PAPER

An Adaptive Various-Width Data Cache for Low Power Design

Jiongyao YE[†a)], **Yu WAN**[†], ***and*** **Takahiro WATANABE**[†], ***Members***

SUMMARY Modern microprocessors employ caches to bridge the great speed variance between a main memory and a central processing unit, but these caches consume a larger and larger proportion of the total power consumption. In fact, many values in a processor rarely need the full-bit dynamic range supported by a cache. The narrow-width value occupies a large portion of the cache access and storage. In view of these observations, this paper proposes an Adaptive Various-width Data Cache (AVDC) to reduce the power consumption in a cache, which exploits the popularity of narrow-width value stored in the cache. In AVDC, the data storage unit consists of three sub-arrays to store data of different widths. When high sub-arrays are not used, they are closed to save its dynamic and static power consumption through the modified high-bit SRAM cell. The main advantages of AVDC are: 1) Both the dynamic and static power consumption can be reduced. 2) Low power consumption is achieved by the modification of the data storage unit with less hardware modification. 3) We exploit the redundancy of narrow-width values instead of compressed values, thus cache access latency does not increase. Experimental results using SPEC 2000 benchmarks show that our proposed AVDC can reduce the power consumption, by 34.83% for dynamic power saving and by 42.87% for static power saving on average, compared with a cache without AVDC.

key words: *low power, data cache, frequent values, narrow-width values*

1. Introduction

In recent years, power consumption has become a major constraint factor on the development of integrated circuits, especially in microprocessors. The cache, as a critical component of the modern processor, will constitute an increasingly larger portion of total microprocessor energy dissipation due to large size, high speed, and frequent access. For example, the DEC Alpha21164 dissipates 25% [1] and the StrongARM SA-110 dissipates 43% [2] of its total power in caches. Thus, reducing the power consumption of cache is thought to be effective to reduce overall processor power consumption.

In this paper, we propose an Adaptive Various-width Data Cache (AVDC) to reduce power consumption by exploiting value locality with little performance overhead. In AVDC, the data storage unit consists of three sub-arrays to store values of different data width. By checking the range of value, AVDC adaptively shuts off unused sub-arrays for reducing power consumption. Meanwhile, AVDC would not increase access latency. The proposed approach directly exploits the redundancy of values instead of using the compressed data, so the data can be directly accessed without decoding. Experimental results using SPEC 2000 benchmarks show that our proposed AVDC can reduce the power consumption, by 34.83% for dynamic power saving and by 42.87% for static power saving on average, compared with a cache without AVDC.

The rest of the paper is organized as follows. In the next section, related works and research motivation are described. In Sect. 3, we investigate the Various Width Value (VWV) that is the theoretical basis for this paper. In Sect. 4, we describe the AVDC. In Sect. 5, we present experimental results, and Sect. 6 concludes the paper.

2. Related Works and Motivation

The power consumption of integrated circuit is classified into dynamic power consumption and static power consumption. The dynamic power consumption is consumed by the state transition of transistor switch, which depends on the square of the supply voltage, and it is determined by the switch frequency of the transistor if the operating condition is determined. The static power consumption is caused by leakage current which appears even when no switching is taking place.

The static power consumption approximately equals the product of supply voltage and leakage current, and it was very small under the early technology. As a result, numerous approaches have been proposed to reduce the dynamic power consumption. For example, Block Buffering [19] increases a smaller storage between a CPU and a cache, which is used to shoulder most of cache access. In [4], another approach was proposed, where a cache is divided into several subbanking, and only a subbanking of data where the block is accessed reduces the redundant energy dissipation. Furthermore, a low-power reconfigurable data design based on locality and frequent value locality was investigated. With a little modification to the conventional architecture, the reconfigurable cache architecture could be reconfigured by itself with regard to a three-dimensional space, namely, cache capacity, line size and associativity to make compromise between performance and power consumption [6].

Other research efforts actively pursued a Frequent Value (FV) based compression method (e.g. FV Cache [7], [12]), which reduces the energy consumption by trading off between lower dynamic energy consumption for frequent value accesses and higher access times for non-frequent value accesses. FVs can be stored in the FV cache using a few bits after encoding instead of using full words. The

Manuscript received August 11, 2010.

[†]The authors are with the Graduate School of Information, Productions and Systems, Waseda University, Kitakyushu-shi, 808–0135 Japan.

a) E-mail: yejy_asgard@suou.waseda.jp

DOI: 10.1587/transinf.E94.D.1539

FV cache needs one cycle to read FV from the low-bit array. Then, the frequent value is still decoded for the required word. The latency for decoding can be reduced by using the subbanking scheme. For a non-frequent value, the required word need not be encoded. But, accessing a non-frequent value needs two cycles where the low-bit array is accessed at the first cycle, and the remained bit array is accessed at the second cycle.

In recent years, with the continuous development of integration, the threshold voltage becomes lower and lower, so that the static power consumption accounts for a larger portion of total power consumption [8]. So the static power reduction has been a significant problem, and many techniques have been proposed. For example, Dual-V_t [9] adopts higher threshold voltage to reduce leakage current on the premise of sacrificing the access speed; Gated-V_{dd} [10] reduces leakage in deep-submicron cache memories. Gated-V_{dd} inserts an extra transistor between the voltage source and the SRAM cell to selectively shut off some unused on-chip cache line, but it causes the loss of stored information. MTCMOS [11] dynamically changes the threshold voltage to make some storage cell in the dormancy state. But the dormancy storage cell maybe lose the stored information. Accessing the dormancy storage cell needs to wake it up in advance, which increases the access latency.

