Reducing Power Consumption of Instruction ROMs by Exploiting Instruction Frequency

Inoue, Koji
Dept. of Electronics Eng.and Computer Science, Fukuoka University

Moshnyaga, Vasily G.
Dept. of Electronics Eng.and Computer Science, Fukuoka University

Murakami, Kazuaki
Department of Informatics, Kyushu University

REDUCING POWER CONSUMPTION OF INSTRUCTION ROMS BY EXPLOITING INSTRUCTION FREQUENCY

Koji Inoue†, Vasily G. Moshnyaga†, and Kazuaki Murakami‡

†Dept. of Electronics Eng.and Computer Science,
Fukuoka University
8-19-1 Nanakuma, Jonan-ku,
Fukuoka 814-0180 JAPAN
Email:{inoue, vasily}@tl.fukuoka-u.ac.jp

‡Department of Informatics,
Kyushu University
6-1 Kasuga-Koen, Kasuga,
Fukuoka 816-8580 JAPAN
Email:murakami@c.csce.kyushu-u.ac.jp

ABSTRACT

This paper proposes a new approach to reducing the power consumption of instruction ROMs for embedded systems. The power consumption of instruction ROMs strongly depends on the switching activity of bit-lines. If a read bit-value indicates '0', the precharged bit-line is discharged. In this scenario, a bit-line switching takes place and consumes power. Otherwise, the precharged bit-line level is maintained until the next access, thus no bit-line switching occurs. In our approach, the binary-patterns to be assigned to op-codes are determined based on the frequency of instructions for reducing the bit-line switching activity. Application programs are analyzed in advance, and then binary-patterns including many '1' are assigned to the most frequently referenced instructions. In our evaluation, it is observed that the proposed approach can reduce 40% of bit-line switching.

1. INTRODUCTION

Achieving low power consumption is one of the most important requirements for embedded systems, because it directly affects not only battery life but also cost, reliability, and so on. Here, we consider processor-based embedded systems which consist of a CPU core, an instruction ROM (I-ROM), a data RAM, and peripheral logics. In this kind of embedded systems, mainly there are two factors for power consumption: one is the power consumed in the CPU core and the other is that in the memory system. The later depends largely on the memory size and the memory-accesses frequency. With the recent increase in the requirements of advanced application programs, the code size which limits the I-ROM size has been increasing. In addition, since the I-ROM is accessed at every clock cycle, it has been becoming a major contributor to the overall chip power. Therefore, we focus on the power dissipated by the I-ROM.

To achieve high-speed memory accesses, each bit-line is precharged to a reference voltage. Here, it is assumed that a single bit-line scheme is employed. The address decoder activates a word-line when a memory access takes place, and the stored data corresponding to the selected memory cells appear on bit-lines. Each sense-amplifier amplifies the difference between the bit-line voltage and the reference voltage. Here, we refer '*conforming-bit-value (CBV)*' to as the same bit value as the precharging bit value. Namely, if we define the precharged bit-line value is '1', '1' is the CBV, and '0' is called '*unconforming-bit-value (UCBV)*'. Through this paper, we assume that the conforming-bit-value is '1' unless stated otherwise. When the UCBV is read out from a memory cell, the corresponding bit-line is discharged. Therefore, the power is consumed due to the bit-line switching. On the other hand, when the CBV is read, bit-line discharging does not take place. In this case, the bit-line does not consume the power because the precharged value is maintained until the next precharge operation [3].

In a large memory array, bit-lines are major contributors to the overall memory power consumption. By reducing the total number of UCBVs to be accessed, we can reduce the power consumption of the I-ROM. In [3], the authors proposed *horizontal-strip-inversion (HSI)* approach. The total number of memory cells memorizing the UCBV is counted word by word. A 1-bit flag, called invert-flag, is attached to each word data. If the total number of memory cells having the UCBV is greater than half of the word size, the whole word data is inverted and corresponding invert-flag is reset to 0. On an I-ROM access, the invert-flag corresponding to the read data is checked. If the flag is 1, the read data is inverted and provided to the processor. Figure 1 depicts the concept of the HSI approach. In the case that the CPU accesses from address-0 to

