

White Paper

The right microcontroller for low-power applications

The 8051 has been a popular choice for use in ASIC designs where a small and relatively cheap microcontroller is required. 30 years on from its introduction, is it still a wise choice for low-power applications?

The 8051

The original 8051 was developed by Intel in 1980. It is an 8-bit processor, which means that its ALU (Arithmetic and Logic Unit), accumulator, registers and data bus are all 8-bits wide. The program and data memories are in separate address spaces (This is often referred to as being a Harvard architecture). The address bus is 16-bits wide, which allows up to 64kBytes of program memory and 64kBytes of data memory. Peripherals (such as timers and UARTs) are memory mapped into the data memory address space.

The 8051 has a single accumulator, which is used to store the result for the majority of the arithmetic and logical instructions. The source operands for an instruction will typically come from the accumulator and one memory location.

The 8051 supports both direct and indirect addressing for the first 256 bytes of data memory. Indirect addressing via the 16-bit data pointer register is required to access the rest of the data memory.

The original Intel part was clocked at 12MHz, with the simplest instructions taking 12 clock cycles to execute, with other more complicated instructions taking 24 or 48 clock cycles to execute.

The enhanced 8051

While there are a number of IP cores that implement the 8051 with the exact timings and memory architecture as the original Intel part, the majority of implementations available today label themselves as being enhanced 8051s. This is because they take fewer than the original 12 clock cycles per instruction. For example, Synopsys' DW8051 core takes 4 clock cycles per instruction, resulting in three times faster execution¹. There are other implementations that can execute some instructions in a single clock cycle. Further, by utilising the latest process technology, maximum clock frequency has been increased to 300MHz in a 0.13µm process².

Some implementations have also added new features, such as a second data pointer. This particular feature can greatly improve the efficiency of moving blocks of data. However, this breaks code compatibility with the original 8051, which is often one of the main reasons cited for using the 8051 architecture.

16-bit RISC architectures

The eSi-1600 is a best in the class 16-bit RISC processor IP core, developed in 2003. It has 16 16-bit general purpose registers, which can be used to store the results of instructions or as source operands or addresses. The memory architecture can be configured as either separate program and data address spaces (Harvard architecture) or as a single unified address space (von Neumann architecture), supporting up to 128kBytes of program memory and 64kBytes of data memory. AMBA APB peripherals are memory mapped into the data memory address space. A variety of addressing modes are supported, including: direct, indirect, offset, scaled indexed and pre or post increment / modify. Instructions are encoded in either 16 or 32-bits, depending upon the number and size of the operands. The instruction set can be extended with user defined instructions to optimise software

¹ <https://www.synopsys.com/dw/doc.php/ds/i/DW8051.pdf>

² <http://www.cast-inc.com/cores/r8051xc/index.shtml>

inner loops. The eSi-1600 has a 5 stage pipeline allowing very high clock frequencies to be achieved. Most instructions can be executed in a single clock cycle.

8-bit verses 16-bit

The first possible concern of using a 16-bit processor in place of an 8-bit processor is likely to be regarding a potential increase in system cost or gate count. However, the eSi-1600 has a similar gate count, and in some cases less, than the enhanced 8051 cores surveyed.

The gate count for an 8051 varies between 6k gates³ and 17k gates⁴ depending upon the vendor and how “enhanced” it is. For the eSi-1600, the gate will be between 8.5k gates and 13kgates, depending upon which options are included.

This small size is possible in part due to the very simple control logic circuitry that is required by RISC processors. Another reason why the gate count is similar is that during the process of enhancing an 8051, logic that was originally 8-bits wide and took multiple clock cycles to execute, is converted to be 16-bits wide, so that instructions can execute in a single clock cycle.

While in 1980, the width of a data bus would have a significant cost impact, due to each bit requiring a corresponding physical pin, bus width is often no longer a concern for current SoCs, where the memory is implemented on chip and many metal layers are available for routing.