To overcome those drawbacks, the other solutions [3], [5], [20] have been proposed, which are based on turning off portions of the cache at the cost of increasing miss rates. A more aggressive approach proposed in [13], which is based on the FV cache [7], [12], allows shutting off the unused bit in the larger sub-array and uses 1-cycle latency for non-FVs as well as for FVs. Since FVs are stored in encoded form using only a few bits in the low-bit array, the remaining bits in the high-bit array can be shut off. This approach reduces data cache static energy by over 33% on average.

Most of above-mentioned technologies optimize either the dynamic power consumption or static power consumption. While the low static-power FV data caches [13] give consideration to both the dynamic and static power consumption, it still has some problems, that is: 1) To reduce the power consumption of the FV finder, the preceding study [13] runs the FV finder for the first 5% of memory accesses, but the partial runtime monitoring makes it difficult to select the appropriate FVs. 2) FV finder, FV encoder, and decoder register file cause additional power consumption. 3) FV caches cannot be adapted to General Purpose Processor (GPP) because it is very difficult to determine the monitoring time for finding an appropriate set of FVs.

3. Various-Width Value

A narrow-width value is defined as the value with a smaller width than the full-width of the dynamic range supported by typical 32 bits or 64 bits processors. The presence of narrow-width values has been well studied and exploited for performance and power optimizations in [21], [22]. We focus on exploiting narrow-width values to reduce power consumption of a data cache. Almost all of the modern processors use 32 bit or 64 bit data width, but they often deal with a large number of narrow-width data because a lot of small variables (e.g. loops, array suffixes etc.), are widely used in a program. There is similar situation in cache, too. According to the previous research [16] based on 64-bits data-width, on average, about 40% of all values can be represented using just 16-bits, another 45% of the values using 32-bits. Only about 15% of the values require full-width bits.

To investigate the case for 32-bits data-width, we experimented with a 32-bit RISC architecture. First of all, we analyzed all values in the data cache by executing the SPEC 2000 benchmarks and obtained the width distribution as shown in Fig. 1. On average, about 52% of all values have data-width of 8 bits or less, and about 82% values have data width of 16 bits or less. Furthermore, the values less than or equal to 4 bit constitute about 43% of all. Similar results are also reported in previous researches, in which the processors use wider data sizes (64-bit processors and beyond) [16].

Based on above observation, we carry out a non-uniform quantization for all of the values in the range of 0 to 2^{31}. The values are classified into three patterns as Short-Width Value (SWV) if its data-width is 4 bits or less, Medium-Width Value (MWV) for 5–16 bits, or Long-Width Value (LWV) for larger than 16 bits. SWV, MWV and LWV are called Various-Width Value (VWV) together. We also analyzed the VWV patterns in the data cache by executing the SPEC 2000 benchmarks, and compared with the FVs according to the number of the selected FVs in the data cache as shown in Fig. 2 (In Sect. 5, the experimental environment is described in detail). The results show that on the average about 43% values are SWV and 82% values are SWV+MWV (called as SM-WV). It is noted that the SWV occupies significant proportion, that is, small values are frequently used in the benchmarks (especially for 0 and 1). Figure 2 also shows the distribution of different grade VWV compared with FV-32 and FV-64 that mean the number of the selected FVs are 32 and 64, respectively. The results show that the distribution of SWV is similar to FV-32, and the most contribution of SM-WV of benchmark is larger than FV-64. SWV and SM-WV present very large proportion in the data cache according to these experimented re-

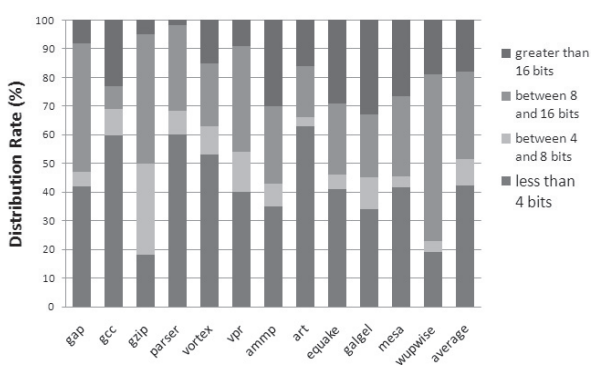

Fig. 1 Bit width distribution.

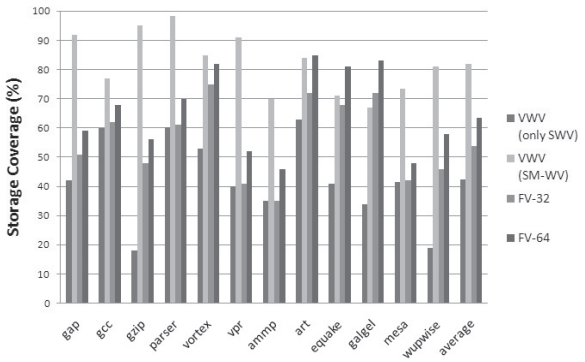

Fig. 2 Distribution of VWV compared with frequent value.