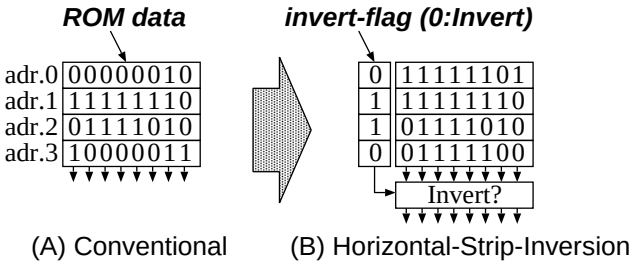

Figure 1: HSI Approach

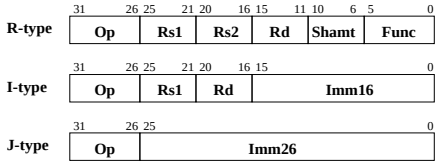

Figure 2: Instruction Format

address-3, the total count of bit-line switching (or the total number of UCBVs accessed) is reduced from 15 in conventional organization to 10 in the HSI approach.

It is well known that there are hot-spots, where are parts of the program code to be executed frequently. If instructions in the hot-spots have CBV-rich binary-patterns, we can effectively save the bit-line switching. However, the HSI does not exploit this advantage, because it relays only on the spacial exploration to determine whether each word should be inverted. In order to produce more power reduction by exploiting the temporal characteristics of program behavior, we propose a new op-code assignment approach. The binary-patterns of op-code are determined for minimizing the total number of UCBVs to be accessed. The execution count of each instruction is investigated in advance. Then, we assign CBV-rich binary-patterns to the most frequently executed instructions. By combining our temporal strategy with the spacial strategy as the HSI, we can achieve a significant power reduction for I-ROMs.

The rest of this paper is organized as follows: Section 2 shows the concept of our approach and proposes the binary-pattern assignment methodology for low power I-ROMs. Section 3 evaluates the effectiveness of the proposed method. We also consider the combination with the HSI approach. Section 4 gives some concluding remarks.

2. INSTRUCTION CORDING FOR LOW POWER I-ROMS

2.1. Concept

It is well known that there is a rule for program-execution behavior called *90/10 Locality Rule*: a program executes about 90% of its instructions in 10% of its code [4]. That is, there are some portions of program-address space executed frequently. We have analyzed the execution frequency of instructions for MPEG-decoder program by using an instruction-level dlx-simulator [6]. Figure 2 depicts the instruction format we assumed.

Table 1 shows the ranking list (top five) of the most frequently executed instructions and referenced registers when we look at each field in the instruction format. The corresponding binary-patterns based on the *dlx-sim* simulator [1] [7] are also shown in the table. From the table, we see that UCBV-rich binary-patterns are assigned to the most frequently executed instructions. For example, binary-patterns "000000" and "001000" are assigned to the SPECIAL instruction (i.e., R-type instructions) and ADDI instruction, respectively. These instructions occupy more than 34% of the total instruction count. Similar situations also can be seen for register references.

As explained in Section 1, I-ROM power consumption depends largely on the total number of UCBVs to be accessed. Unfortunately, binary-patterns of op-codes in conventional CPU design are defined with the consideration of the instruction decoder's complexity. Since programs inherently include the locality of instructions, however, properly choosing the binary-patterns of op-codes is a key to reduce the number of UCBVs accessed.

In order to reduce the bit-line switching activity, we re-assign new binary-patterns to the op-codes based on the instruction frequency. The concept of our approach is quite simple. We attempt to assign CBV-rich binary-patterns to the op-codes of frequently executed instructions. We also consider the register-field. Usually, compiler attempts to use effectively the limited register resource. As a result, we see the locality of register references, as shown in Table 1. For the immediate-field, we employ the HSI approach as shown in Figure 3. We implement two invert-flags to each word data: one is for 16-bit immediate data (I-type instructions) and the other is for 26-bit immediate data (J-type instructions). Our approach requires the following steps.

[1] The binary-patterns of the dlx-sim op-codes were defined based on a MIPS machine.

