The Energy Efficiency of 8-bit Low-power Microcontrollers

Mark H. Jones
School of Science & Engineering
University of Waikato
Hamilton, NEW ZEALAND.
Email: mjones1984@gmail.com

Jonathan B. Scott
School of Science & Engineering
University of Waikato
Hamilton, NEW ZEALAND.
Email: scottj@waikato.ac.nz

***Abstract*—We have measured the energy cost of processing, sleeping, non-volatile memory writes and ADC measurements of six 8-bit microprocessors from three manufacturers. These measurements compare the chips directly to one another and reveal ideal operating points which can be used to reduce energy consumption.**

I. Introduction

Moving to a new family of microprocessor (MCU) can be a daunting task for an electronic engineer. Making comparisons between families of processors is difficult because of the differences in architecture and the types of available data. Therefore most engineers tend to favour one family of processor simply because it's what they know or understand. We aim to compare a small set of 8-bit microprocessors based on energy consumption and computational efficiency. The MCUs that will be investigated are shown in Table I. This investigation has been carried out in order to estimate the energy requirements of a low power digital water meter with a wireless transmitter to determine the viability of harvesting energy from a domestic water supply. The authors have investigated operational aspects that are deemed relevant for water consumption metering such as processing, analog-to-digital converter (ADC) measurements, non-volatile memory writes and sleeping. Access to an Agilent E5270B Precision Measurement Mainframe provided current measurements with a $1\,fA$ resolution and the ability to sweep voltages while providing accurate voltage output.

Microchip	PIC12F675
	PIC16F688
	PIC16F1827
Atmel	ATtiny13V
	ATtiny25V
Freescale	MC9S08QG8

TABLE I
Manufacturers and models of compared MCUs.

II. Measurements

All measurements, except sleep current, were carried out on bread-boards using dual in-line packaged MCUs. Sleep currents were measured with chips placed in a chip carrier to allow for easy washing and drying in isopropyl alcohol before measurements were made. During measurements each MCUs had its pins configured as digital outputs, outputting high, and tied to V_{dd} via $10\,k\Omega$ external resistors. Microchips placed on breadboards were decoupled with $47\,nF$ capacitors. Programs where written in C and built using compilers supplied with each of the development kits (Microchip's MPLAB, Atmel's AVRStudio 4 and Freescale's CodeWarrior).

A. Processing

Measuring or predicting the amount of energy required to process information is complex. The operations each chip carries out internally can be different from one another yet produce the same result. The differences can be the result of programming style, compiler optimisation [1], [2], memory usage [3] and what instructions are supported by each MCU. Per instruction energy consumption is also affected by other factors such as the state of the processors internal circuitry [4], the operands of a given function and neighbouring instructions [5]. Source code optimisation such as loop unrolling and appropriate selection of variable types can lower energy usage, which may be applied by the programmer [6]. Test programs used in this investigation have been kept as short as possible with a minimum of code changes between MCUs to reduce variation.

1) Processing performance: To assess how efficiently each MCU executes code, each ran a benchmarking function for which the total number of instruction cycles taken to complete the benchmark was calculated. The use of MiBench, a set of benchmarking routines targeted at embedded processors [7], was investigated but most of the provided routines couldn't be executed due to resource constraints. Instead, the benchmarking function used was a 16-bit pseudo-random number generator implemented using a 16-bit linear-feedback shift register as found in [8]. The routine generates 16-bit numbers, 1 through 65536 (in a pseudo-random order), and ends once the initial seed number is regenerated. The number of instruction cycles each chip took was calculated using equation 1, where n is the number of instruction cycles, T is the benchmark period and f_i is the instruction clock frequency.

$$n = T \times f_i \tag{1}$$

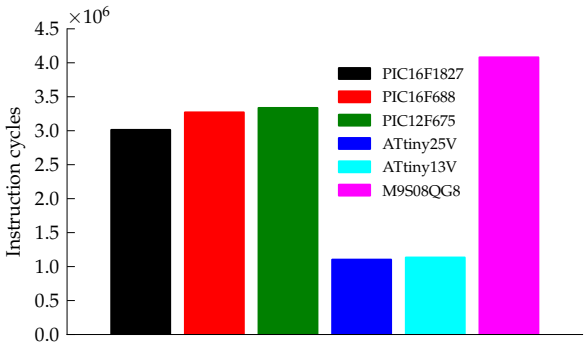

Fig. 1. Number of instruction cycles taken to complete a benchmarking function.

The results of the benchmark routine are shown in Figure 1. As shown, the Atmel MCUs complete the benchmarking function using approximately one third of the number of instruction cycles taken by the others. These results do not represent absolute performance of the MCUs, only the efficiency in terms of processing cycles needed to complete a set piece of code. Note that since this comparison is made between the number of instruction cycles the results are independent of clock frequency.