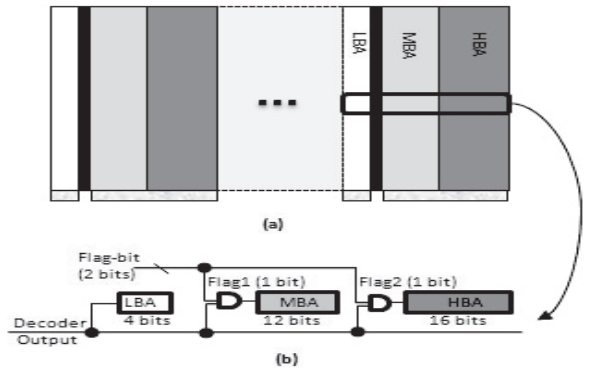

Fig. 3 Modified data array for AVDC (a) AVDC architecture (b) AVDC line architecture of one word.

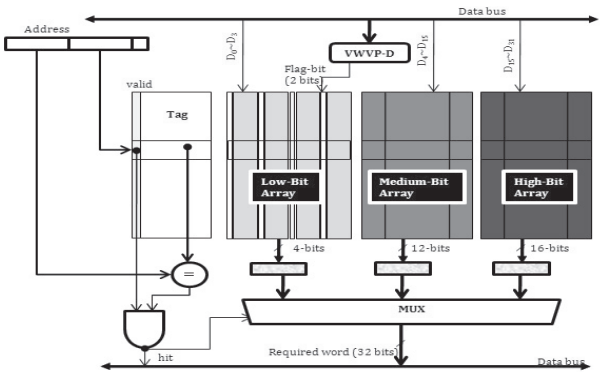

Fig. 4 Architectural design overview.

sults. Therefore, by storing the SWV or SM-WV in our proposed AVDC, the power consumption can be effectively reduced by shutting off unused higher bits.

4. Design of AVDC

This section introduces how the AVDC can reduce cache power consumption by modifying the SRAM architecture, and discusses the influence of this scheme to access delay and cache size.

4.1 AVDC Architecture

In AVDC, a data word is comprised of three sub-arrays and an additional 2 bits flag-bit. The three sub-arrays are 4-bits Low-Bit Array (LBA), 12-bits Medium-Bit Array (MBA) and 16-bits High-Bit Array (HBA). Figure 3 (a) shows the AVDC architecture. The contents of the flag-bit and output of the index driver are the two inputs of the AND gate, as shown in Fig. 3 (b). The flag-bit is composed of flag1 and flag2. If the flag is 0, the corresponding sub-array is shut off. Otherwise the sub-array is normally accessed. Figure 4 illustrates the overall architecture. We add a VWV Patterns Detector (VWVP-D) to capture VWV patterns, and decide whether the corresponding flag-bit needs to be set or reset. The VWVP-D is a very simple OR logic, so Fig. 4 does not give a specific circuit. For the value of width D (D = 32), the logic expression of flag-bit is as follows:

$$Flag1 = OR(D_4 : D_{31}) \quad (1)$$

$$Flag2 = OR(D_{16} : D_{31}) \quad (2)$$

Different with the reading operation of FVs cache, when reading a word from AVDC, the value stored in AVDC is not compressed data, so AVDC need not decode time. Thus, the whole 32 bit would be read out without access latency. If the value is LWV, all of the output 32 bits come from storage unit of word line. If the value is SWV or SM-WV, the used sub-arrays are normally accessed, and the unused bits can automatically export 0 through the modified sense amplifier (described in Sect. 4.3). The unused SRAM cell of an data array and sense amplifier are turned off, so the access to the unused bits are avoided, and the cache activity is reduced, too.

A write operation to AVDC is performed as follows: after the word to be written is identified by the VWVP-D, the value can be stored in LBA if the value is less than 2^4, or stored in LBA+MBA if the value is between 2^4 and $2^{16} - 1$, and the corresponding flag-bit is reset. In these two case, accessing to the unused array is avoided. Otherwise, the value is LWV, and all of LBA, MBA and HBA are accessed as well as the flag-bit being closed. For a cache write access, write data is usually held in a pipeline buffer for a cycle while the cache tags are checked. The VWVP-D can occur while the write data is waiting in the buffer and hence we expect no visible delay penalty.

4.2 SRAM Cell and Its Modification

Here, we need to modify the SRAM cell except LBA. The architecture of flag-bits SRAM cell has been explained in [13]. The flag-bit of AVDC adopts the standard 6T SRAM storage cell. Once the value is written into the bit, it must be kept until the next value is written. When the flag-bit is 1, the SRAM cells are normally used. When the flag bit is 0, the corresponding SRAM cell is shut off. Figure 5 shows the conventional SRAM cell and the modified SRAM cell. The modification consists of two ways: 1) using a Gated-V_{dd} technique [10] controls the cell to open or close. As extra pMOS transistor is integrated into the conventional SRAM

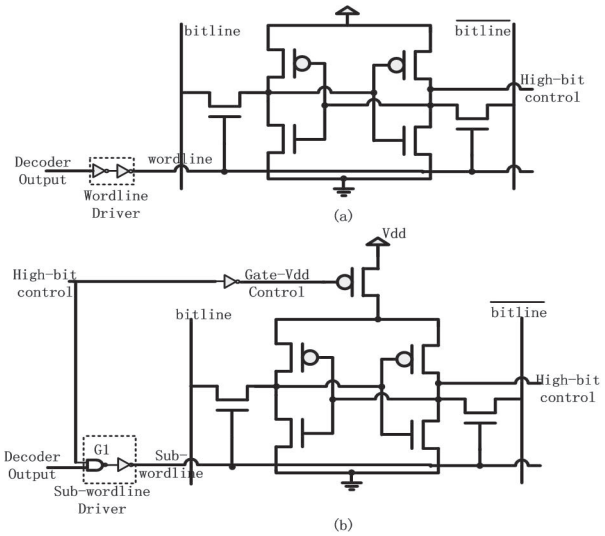

Fig. 5 SRAM cell modification. (a) Conventional SRAM cell and (b) modified high-bit SRAM cells for AVDC.

