lower defect rates however in lower voltages quickly downsize reaching 50% loss at 0.62V. The RDC-Cache However is able to constantly realize a higher effective cache size compare to other schemes.

Table 4: Low voltage properties of RDC-Cache for 32KB and 2MB cache

L1 & L2 are 8 way associative.		**Cache size**	
		32 KB	**2 MB**
Area Overhead		**6.72%**	**6.83%**
increase in access time in high Vdd		**0.91%**	**0.18%**
increase in access time in low Vdd		**32.1%**	**9.8%**
Reduction in leakage in 500mV	**1.3V**	**86.26%**	**85.42%**
	0.87V	**61.37%**	**60.14%**
Reduction in dynamic Power at 500mV	**1.3V**	**79.2%**	**78.36%**
	0.87V	**59.39%**	**57.92%**
Total Power reduction at 500mV	**1.3V**	**83.79%**	**82.64%**
	0.87V	**60.68%**	**59.16**
Vccmin to get 999 in 1000 yield*		**450mV**	**485mV**
Power reduction at Vccmin	**1.3V**	**85.54%**	**83.06%**
	0.87V	**61.23%**	**59.93%**

Table 4 summarizes the cost related to RDC-Cache. The area overhead of the RDC-Cache for a 32KB cache including RADM, CLU and DLU obtained after layout analysis is ~6.72% in 65nm technology. In higher voltages the RADM, DLU and CLU are power gated and by muxing they are removed from critical access path but still the access time is increased by 0.91% in L1 cache and %0.18 in L2 Cache. The percentage increase in the access time of the L2 cache is larger since the L2 cache is designed for 20 cycle access time where as L1 cache has a 3 cycle access time. In lower voltages the RADM, CLU and DLU are in the critical path and they increase the access time by 32.1% and 9.8% for L1 and L2 cache respectively. This results in an increase in the access time of the L1 from 3 to 4 cycles and in L2 from 20 to 22 cycles. Table 4 also lists the Vccmin for a 32KB and a 2MB

cache to be ~450mV and ~485mV. The reduction in dynamic, leakage and total power consumption of both cache instances is reported. For obtaining the percentage power reduction we have compared them to a cache operating at 1.3V (Fast-Fast corner in 65nm) and one at 0.87V (which is the Vccmin of a 6T with no fault tolerance in place). The reduction in power when each cache instance is operating at 500mV and also when it operates in its Vccmin is reported.

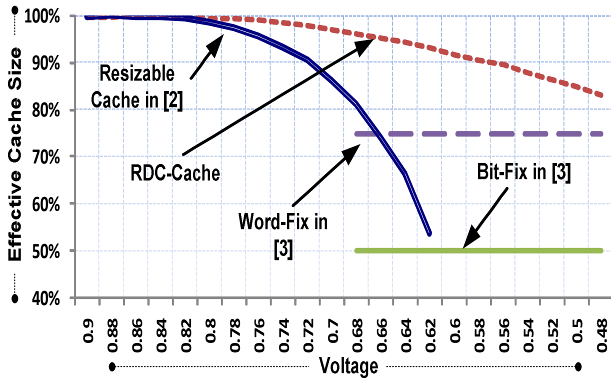

Figure 10: Effective cache size after resizing to cover all defects.