By having a full 16-bit data path, the eSi-1600 can often implement many code sequences in far fewer instructions than the 8051, resulting in improved performance, smaller code size and lower power consumption. Take the following trivial C example, which assigns one short integer to another.

```
short x;  
short y;  
void func1() {  
    y = x;  
}
```

As each of the variables are 16-bits, but the 8051 accumulator is only 8-bits, the value must be moved in parts. The assembler for this code sequence generated by the Keil 8051⁵ compiler, using the medium memory model (which is required to access more than 256 bytes of RAM) is as follows:

```
MOV     R0,#LOW (x)  
MOVX   A,@R0  
MOV     R7,A  
INC     R0  
MOVX   A,@R0  
INC     R0  
XCH    A,R7  
MOVX   @R0,A  
INC     R0  
MOV     A,R7  
MOVX   @R0,A
```

For the eSi-1600, this can be implemented very simply:

```
lhu     r4, ([x])  
sh      ([y]), r4
```

³ http://dcd.com.pl/dcdpdf/asi/dp8051cpu_ds.pdf

⁴ http://www.dolphin.fr/flip/logic/8bit/logic_8bit.php

⁵ <http://www.keil.com/>

Accumulator verses general purpose registers

The 8051 has a single accumulator which is where the results of most arithmetic and logical instructions are stored and where one operand can be sourced from. The eSi-1600 has 16 general purpose registers that all arithmetic and logical instructions can use to store their result or source operands from. While many 8051 instructions can be encoded in just 8-bits, by supporting 16 registers, the eSi-1600s instructions are always at least 16-bits. However, code size is not negatively impacted, as fewer instructions are often needed, because there is no need to constantly save or reload values from memory. The benefits are that performance is increased and power consumption is reduced, by fewer instructions and memory accesses being required.

Consider the following C example:

```
unsigned char x, y, z0, z1;
void func2() {
    z0 = x - y;
    z1 = x + y;
}
```

The Keil 8051 compiler generates the following code:

```
MOV     R0,#LOW (y)
MOVX   A,@R0
MOV     R7,A
DEC     R0
MOVX   A,@R0
MOV     R6,A
CLR     C
SUBB   A,R7
MOV     R0,#LOW (z0)
MOVX   @R0,A
MOV     A,R6
ADD    A,R7
INC     R0
MOVX   @R0,A
```

For eSi-1600, the following instructions suffice:

```
lbu     r5, ([x])
lbu     r6, ([y])
add     r7, r6, r5
sub     r5, r5, r6
sb      ([z0]), r5
sb      ([z1]), r7
```

Addressing modes

The 8051 supports both direct and indirect addressing modes. The addressing modes on the eSi-1600 are far more extensive: direct, indirect, offset, scaled-indexed and pre or post increment / modify. A simple block memory copy C function demonstrates how well suited a target for C code the eSi-1600 architecture is, in this case through the use of scaled-indexed addressing, while at the same time demonstrates the 8051s numerous limitations. The C test function is as follows:

```
void func3(unsigned char *src, unsigned char *dst, int len) {
    int i;
    for(i = 0 ; i < len; i = i + 1)
        dst[i] = src[i];
}
```

The 8051 assembler code generated by the Keil compiler, which must be an interesting challenge to debug, is as follows:

```
_func3:
    USING 0
    MOV R0,#LOW (src?040)
    MOV A,R3
    MOVX @R0,A
    INC R0
    MOV A,R2
    MOVX @R0,A
    INC R0
    MOV A,R1
    MOVX @R0,A
    CLR A
    MOV R7,A
    MOV R6,A
?C0001:
    SETB C
    MOV R0,#LOW (len?042+01H)
    MOVX A,@R0
    SUBB A,R7
    MOV A,R6
    XRL A,#080H
    MOV B,A
    DEC R0
    MOVX A,@R0
    XRL A,#080H
    SUBB A,B
    JC ?C0004
    MOV R0,#LOW (src?040)
    MOVX A,@R0
    MOV R3,A
    INC R0
    MOVX A,@R0
    MOV R2,A
    INC R0
    MOVX A,@R0
    MOV R1,A
    MOV DPL,R7
    MOV DPH,R6
    LCALL ?C?CLDOPTR
    MOV R5,A
    MOV R0,#LOW (dst?041)
    MOVX A,@R0
    MOV R3,A
    INC R0
    MOVX A,@R0
    MOV R2,A
    INC R0
    MOVX A,@R0
    MOV R1,A
    MOV DPL,R7
    MOV DPH,R6
    MOV A,R5
    LCALL ?C?CSTOPTR
    INC R7
    CJNE R7,#00H,?C0005
```

```
        INC    R6
?C0005:
        SJMP   ?C0001
?C0004:
        RET
```