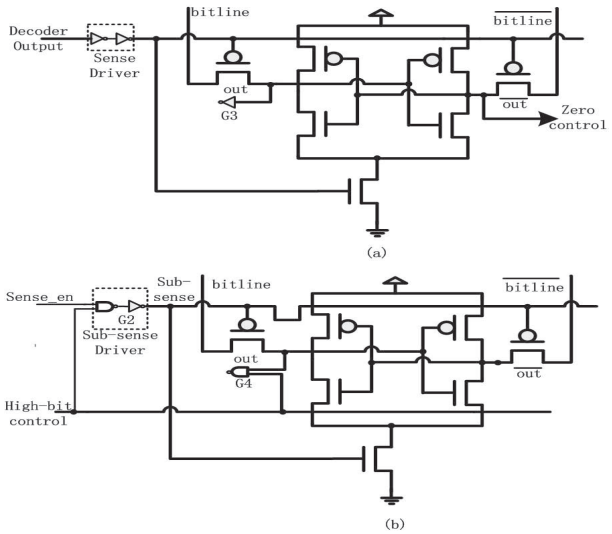

Fig. 6 Sense amplifier modification. (a) Conventional sense amplifier and (b) modified sense amplifier of high-bit cell.

cell. When the "Gated-V_{dd} Control" goes high, the SRAM cell's voltage is floated, turning off the entire cell. 2) A sub-wordline is added to control SRAM cell. The state of sub-wordline is decided by wordline and high-bit control signal that is corresponding flag-bit through the NAND gate (G1). The state of sub-wordline keeps the same with the wordline under normal situation. However, the state of sub-wordline will always remain inactive when the control signal is low.

4.3 Sense Amplifier and Its Modification

Using AVDC, the discharge operation of row line will not happen when the closed cell is accessed. It makes column line not produce obvious voltage difference. Figure 6 shows the traditional sense amplifier and the modified sense amplifier for AVDC. A traditional sense amplifier will introduce competition and jitter under such a situation. This will not only make the output state unpredictable, but also cause a large amount of waste of the power consumption. So we modify the sense amplifier of high-bit cells to solve this problem. Similar to SRAM cell modification, a sub-sense is used to decide whether the high-bit cell works. The sub-sense comes from the output of G2. In addition, NAND gate (G4) replaces an original NOT gate (G3), so sense amplifier does not work in cases where the flag-bit is 0. Modified sense amplifier will automatically stop work, and enforce the output to 0 to ensure the integrity of the output data when the high-bit cell is shut off. For the LWV, the modification does not affect the normal work of high-bit sense amplifier.

4.4 Assessment of Size and Delay

AVDC modifies only the high SRAM cells that belong to MBA or HBA, and the SRAM cell in LBA is not changed. In Fig. 5, replacing an inverter into a NAND gate G1 could increase the wordline's driving delay, because a NAND gate contains more transistors than an inverter. According to [13], delay is increased about 2% under the same transistor size, and this increment can be avoided if the NAND gate transistor's size is tripled without representing a significant overall area increase. It is similar for sense amplifier. So the modified circuit can maintain the same cache access delay as the original, and the same analysis is applied to G4.

We consider read and write delay overhead of AVDC, separately. For each write access, AVDC cannot generate the visible delay penalty because the VWV pattern detection is carried out before the cache access, since the value to be written is known in early stages such as decode stage, which is same as write operation of FV cache [7], [13]. For reads, similar to non-delayed FV Cache design (1-cycle FVC) [13], AVDC also uses 1-cycle latency for non-FVs as well as FVs by the flag bit gating the open/close of the high-bit array. Thus, the flag-bit is read out in parallel with all the data bits. Note that the high-bit data bits have no output if the corresponding flag-bit is reset, since its SRAM cell and sense amplifier do not work. So a NAND gate is necessary to reconstruct zeros when the flag-bit is reset, which may cause a little delay to read cycle because of its more transistors than the traditional output driver (shown in Fig. 6). But, the delay can be avoided as mentioned above, so there is no visible delay penalty for each read access.

Using the Gated-V_{dd} technique affects little on circuit size. Each pair of bit lines only allocates one sense amplifier, so this size increment can be ignored. The size increment is mainly from the flag-bit. Each word (32 bits) adds two bits flag-bit, and the size increases 1/16. Because the number of corresponding control circuit gates is as the same as the number of flag-bit, the overall increase of storage in size is 1/8 = 12.5%. We compared the size before and after circuit modification (evaluation size is 16 kB with about 60% the storage body) by CACTI3.0 [14]. The result shows that the overall increasing of size is 7.5% compared with the

conventional cache, and 3.75% compared with FVs cache.