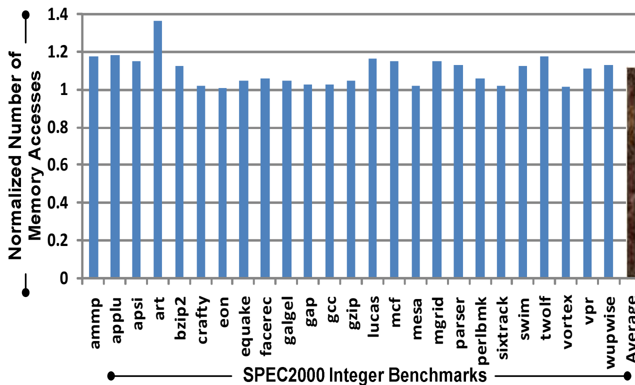

Figure 11: Normalized number of access from L1 to L2 Cache

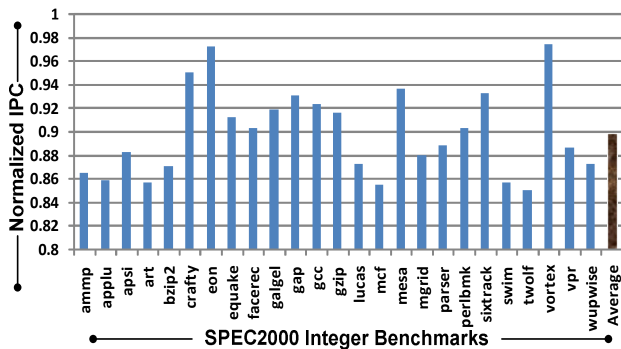

Figure 12: Normalized IPC of SPEC2000 benchmarks

Figures 11 and 12 illustrate the results of SimpleScalar simulation. For obtaining the figures the binaries are fast-forwarded for 2Billion instructions and executed for 2Billion instructions. Figure 11 illustrates the normalized number of accesses from L1 to L2 Cache (i.e. increase in access to lower level memory), and Figure 12 illustrates the normalized IPC. Comparing the figures side by side reveals a non-linear relationship between increase in the number of L1 to L2 cache access and decrease in the IPC for each benchmark. Increase in the number of accesses to lower level memories is due to reduction in the cache size. If the cache size is further reduced (to the limits of [2] or [3]) This results in quick reduction in the IPC. This in turn increases the execution time which is followed by increased energy consumption for doing more work (dynamic power) and leaking over a longer period. Since RDC-Cache achieves the target sub 500mV voltage with larger effective cache size compared to that of [2] or [3], even if they are operated at the same voltage, or if RDC-Cache has slightly higher power consumption, it incurs lower final energy cost. Our simulation results reported 61% reduction in the Energy consumed Per executed Instruction (EPI). Which is higher than that of [3] (53%) and [4] (55%).

7. Conclusion

In this paper, we presented a novel, process variation-tolerant cache architecture. This architecture outperforms previously published designs due to the finer granularization of faults and enhanced packing of fault-free redundant regions, which increases the effective cache size. This technique is not limited to one memory hierarchy and can be extended to model any cache-based memory structure. Many other degrees of freedom can complement the proposed technique to achieve an even more diverse design space exploration and tradeoff between area, power, performance and reliability.

8. REFERENCES

[1] Jaffari, J., Anis, M., "Variability-Aware Bulk-MOS Device Design," Computer-Aided Design of Integrated Circuits and Systems, IEEE TCAD, Feb. 2008

[2] Argawal, A. et. al. “Process Variation in Embedded Memories: Failure Analysis and Variation Aware Architecture” Solid State Circuits, Transaction on , vol.40, no.9 Sep 2005.

[3] Wilkerson C. et. al.. “Trading off cache Capacity for Reliability to Enable Low Voltage Operation.” ISCA 2008.

[4] S. R. Nassif “Modeling and Analysis of manufacturing variation” in Proc. CICC, 2001

[5] S. Borkar, et. al. “Process Variation and impact on circuits and micro architectures,” in Proc DAC 2003 pp338-342

[6] S. Mukhopadhyay et. al. “Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in NanoScaled CMOS” CADICS DEC 2005

[7] A. Bhavnagarwala, et. al. “ The impact of intrinsic device fluctuation on CMOS SRAM cell stability,” IEEE J. Solid-State Circuits vol.36, no.4 pp 658-665 Apr 2001

[8] H. Mahmoodi, at al.. “Modeling of failure probability and statistical design of SRAM array for yield enhancement in nano-scaled CMOS,” IEEE Trans CAD , 2003

[9] J. P. Kulkarni, et. al., “A 160 mV Robust Schmitt Trigger Based Subthreshold SRAM,,” IEEE Journal off Solid-state Circuits, Vol.. 42, no.. 10, pp. 2303-2313, October, 2007.

[10] S. E. Schuster, “Multiple word/bit line redundancy for semiconductor memories,” IEEE J. Solid-State Circuits, vol. SC-13, no. 5, pp. 698–703, Oct. 1978.

[11] M. Horiguchi, “Redundancy techniques for high-density DRAMS,” in Proc. 2nd IEEE ICISS, Oct. 1997, pp. 22–29.

[12] J. Kim, et. al., “Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding,,” Micro-40, December 2007.

[13] M. A. Makhzan (A. Sasan), A. Khajeh, A. Eltawil, F. J. Kurdahi, "Limits on voltage scaling for caches utilizing fault tolerant techniques," Computer Design, 2007. ICCD 2007. 25th International Conference on , vol., no., pp.488-495, 7-10 Oct. 2007

[14] H. Mahmoodi, at al.. “Modeling of failure probability and statistical design of sram array for yield enhancement in nano-scaled CMOS,” IEEE TCAD , 2003

[15] A. Bhavnagarwala et. Al.. “The impact of intrinsic device fluctuations on CMOS SRAM cell stability,” JSSC, April 2001.

[16] http://www.eas.asu.edu/~ptm/

[17] http://www.simplescalar.com/

[18] http://www.design-reuse.com/news/13813/tsmc-continues-reference-flow-7-0.html