2) Energy efficient processing: Energy efficient processing in this context refers to the efficiency of completing instruction cycles, as opposed to the power consumed whilst operating. The energy consumed per instruction cycle is computed using equation 2, where Ei is the energy consumed per instruction cycle, I is the current consumption, V_{dd} is the input voltage and f_i is the instruction cycle frequency.

$$E_i = \frac{I \times V_{dd}}{f_i} \tag{2}$$

By measuring the instructional efficiency of each MCU over both frequency and voltage, points of maximum instruction efficiency were found. Figure 2 compares each of the MCUs while operating in what was found to be their most energy efficient operating points. The energy consumption between each MCU is so similar, and with so much overlap between not only models but manufacturers, that one could conclude that these results are representative more of the limitations of microprocessor fabrication technology than processor design.

The general rule for finding the most efficient operating point tended to be to clock at the fastest frequency that allows the absolute minimum input voltage and then operate at that minimum voltage. For example, Figure 3 shows the safe operating areas of the Atmel ATtiny13 for frequencies below 10 MHz. Operating the MCU at 4 MHz with a voltage of 1.8V produces the most efficient instruction cycle execution. To show how instructional frequency affects instructional efficiency, Figure 4 has been included to show the relative costs in joules per instruction cycle of running the Atmel ATtiny13V at internally generated clock frequencies. The results seen here where typical across most of the measured MCUs.

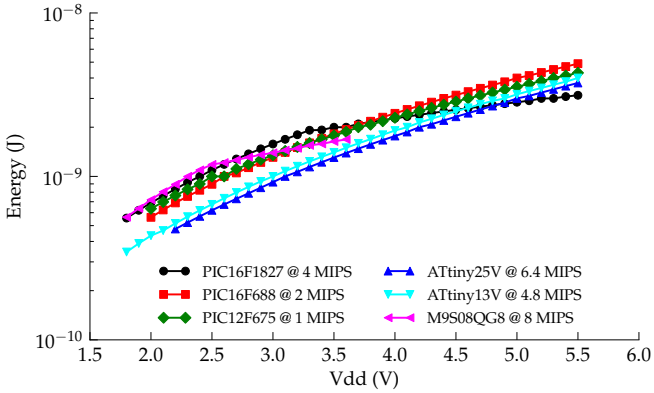

Fig. 2. Energy consumption per instruction cycle of each tested MCU at their optimum operation frequencies.

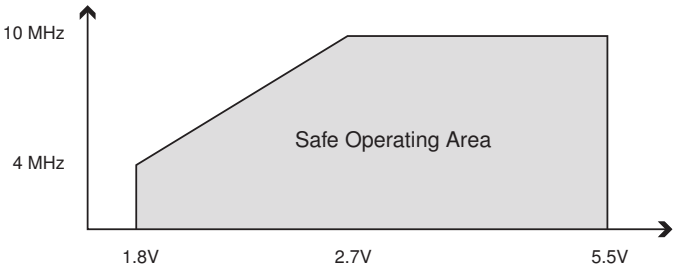

Fig. 3. Safe operating area for the Atmel ATtiny13V as given shown in [9]

B. Sleeping

Each MCU supports a sleep or stop mode which allows the clock to be disabled thereby halting any processing and conserving energy. These states can be entered by issuing a software command, placing the chip into a dormant but memory retentive state. Waking the processor can be done with either a low power timer or an external interrupt, each of which consume additional energy and have been disabled for these measurements. Keeping MCUs in sleep mode whenever possible can significantly reduce energy consumption and thereby extend the life of battery or power harvesting microprocessor

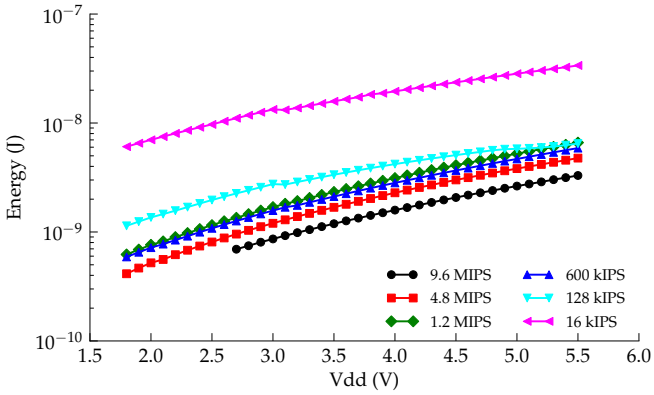

Fig. 4. Energy consumption per instruction cycle of of the ATtiny13V.

based devices. While some processors support the operation of certain peripherals in sleep mode, i.e. the Atmel and Freescale chips support operating the ADC in full or partial sleep mode respectively, these modes have not been used. Figure 5 shows the rate of energy consumption for each MCU while in their deepest sleep modes. The Microchip PIC12F675, part of Microchips lower-end family of MCU, required at least an order of magnitude less power than the other tested microprocessors. However, the Microchip PIC16F1827 featuring Microchip's eXtreme Low Power (XLP) technology was unable to achieve the specified sleep current of 30 nA as specified in [10]. Even after trying five different PIC16F1827 chips, following the manufacturers recommendations, rewriting code in assembler, manually setting relevant initialisation registers and washing the chips with isopropyl alcohol, similar results where obtained.