5. Experiments

5.1 Simulation Environment

To evaluate the power and performance in 70 nm CMOS technology, we employ the HotLeakage 1.0 power/performance simulator [17], which is built upon Wattch 1.02 power/performance simulator [18], and has circuit-level accuracy for modeling the leakage current of cache- like structures. The Wattch simulator is built on the Simplescalar 3.0 simulation tool set [15] and integrates the CACTI timing, power and area models [14]. The baseline configuration we use is listed in Table 1. We compared the proposed AVDC with FV cache proposed by Zhang's [13] and the traditional cache. L1 Data Cache (L1DC) is analyzed.

5.2 Benchmark

Twelve SPEC CPU2000 benchmarks were employed (include the six SPECint and the six SPECfp benchmark). All SPEC applications use the reference inputs. In order to verify the performance, all of the benchmarks are wholly completed, and it is ensured that the number of instructions of each benchmark is more than one hundred million. We compiled the SPEC CPU2000 benchmarks for the Alpha 21264/Unix using gcc-2.7.2 compiler and link. The statistical information of benchmarks is shown in Table 2.

5.3 AVDC Granularity

While a recommended architecture of AVDC has been described in Sect. 3, the AVDC can also be designed with different granularity of line architecture. In other words, there are many configurations of dividing cache array. For example, 16-bit granularity means that the cache array is divided into two sub-arrays with each sub-array being 16 bits. We conducted a study to see how various granularities achieve ideal results. Figure 7 shows the reduction in total power consumption for various granularities. Three uniform granularities are used to compare with our proposed non-uniform granularity, where 16-bit, 8-bit and 4-bit granularity means that the data array is divided into 2*16-bit, 4*8-bit and 8*4-bit, respectively. We show results for three uniform granularities (16-bit, 8-bit and 4-bit granularity) and our proposed non-uniform granularity, where all of the results includes the power consumption of the additional flag-bit. We see that the 8-bit granularities gives the greatest power savings overall in all of the uniform granularities. Usually, a large granularity decreases AVDC efficiency than a small granularity because a large granularity decreases bit-width that can be shut off when storing a small value. For example, storing a very small value (i.e. 0 or 1), AVDC can not shut off the SRAM cells from the second bit to the 15th bit at the 16-bit granularity, but can not shut off the SRAM cells from the second bit to the 7th bit at the 8-bit granularity. In addition, excessive "narrow" granularity is also fatal because each sub-array except the lowest sub-array needs one bit flag-bit to control the open or close the SRAM cell. The additional flag-bit not only produces the power consumption but also increases the complexity and size of cache. For example, a 4-bit granularity would almost double the area

Table 1 Simulation processor configuration.

parameters	value
Fetch/Decode/Issue/Commit	4 Instructions Width
Branch Direction Predictor	16 K-entry Gshare
Branch Target Buffer	512-Entry, 2-Way
LSQ Size	32
Instruction Fetch Queue Size	32
Functional Units	4 Int ALU, 2 Int mult/div, FP ALU, 2 FP mult/div, 2 MEMPORT
Branch Misprediction Penalty	6 cycles
Instruction L1 Cache	16 KB, 32 Byte Blocks, Direct Mapped, Latency: 1 cycle
Data L1 Caches	16 KB, 32 Byte Blocks, 4-way Mapped, Latency: 1 cycle
UL2 Cache	256 MKBs, 64 Byte Blocks, 4-way Mapped, Latency: 6 cycle
Memory	Ideal size, Latency: 100 cycle

Table 2 Benchmark program.

Gategory	Benchmark	Instruction count	Data Count	
			load	store
CINT2000	gap	1,169,578,056	363,225,750	119,277,734
	gcc	2016068927	670,458,019	218,411,293
	gzip	3367274041	818,223,499	249,242,827
	parser	4202270641	1,225,284,439	438,871,541
	vortex	9808194342	2,651,339,186	1,553,547,379
	vpr	1566700982	442,016,590	124,332,330
CFP200	ammp	5490987557	1,593,504,494	391,400,434
	art	1478590543	464,186,314	128,543,916
	equake	1443346165	438,727,559	111,482,808
	galgel	4339030093	1,530,989,793	320,177,717
	mesa	2880377511	630,769,378	304,329,596
	wupwise	10196124865	2,114,697,013	764,083,412

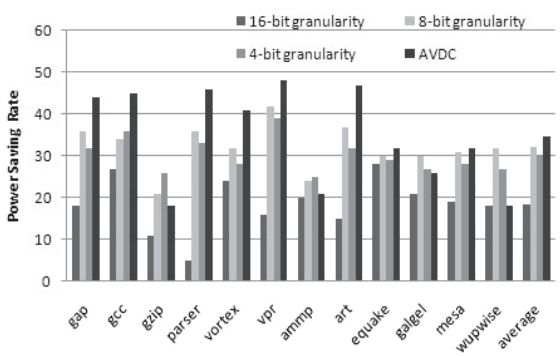

Fig. 7 Power saving for accesses when applying various sized bit fields.

overhead.