Table 1: Frequency of Instructions (*mpeg_decode* using "mei16v2.m2v" input file)

Ranking	Op-Field			Func-Field			Reg-Field		
	Inst.	Freq.	Op-code (#of UCBVs)	Inst.	Freq.	Op-code (#of UCBVs)	Reg.	Freq.	Op-code (#of UCBVs)
1	Special	17.9%	000000(6)	ADD	9.0%	100000(5)	R1	26.8%	00001(4)
2	ADDI	16.5%	001000(5)	MOVI2FP	1.9%	110101(2)	R29	14.8%	11101(1)
3	LW	15.0%	100011(3)	SLT	1.9%	101010(3)	R2	12.3%	00010(4)
4	SW	12.8%	101011(2)	SUB	1.5%	100010(4)	R3	10.7%	00011(3)
5	BNEZ	4.2%	000101(4)	OR	1.3%	100101(3)	R31	6.4%	11111(0)

1. The target application is executed on an instruction-level processor-simulator by using a sample input data. Then we obtain a ranking list for the frequency of instructions and that of register references, as showed in Table 1.

2. The binary-patterns of op-codes for the op-field, the func-field, and the register-field are determined based on the ranking list. For each field, CBV-rich binary-patterns are assigned in ranking order. For example, "111111" is assigned to the op-field of the SPECIAL instruction for the MPEG program. Next, "111110" and "111101" are assigned to the op-field of the second (ADDI) and the third (LW) instructions, respectively.

3. For the immediate-field of I-type and J-type instructions, the HSI approach is applied.

4. The object code of target program is translated based on the new binary-patterns determined by the above steps.

Of course, the instruction decoder has to be modified based on the new op-codes (the effect of the instruction-decoder complexity is discussed in Section 3.4. Therefore, our technique is best suited for dedicated processors, since they are commonly used within embedded systems to execute the dedicated application or the same portion of code over and over. Figure 4 shows a part of the new binary-patterns for the MPEG-decode program.

2.2. Comparison with Other Techniques

Chang et al.[3] proposed the horizontal-strip-inversion technique explained in Section 1. In addition, they proposed another method called vertical-strip-inversion (VSI) for I-ROMs. The total number of UCBVs is counted column by column, and all column data are inverted if the result is greater than half of the column size.

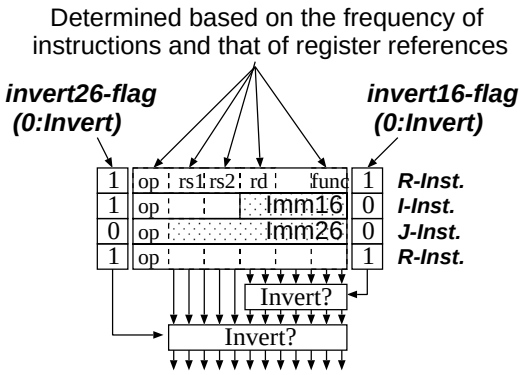

Figure 3: Concept of Proposed Approach

Benini et al. [2] proposed instruction set encoding techniques for low power consumption. Clever binary-pattern assignment to op-codes reduces the switching activity of instruction fetch and decode logic. A variety of coding techniques for reducing buses and I/O power have been proposed [5][1]. The data to be transferred between the memory and the CPU core is coded in order to reduce bus switching activity. On the other hand, the purpose of our approach is to reduce the power consumed by the I-ROM. The power consumption of instruction-fetch logic, decoder, and buses depends on the total number of '0-1-0' or '1-0-1' bit transitions, so that the relationship between successively executed instructions has to be considered. However, as explained in Section 1, the power consumption of I-ROM employing a single bit-line scheme depends on the total number of UCBVs accessed. Therefore, we need to focus not on the instruction sequence but on the frequency of instructions.