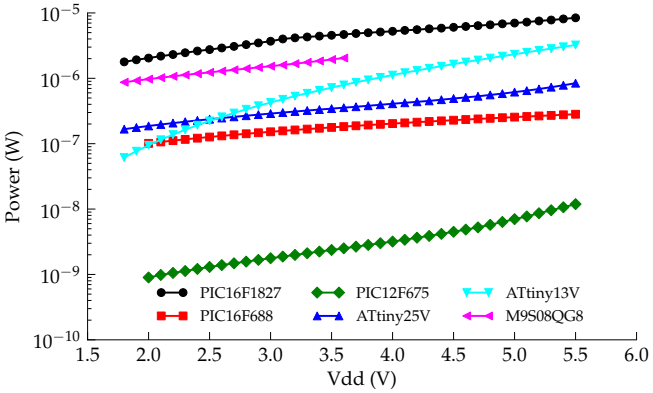

Fig. 5. Power usage of each MCU whilst in their deepest sleep mode.

C. Non-volatile memory writes

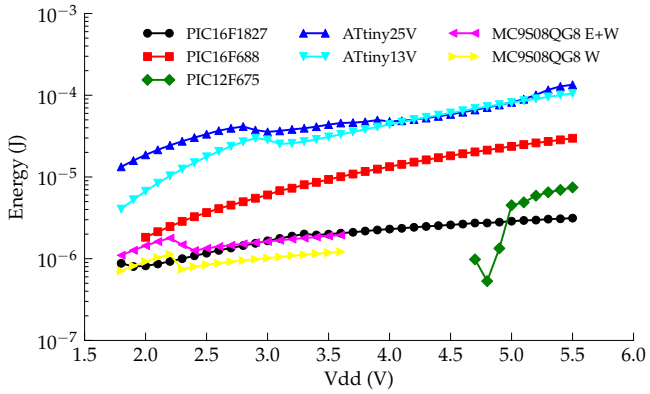

Fig. 6. Energy consumed by each MCU when writing one byte to non-volatile memory.

EEPROM by design requires a relatively large amount of power to erase and set bits. This is due to the need to create a high voltage pulse onto a terminal of a floating gate transistor (which represents a single bit) in order to deliver electrons onto or remove electrons from that gate (setting it to a one or a zero). The process of writing to EEPROM therefore requires generating large voltages internally, done by the use of on-chip charge pumps. This is also true of flash memory, which is based upon EEPROM technology.

Figure 6 shows the energy consumed by each MCU per byte written to non-volatile memory. In the case of the Freescale MC9S08QG8, which has flash instead of EEPROM, there are two traces shown. Since flash memory must be erased in blocks (pages), the trace labelled 'MC9S08QG8 E+W' has been calculated by taking the total energy cost of erasing a page (512 bytes), dividing that by 512 and adding it to the amount of energy required to write one byte. The trace labelled 'MC9S08QG8 W' represents the amount of energy used to write one byte assuming the destination byte has already been erased, a requirement when writing to flash memory. All other traces represent the energy cost of erasing and writing one byte to EEPROM memory. The Microchip PIC12F675 showed erratic behaviour while writing to EEPROM. It consistently used less energy whilst performing writes (below a V_{dd} of 4.7 V) than the stand-by routine (incrementing a number). The authors are of the opinion that something has been overlooked, however since further investigation and repeating the measurement produced the same result it has been included and the cause remains undetermined.

D. Analog-to-digital conversion

ADCs are commonly used in sensing applications as they allow a microprocessor to measure and record events or parameters of the analogue world. In the case of smart water metering, ADC measurements will most likely play a role in determining water consumption and therefore may be used extensively in this application.

The power consumption per ADC measurement was measured for each of the chips and is shown in Figure 7. These measurements represent the total amount of energy to enable the ADC, make a measurement and return the ADC peripheral to its disabled state again. The results show that the two Atmel chips used considerably less energy per measurement than the other MCUs.