The further experimental results show the uniform granularity is not good because of the value locality. A good engineering compromise is to balance between granularity and value locality by a more effective approach where line architecture is grouped into non-uniform granularity to achieve a high value coverage rate without sacrificing efficiency of AVDC. As described in Sect. 3, the values that only need one fourth of word and half of word occupy about 82% of all values. Thus, two upper granularities should be merged to reduce the size and to increase the efficiency of AVDC. Meanwhile, we discovered that values less than 4 bits occupy the lager proportion in the range of values less than 8 bits. Figure 8 shows that Value Access Coverage (VAC) rate and Value Storage Coverage (VSC) rate increase following with bit-width in the L1 Date Cache, where the results are the average value of all the benchmarks. It is obvious that two curves are approximate, where data width increases after certain degree, the coverage rate enhancement became slow. It is because that VAC and VSC already achieve a high coverage rate when the data width is small. The results show that VAC and VSC is equal to 42.21% and 40.35% respectively when the data width is equal to 4. So the first granularity is 4 bits instead of 8 bits to achieve higher efficiency. Figure 7 also shows that the power saving of non-uniform granularity is better than other uniform granularity. Therefore, the architecture of AVDC that is divided into three sub-array (the values less than 4 bits stored in LBA about occupy 40%, the value less than 16 bits stored in LBA+MBA also about occupy 80%, using all of the arrays is only less than 20%) is feasible.

5.4 Power Saving

We discuss the power saving from two respects, dynamic power consumption and static power consumption.

5.4.1 Dynamic Power Saving

In the AVDC design, the energy consumption can be separated into two major components. First, there is a fixed cost that all accesses must incur regardless of value patterns, caused by the peripheral circuitry such as decoder, tag bitlines and data pattern detection. The second component arises in computing the data array energy due to the wordline length and number of bitlines driven such as data bitlines and sense amplifier, which vary according to the value pattern in AVDC. It is well known that the second component is the biggest power consumption contributor in AVDC like as that in the traditional cache. Most energy is dissipated in the bitlines which are areas where we expect to obtain power savings through preventing high-bit array of the SM-WV from accessing.

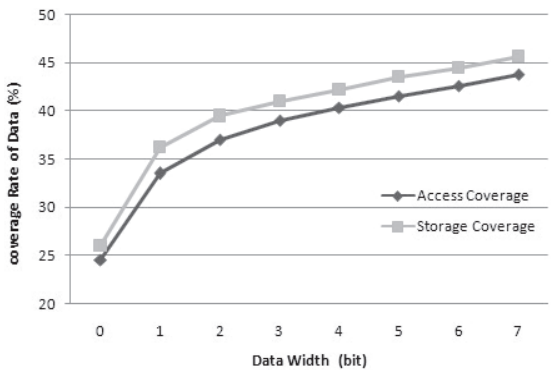

Fig. 8 Data width contribution to VAC and VSC.

The most major dynamic power consumption that arises due to the bitlines and sense amplifier is a function of the wordline length. Thus, the dynamic power consumption can be reduced because AVDC can prevent high-bit arrays of SM-WV from accessing. Contrarily, the dynamic power consumption is increased due to the extra two bits of the flag-bit when LWV is accessed. On the other words, AVDC represents tradeoff between lower dynamic energy consumption for SM-WV accesses and higher dynamic energy consumption for LWV accesses. The tradeoff depends on the access coverage of each value pattern and the energy consumption of each value pattern. In the Sect. 3, we have presents the SM-WV occupies a large proportion of data access. Here, we obtained the energy consumption of AVDC from HSpice simulations from extracted layout. Table 3 shows the energy consumption for basic wordline (32 bits) and each value patterns in AVDC. Comparing with the basic wordline, reading a SWV and a MWV can obviously reduce the energy consumption about 46% and 17%, respectively. Writing a SWV and MWV can reduce the energy consumption about 64% and 34%. But the energy consumption increase about 6.2% for reading a LWV and 4.4% for wring a LWV. Both results (access coverage and the energy consumption of each value pattern) show that the energy reduction due to accessing SM-WV is much larger than the energy increment due to accessing LWV.

Furthermore, the increment in power consumption arises due to VWVP-D that must be carried out during write operations. Data patterns must be detected every write access since the information of the data is not known a prior. Fortunately, the power consumption for VWV pattern detection is small because of its simple logic. In fact, the VWVP-D circuit gives an overall power consumption of under 7% on average, comparing the results of Fig. 9 and Fig. 10. Summarizing, the power saving due to reducing the wordline length is much large, and power increment due to the flag-bit logics and the VWVP-D is neglected.

We employ XCACTI 3.0 [14] to measure the cache power consumption, and implement AVDC model on

Table 3 Energy consumption for each value pattern in the AVDC design. (pJ)

Operation	Basic Wordline	LWV	MWV	SWV
Read	45.6	48.43	37.83	24.64
Write	107.6	112.33	71.5	39.26

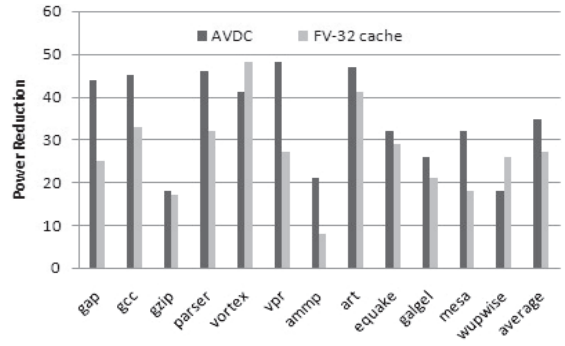

Fig. 9 Dynamic power saving.