Op-Field

upper \ lower	000	001	010	011	100	101	110	111
000	SP	FP	J	JAL	BEQZ	BNEZ	BFPT *BFPT*	BFPF *SLTUI*
001	ADDI	ADDUI *MULI*	SUBI *BFPF*	SUBUI *LHU*	ANDI *SF*	ORI *DIVPI*	XORI *XORI*	LHI *NOP*
010	RFE	TRAP *BREAK*	JR *SGEUI*	JALR *MODUI*	*MODI*	*MULUI*	MULI *SGEI*	MULUI *SNEI*
011	SEQI *SEQUI*	SNEI *SLEUI*	SLTI *ORI*	SGTI *SLLI*	SLEI *ANDI*	SGEI *JAL*	DIVPI *SLEI*	DIVPUI *LH*
100	LB	LH *RFE*	*SD*	LW *DIVPUI*	LBU *SNEUI*	LHU *JALR*	LF *SLTI*	LD *FP*
101	SB *LD*	SH *TRAP*	*SGTI*	SW *SRAI*	*SEQI*	*JR*	SF *SB*	SD *LBU*
110	*LF*	*SUBI*	*SGTUI*	BREAK *SUBUI*	MODI *J*	MODUI *LB*	SRLI *SH*	SRAI *BNEZ*
111	SEQUI *SRLI*	SNEUI *BEQZ*	SLTUI *ADDUI*	SGTUI *SW*	SLEUI *LHI*	SGEUI *LW*	SLLI *ADDI*	NOP *SP*

Register-Field

11111	R31 *R1*
11110	R30 *R29*
11101	R29 *R2*
11011	R27 *R3*
10111	R23 *R31*
01111	R15 *R30*
11100	R28 *R4*
●	
●	
00000	R0 *R25*

Base ***Code for MPEG***

Figure 4: Op-Code Assignment for MPEG decoder

3. EVALUATION

3.1. Simulation Environment

There are several components which consume the power on I-ROM accesses, address decoder, memory array, sense amplifier, buses between the I-ROM and the microprocessor, and so on. How much each component affects the total memory power depends on implementation. In this paper, we focus on the power consumed in the memory array, and assume that it depends on the bit-line switching activity. Power consumption can be expressed by the following equation.

$$P = SW \times C \times F \times V^2, \quad (1)$$

where SW is the switching activity of bit-lines, C is the total load capacitance of bit-lines, F is the operation frequency, and V is the voltage swing. Our approach does not affect the parameters C, F, and V, so that we assume they are constant values. We have used two integer programs from the SPEC95 benchmark suite (*099.go* and *129.compress* with train input data) and two media programs from the Mediabench (*ADPCM decoder* and *MPEG2 decoder*) as benchmark programs. In this evaluation, we compare the power consumption of the following models.

- BASE: This is a base model. The dlx-sim op-codes are assumed. No low-power technique is employed except that all data are inverted (not word by word as the HSI approach). Therefore, sense amplifiers invert the read data, and output it to the microprocessor. We have found in our simulation that this inversion brings better results than non-inverted conventional model for all benchmark programs.
- HSI (Horizontal Strip Inversion): This is a model proposed in [3] (explained in Section 1).
- CODEsp: This is a model proposed in Section 2. The binary-patterns assigned to the op-codes are optimized for each program.
- CODEall: This is the same model as the CODEsp except that the binary-patterns are determined based on total simulation results: the ranking list of instruction frequency for all benchmark programs are merged, and the binary-patterns are assigned. Namely, the binary-patterns of op-codes are optimized for all programs.

Of course, in our simulation, the bit-line switching of the invert-flags for the HSI approach is included. In addition, we have assumed that the unused fields in instructions (e.g., the shumt-field of R-type instructions except for shift instructions) are filled by CBVs (no power consumption).

3.2. Simulation Results

Figure 5 shows the power consumption of each model and its breakdown for each field in the instruction format. All results are normalized to the power consumption of the base model.

We can see from the figure that the HSI approach can make about 10% power reduction in the best case *129.compress*. In this simulation, we assumed that the invert-flag is set to 0 (i.e., UCBV) when the corresponding data is inverted. If the total count of UCBV invert-flags is greater than the half of the total I-ROM access count, we can reduce the power overhead by changing the active level of the invert-flags (i.e., '1' of the invert-flag means that the corresponding data has been inverted). If we can completely eliminate the power dissipated by the invert-flags, the HSI approach reduces the power consumption by from 6% to 18%.