III. Disscussion

The measured data shows large variations in the energy consumption between chips while sleeping, performing ADC measurements and non-volatile memory writes. Measurement data indicates that the energy required to conduct an ADC measurement and store it in non-volatile memory (to prevent measurement data-loss) depends heavily on the cost of non-volatile memory writes. Although the Atmel chips were the most energy efficient when conducting ADC measurements, their reduced efficiency when writing to non-volatile memory would far outweigh any ADC gains in applications where it is necessary to store those measurements in non-volatile memory (such as a power harvesting smart meter).

In applications where measurement data needs to be wirelessly transmitted, storing data in non-volatile memory greatly

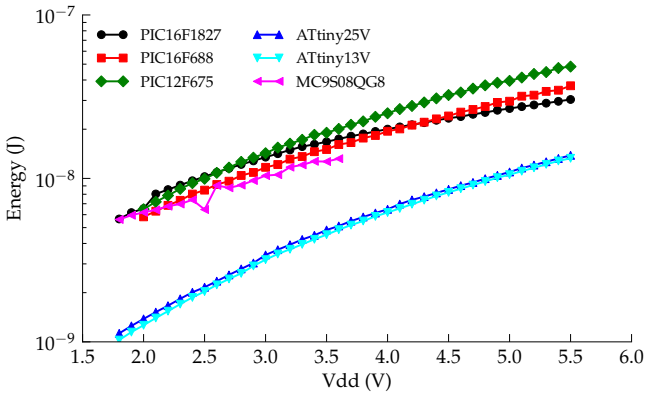

Fig. 7. Energy consumption for a single ADC conversion for each MCU, including enabling and disabling the ADC peripheral.

improves power efficiency. [11] has shown that logging data as opposed to constantly transmitting it can reduce power consumption from 45 mW to 5 mW (approximate), a reduction of 89%.

IV. Conclusion

Relative to on-chip functions, writing to EEPROM is very expensive in terms of energy consumption. The use of flash memory in place of EEPROM in this investigation proved efficient, although there are implementations of EEPROM (such as on the PIC16F1827) that can match the energy consumption of flash. There appears to be a general rule to follow when finding the most efficient processing operating point. Once that optimum operating point has been found there is negligible difference, in terms of energy consumed per instruction cycle, between MCU families and models. Instruction cycle speed does not wholly determine code execution speed as was shown by the Atmel family of MCUs. The amount of power consumed whilst sleeping varies significantly between chips, which is especially important for low power sensing and metering applications. The energy consumption of an ADC measurement is negligible, even when considering enabling and disabling the ADC module. It is likely that the processor will consume more energy processing the result than it took to make the measurement. For the energy cost of writing a single byte to non-volatile memory write a typical MCU could complete over one hundred thousand instruction cycles, sleep for just under a minute or perform over five thousand ADC measurements.

References

[1] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, "An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations," in *Int. Wkshp Power & Timing Modeling, Optimization & Simulation (PATMOS)*, 2001.

[2] O. Zendra, "Memory and compiler optimizations for low-power and -energy," *Implementation Compilation Optimization of ObjectOriented Languages Programs and Systems ICOOOLPS2006*, p. 8, 2006.

[3] J. Zambreno, M. Kandemir, and A. Choudhary, "Enhancing Compiler Techniques for Memory Energy Optimizations," in *ProcSecond IntWorkshop Embedded Software EMSOFT02*, 2002.

[4] S. Nikolaidis and T. Laopoulos, "Instruction-level power consumption estimation embedded processors low-power applications," *Proceedings of the International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. IDAACS'2001 (Cat. No.01EX510)*, no. July, pp. 139–142, 2001.

[5] V. Konstantakos, K. Kosmatopoulos, S. Nikolaidis, and T. Laopoulos, "Measurement of Power Consumption in Digital Systems," *2005 IEEE Instrumentationand Measurement Technology Conference Proceedings*, vol. 1, no. May, pp. 37–42, 2005.

[6] D. a. Ortiz and N. G. Santiago, "Impact of source code optimizations on power consumption of embedded systems," *2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference*, pp. 133–136, June 2008.

[7] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench : A free , commercially representative embedded benchmark suite The University of Michigan Electrical Engineering and Computer Science," *Electrical Engineering*.

[8] Wikipedia, "Linear Feedback Shift Register." Retrieved: 15th August 2011. Available: http://www.en.wikipedia.org.

[9] Atmel Corporation, "ATtiny13." Technical Datasheet, 2010. Available: http://www.atmel.com.

[10] Microchip Technology Inc., "PIC16F/LF1826/27." Technical Datasheet, 2010. Available: http://www.microchip.com.

[11] S. W. Arms, "Power management for energy harvesting wireless sensors," *Proceedings of SPIE*, vol. 5763, no. March, pp. 267–275, 2005.