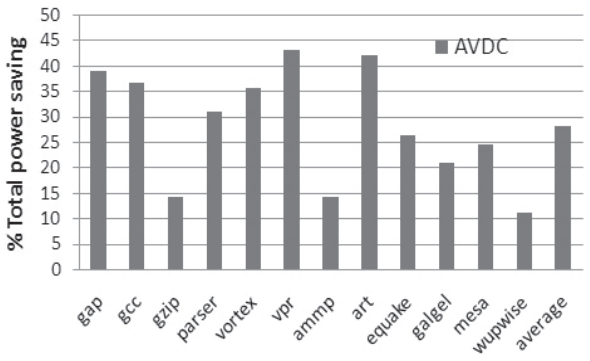

Fig. 10 Total power saving compared with the traditional data cache.

XCACTI. The variables Ndwl and Ntwl used in XCACTI are set to 1 because our scheme does not support the column line to cut apart. FV cache proposed by Zhang [13] is well done in low dynamic-power and low static-power. So we also modified XCACTI 3.0 to incorporate a model of the FV cache design to compare with AVDC. To simplify comparison and modeling, the FV finder and the encoder were simulated with the SRAM register file. The power consumption of CAM memory cells and corresponding combinational logic overhead are excluded from FV cache. Meanwhile, the power consumption of the VWVP-D is also excluded from the AVDC for fair comparison.

Figure 9 shows the power saving of AVDC and FV-32 cache. The power saving rate of some benchmarks with high access coverage like parser, vpr, and art is more than 45%. However, for benchmarks with low access coverage, the power saving rate is also near 15%. Thus, the power saving becomes larger following the value coverage increment. Figure 9 illustrates that AVDC reduced the dynamic power consumption by 34.83% of the data cache, and Zhang's FV-32 cache reduced 27.08% on average. The main reason of the above results is that the access coverage of the AVDC is higher than that of Zhang's FV-32 cache.

Finally, the overall power consumption is also simulated. Figure 10 presents the overall power saving of the AVDC. The power consumption compared with the traditional data cache reduces about 28.2%, on average. Although the power reduction is less than the results shown in Fig. 9 (under 7%, on average) because of the VWVP-D circuit costs per writing access, the proposed AVDC still outperforms the traditional cache and the FV cache.

5.4.2 Static Power Saving

AVDC can reduce the static power consumption. As mentioned in Sect. 4, the overall static power consumption saving depends on the coverage of VWV pattern in data cache. Through Fig. 1, we found that there is abundant VWV in the L1 data cache for SPEC 2000 benchmark. On average, 82% of the total values are the SM-WV, the highest is 98% for benchmark 'parser' and the lowest 67% for benchmark 'galgel'. The static power saving is proportional to the number of bit-width that can be shut off. The calculation formula of the percentage of shutoff unit (σ) is given by Eq. (3):

$$\sigma = SWV\% \times 28/34 + MWV\% \times 16/34 \quad (3)$$

Equation (3) shows the value that is SWV can turn off 28 bit unused high-bit cell, the value that is SM-WV can turn off 16 bit unused high-bit cell. We also need 2 bits flag-bit per 32 bit word. So the result of above formula equals 53.76%. Gated-V_{dd} technique using pMOS can reduce the static power consumption to 86%, so the static power saving using AVDC are 46.23% (53.76% × 86%) on average. Comparing with the conventional 32-bit per word cache, the static power saving can be calculated as 100% − (100% − 46.23%) × 34/32 = 42.87%. The static power reduction by Zhang's FV-32 cache is about 33%. So the proposed AVDC can reduce the static power more than FV cache.

6. Conclusion

We proposed the Adaptive Various-width Data Cache for reducing the power consumption of a data cache memory, which is predicated on the observation that many cached values are narrow-width values. AVDC can reduce both the dynamic and static power consumption without increasing cache access. Different from the traditional FV cache technique, our approach is applicable not only to specific instruction set processor but also to general purpose processor, because it does not need to find the frequent value dedicated for each program, and the narrow-width value are frequently used in a program. Therefore, ADVC can access in one cycle for all values. Experimental results show that AVDC achieved 34.83% dynamic power reduction and 42.87% static power reduction on average compared with the cache without AVDC, each improved by 7.75% and 9.87% compared with the FV cache respectively. Furthermore, AVDC adds only two bits based on the conventional cache and one bits more than FV cache, so area increment is very little.

Acknowledgments

We would like to thank the anonymous reviewers for invaluable and helpful comments on this paper. This research was

supported by CREST, JST and partially by a grant of Regional Innovation Cluster Program 2nd stage, MEXT.