On the other hand, the CODEsp, which has the optimized op-codes for each target program, can achieve

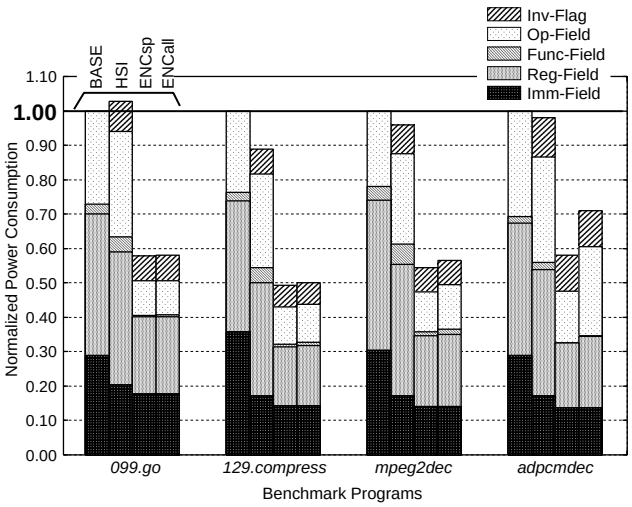

Figure 5: Power consumption of bit-lines

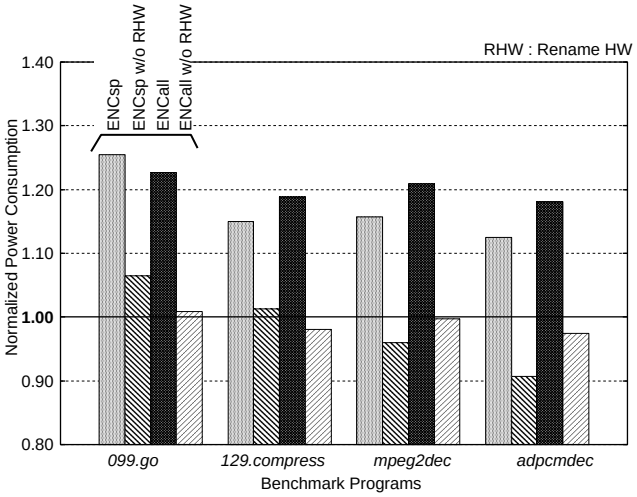

Figure 6: Power consumption of instruction decoder

more than 40% of power reduction for all programs. The new binary-pattern assignment scheme makes a significant power reduction due to large amount of locality of executed instructions and referenced registers. In addition, our approach applies the HSI technique not to the whole word data but only to the immediate-field. This fine-grain inversion control effectively reduces the power consumed in the immediate-field. Although the proposed approach includes two invert-flags per word, only one invert-flag is assigned to the UCBV in the worst case. Because one of the invert-flags is for 16-bit immediate data (I-type instructions) and the other is for 26-bit immediate data (J-type instructions). Therefore, the power overhead caused by the invert-flags is comparable with that of the HSI approach. For all but one (*adpcm_dec*), the CODEall, which has the op-codes optimized for all programs, makes power reduction as well as the CODEsp. This is because that we used a dlx compiler (*gcc-dlx*) and all programs were compiled with the same optimization policy. Therefore, the similar trends in the frequency of instructions is observed through all programs.

In addition, we have simulated the three benchmark programs (*099.go*, *129.compress*, and *mpeg_decode*) with other input data to evaluate the availability of our approach. Note that the binary-patterns of op-codes are determined based on the base input data. As a result, it was observed that our approach can achieve the power reduction as well as the results reported above (from 42% to 60% of power reduction is achieved). We can consider that the frequency of instructions may not be largely affected by the difference of input data.

3.3. Switching Activity of Buses

In the previous section, we have shown that optimizing the binary-patterns assigned to op-codes can reduce the power consumption of I-ROMs. However, it may increase the power consumed by buses between the I-ROM and the microprocessor, because our approach does not take account of instruction sequence. In order to clarify the effect of our approach to the bus power, we also measured the total switching count of the buses. As a result, it was observed that our approach brings from 8 % to 18 % reduction of bus switching activity. Based on the *90/10 Locality Rule* explained in Section 2.1, we can also consider that the most frequently executed instructions may be referenced successively. In other words, in our approach, the instructions having CBV-rich binary-patterns are executed consecutively. Therefore, the possibility of reducing the bus switching activity is increased.

3.4. Complexity of Instruction Decoder

The main drawback of our approach is the complexity of instruction decoder, resulting in more power consumption. In order to evaluate the negative effect, we designed a MIPS-base microprocessor by using HDL (Hardware Description Language) at RT level, and measured the power consumption of the instruction decoder. We used a 0.35 um CMOS technology and Synopsys Design Compiler tool provided by the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo.