In comparison, the eSi-1600 assembler generated by GCC could not be much simpler:

```
func:
        cmp     r10, 0
        ble    .L4
        lh     r11, 0
.L3:
        lbu    r12, (r8+[r11])
        sb     (r9+[r11]), r12
        add    r11, 1
        cmp    r10, r11
        bg     .L3
.L4:
        ret
```

There are a number of optimisations that the eSi-1600 C compiler is able to perform, when the highest optimisation level is set, that are not shown in the code above. These include transforming the code so that 16-bit memory accesses are used instead of 8-bit accesses (providing various runtime conditions are met) and also loop unrolling, so that the ratio of move instructions to conditional and branch instructions is increased. Both of these optimisations can be used to increase performance at the cost of increased code size.

MIPS verses DMIPS

When comparing processor datasheets, it is useful to understand the difference between MIPS and DMIPS, particularly when the authors themselves have not made it clear which is truly meant.

MIPS stands for Millions of Instructions Per Second. It is usually a peak figure, indicating the maximum number of instructions that can be executed in one second. For the original 8051, the fastest instruction took 12 clock cycles to execute, so at 12MHz, the 8051 was said to execute at $12\text{MHz}/12 = 1\text{MIPS}$.

For IP cores where the clock frequency is determined by the user, MIPS/MHz is often used to indicate how many millions of instructions are executed per million clock cycles. For a processor that can execute 1 instruction per clock cycle, peak performance would be quoted as 1MIPS/MHz. For a processor that takes 4 clock cycles for its fastest instruction, the figure would be 0.25MIPS/MHz.

Of course, not all instructions take the same number of clock cycles to execute. Therefore, real world performance will be less than this figure. While both the eSi-1600 and some enhanced 8051s can achieve a peak of 1MIPS/MHz, their real world performance is significantly different.

The Dhrystone benchmark was an attempt to provide a more realistic measure of performance. It is a C program consisting of a mixture of function calls, loops, comparisons, arithmetic operations and memory accesses. While it is quite a limited benchmark (⁶ provides a good description of its shortcomings), it is still by far the most commonly given benchmark figure.

The VAX 11/780 serves as the reference for the benchmark. It performs 1757 Dhrystones per second. To obtain a figure for any given processor, the benchmark is run and the number of iterations it performs in one second is divided by the reference number 1757. This produces the DMIPS figure, which again, is often normalized to DMIPS/MHz for IP cores.

⁶ <http://www.arm.com/pdfs/Dhrystone.pdf>

However, care should be taken to not forget about this normalization. It is harder to achieve high DMIPS/MHz figures for processors that will run at very high clock frequencies. For example, a processor with a DMIPS/MHz figure of 0.25 running at 200MHz, will be faster than a processor with a DMIPS/MHz figure of 0.5 running at 50MHz. Of course, should the processors clock at the same frequency, then a higher DMIPS/MHz figure will result in faster performance.

As shown in Figure 1: Dhrystone Performance, the performance of the eSi-1600 at the same clock speed is over 5 times greater than the fastest enhanced 8051, and 65 times greater than the original 8051. If you factor in the 600MHz clock speed that is achievable in a 0.13µm process, the eSi-1600 is over 3600 times as fast as the original 8051.