References

[1] J.H. Edmondson, P.I. Rubinfeld, et al., "Internal organization of the Alpha 21164, a 300 MHz 64-bit quad-issue COMS RISC microprocessor," Digital Technical Journal, pp.119–135, July 1995.
[2] J. Montenaro, et al., "A 160 MHz 32 bit 0.5 W CMOS RISC microprocessor," Int'l Solid-State Circuits, vol.31, pp.1703–1714, Nov. 1996.
[3] D.H. Albonesi, "Selective cache ways: On-demand cache resource allocation," J. Instruction Level Parallelism, p.1, May 2000.
[4] K. Ghose and M.B. Kamble, "Reducing power in superscalar processor caches using subbanking, multiple line buffers, and bit line segmentation," IEEE/ACM Int. Symp. on Low Power Electronics and Design, pp.70–75, 1999.
[5] N.S. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Drowsy instruction caches, leakage power reduction using dynamic voltage scaling and cache sub-bank prediction," Int. Symp. on Microarchitecture, pp.219–230, Nov. 2002.
[6] S.Y. Yang, M.D. Powell, B. Falsafi, and T.N. Vijaykumar, "Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay," 8th Int'l Symp. on High-Performance Computer Architecture, pp.151–161, Boston, Massachusettes, 2002.
[7] J. Yang and R. Gupta, "Energy efficient frequent value data cache design," Int'l Symp. on Microarchitecture, pp.197–207, Istanbul, Turkey, 2002.
[8] L. Li, I. Kadayif, Y.F. Tsai, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, and A. Sivasubramaniam, "Leakage energy management in cache hierarchies," 11th Int'l Conf. on Parallel Architectures and Compilation Techniques, pp.131–140, Charlottesville, VA, 2002.
[9] K. Roy, "Leakage power reduction in low-voltage CMOS designs," IEEE Int'l Conf. on Circuits and Systems, vol.2, pp.167–173, Lisbon, Portugal, 1998.
[10] M. Powell, S.H. Yang, B. Falsafi, K. Roy, and T.N. Vijaykumar, "Gated-V_{dd}: A circuit technique to reduce leakage in deep-submicron cache memories," ACM/IEEE Int'l Symp. on Low Power Electronics and Design, pp.90–95, Rapallo, Italy, 2000.
[11] K. Nii, H. Makino, Y. Tsujihashi, C. Morishima, Y. Hayakawa, H. Nunogami, T. Arakawa, and H. Hamano, "A low power SRAM using auto-backgate-controlled MT-CMOS," Int'l Symp. on Low Power Electronics and Design, pp.293–298, Tokyo, Japan, 1998.
[12] Y. Zhang, J. Yang, and R. Gupta, "Frequent value locality and value-centric data cache design," Proc. Ninth International Conference on Architectural Support for Programming Languages and Operating Systems, pp.150–159, Nov. 2000.
[13] C. Zhang, J. Yang, and F. Vahid, "Low static-power frequent-value data caches," The Design, Automation and Test in Europe Conference and Exhibition, vol.1, pp.214–219, Paris, France, Feb. 2004.
[14] P. Shivakumar and N.P. Jouppi, "CACTI 3.0: An integrated cache timing, power, and area model," http://www.hpl.hp.com.techerports/PCompaq2DEC/WRL-2001-2.html, 2001.
[15] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," Computer, vol.35, no.2, pp.59–67, Feb. 2002.
[16] O. Ergin, D. Balkan, K. Ghose, and D. Ponomarev, "Register packing: Exploiting narrow-width operands for reducing register file pressure," Proc. 37th Annual IEEE/ACM International Symposium on Micro-architecture, pp.304–315, Portland, Oregon, Dec. 2004.
[17] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M.R. Stan, "Hotleakage: An architectural, temperature-aware model of subthreshold and gate leakage," Techical Report CS-2003-05, Department of Computer Sciences, University of Virginia, Virginia, USA, Tech. Rep. CS-2003-05, March 2003.
[18] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," Proc. 27th Annual Intl. Symp. on Computer Architecture, pp.83–94, 2000.
[19] C.L. Su and A.M. Despain, "Cache design for energy efficiency," 28th Int'l System Sciences Conf., pp.306–315, Hawaii, 1995.
[20] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting general behavior to reduce cache leakage power," Int. Symp. on Computer Architecture, pp.204–251, 2001.
[21] D. Brooks and M. Martonosi, "Dynamically exploiting narrow width operands to improve processor power and performance," Proc. 5th International Symposium on High Performance Computer Architecture, pp.13–22, Orlando, FL, USA, Jan. 1999.
[22] T. Sato and I. Arita, "Table size reduction for data value predictors by exploiting narrow width values," Proc. 14th International Conference on Supercomputing, pp.196–205, Santa Fe, New Mexico, United States, May 2000.

Jiongyao Ye was born in ShangHai, China on May, 1978. He received the B.E. degree in Electronic Engineering in 2000, from Shanghai Marine University, where he was an assistant during 2000–2002. In 2005, he received the M.S. degree in Graduate School of Information, Productions and Systems, from Waseda University. He then joined the Sony LSI Design Inc., where he worked in the field of LSI design from 2005 to 2008. He is currently working toward the Dr. Eng. degree in Graduate School of Information, Productions and Systems, from Waseda University. His research interests include micro-architecture, low-power FPGA and their applications.

Yu Wan was born in Tangshan, China on January, 1984. He received the B.E. degree in Electronics Engineering in 2007, from Xidian University. In 2007, he joined the National 863 Program in Xidian University and focused on VLSI/SOC design technology. In 2009, he received M.E. in Graduate School of Information, Production and System, from Waseda University. His current research interests are mainly on low-power LSI design technology.

Takahiro Watanabe was born in Ube, Japan on October, 1950. He received the B.E. and the M.E. in Electrical Engineering from Yamaguchi University, and the Dr. Eng. from Tohoku University. In 1979, he joined Research and Development Center of TOSHIBA Corp., where he worked in the field of LSI design automation. In August 1990, he joined Yamaguchi University, the Department of Computer Science and Systems Engineering, and in April 2003, he moved to Waseda University, Graduate School of Information, Production and Systems. His current research interests are EDA algorithm, Microprocessor and MPSoC, NoC, FPGA and their applications. He is a member of IPSJ and IEEE.