The dynamic power in CMOS logic is consumed

when a charged wire (including connected gate) is discharged, as expressed by the equation (1). We obtained the switching activity of the instruction decoder, SW, from the simulation results reported in Section 3.3, which is the switching activity of instructions to be decoded. In addition, we assumed that all wires have the same value of load capacitance, C_{wire_ave}. Thus, we can approximate the total load capacitance, C, by the multiplication of C_{wire_ave} and the total number of wires in our design. The clock frequency F and supply voltage V were assumed as 300 MHz and 3.3 V, respectively.

The bars labeled as 'CODEsp' and 'CODEall' in Figure 6 depict the power consumption of instruction decoder for each program. All results are normalized to the power of BASE model. For all programs, the proposed approach increases the power consumption of instruction decoder by from 15% to 25%. This overhead may not be acceptable, because the instruction decoder is activated at every clock cycle. In our approach, the binary-patterns for source/destination register field are also optimized. Therefore, the instruction decoder has to translate the coded register number to the physical number.

Although the register-address translation produces large hardware overhead, we can eliminate it at compile time. The compiler allocate the register resources having a CBV-rich register-number (e.g., "11111") to the most frequently referenced values. In this case, the register-rename hardware is not required because the binary-patterns of register numbers are not encoded. The bars denoted as 'CODEsp w/o RHW' and 'CODEall w/o RHW' in Figure 6 show the power consumption without the renaming hardware. It is observed that the power overhead caused by the instruction decoder is trivial for all but one (*099.go*). Even in the worst case, the overhead is only 6.5%. In addition, we see that the power consumption can be reduced in some cases (*129.compress*, *mpeg2dec*, *adpcmdec*), though the hardware complexity is increased. This is because that our binary-pattern assignment reduces the switching activity of instructions as reported in Section 3.3. Thus, if the switching-activity reduction is larger than the power overhead caused by the logic complexity, the total power consumption of instruction decoder is reduced.

4. CONCLUSIONS

In this paper, an instruction encoding approach for reducing I-ROM power consumption has been proposed. The target program of processor-based embedded systems is analyzed in advance by using instruction-level simulator. We generate a ranking list for the frequency of instructions and that of register references. Based on the generated list, we assign new binary-patterns to op-codes in order to reduce bit-line switching activity in the I-ROM.

In our simulation, it has been observed that we can achieve more than 40% power reduction of the I-ROM for all benchmark programs. In our evaluation, the power consumed in address-decoder logic, sense amplifiers, and word lines were not considered. Our on going work is to evaluate the effectiveness of our approach by using a more accurate power model.

Acknowledgments

We thank Hiroto Yasuura who gave us advice. This research was supported in part by the Grant-in-Aid for Creative Basic Research, 14GS0218, for Scientific Research (A), 12358002, 13308015, and for Encouragement of Young Scientists (A), 14702064.

REFERENCES

[1] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems," *Proc. in the Grate Lakes Symposium on VLSI*, pp. 77–82, Mar. 1997.

[2] L. Benini, G. De Micheli, A. Macii, Enrico Macii, and M. Poncino, "Reducing Power Consumption of Dedicated Processors Through Instruction Set Encoding," *Proc. in the Grate Lakes Symposium on VLSI*, Feb. 1998.

[3] Y. Chang, B. Park and C.Kyung, "Conforming Inverted Data Store for Low Power Memory," *Proc. of International Symposium on Low Power Electronics and Design*, pp.91–93, Aug. 1999.

[4] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach," *Morgan Kaufmann Publishers, Inc*, 1990.

[5] M. R. Stan and W.P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, vol. 3., no. 1, Mar. 1995.

[6] University of Minnesota Computer Engineering Research Group, *URL: http://www-mount.ee.umn.edu/ okeefe/mcerg/fast-dlx/*

[7] "DLXsim - A Simulator for DLX," *URL: http://hardy.ocs.mq.edu.au/mpce _courses/dlxsim/report.html*

[8] SPEC (Standard Performance Evaluation Corporation), *URL: http://www.specbench.org/osg/cpu95.*

[9] Mediabench, *URL: http://www.cs.ucla.edu/ leec/mediabench/.*