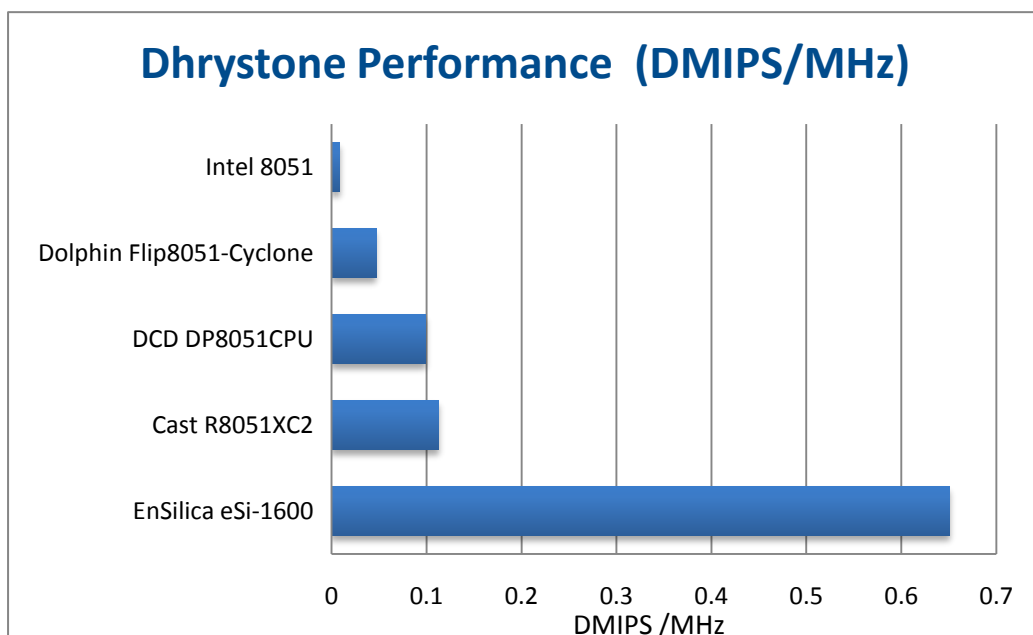


Figure 1: Dhrystone Performance

Processor	Dhrystone kernel code size (bytes)	
	Optimised for Speed	Optimised for Size
eSi-1600	2470	1162
8051	3710	1951

Table 2: Code Size

Low-power through high performance

While the performance figures above may be impressive, an 8051 would most likely only be being considered for a new ASIC design if the performance requirements of the target application were unlikely to be demanding. The main advantage therefore of having such great performance from a similar gate count, is that unneeded performance can be converted into lower power operation.

As shown in Figure 1: Dhrystone Performance, the eSi-1600 processor will execute the Dhrystone benchmark over 5 times faster than the closest enhanced 8051. For designs where low power operation is a concern and the extra performance is not required, the eSi-1600 could either be clocked at a 5x lower frequency, or have its clock disabled for 80% of the time relative to the 8051.

Vendor	Processor	Power Consumption ($\mu\text{W}/\text{MHz}$)	Power/DMIPS ($\mu\text{W}/\text{DMIPS}$)
EnSilica	eSi-1600	70.4 ⁷	108.1
Dolphin	Flip8051-Cyclone	91.8 ⁸	1953.2

Table 3: Power Consumption

Scalability

In addition to the extra performance and lower power operation already offered by the eSi-1600 over the 8051, there are several other advantages to the eSi-1600. For software inner loops that can more easily be performed in hardware (such as bit-twiddling in encryption algorithms), these can easily be accelerated using user-defined instructions, via a very simple interface. For applications that require a larger address space or even more computational power, the eSi-32xx range of processor cores are available. These 32-bit cores are based upon the same RTL and software toolchain as the eSi-1600, ensuring an easy migration path for both hardware and software developers.

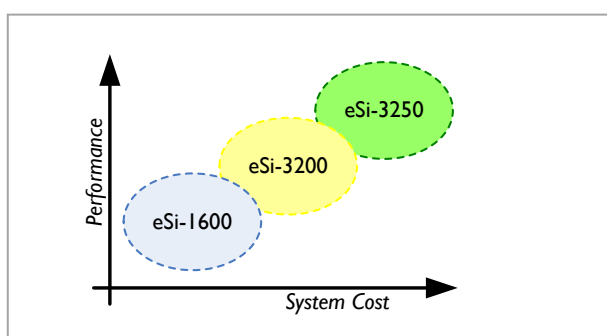


Figure 2: eSi-RISC family members

Conclusion

16-bit RISC processors such as the eSi-1600 are able to offer more than 5 times the performance of enhanced 8051 cores, while consuming a fraction of the power, with little or no increase in system cost.

About EnSilica

EnSilica is an established company with many years experience providing high quality IC design services to customers undertaking FPGA and ASIC designs. We have an impressive record of success working across many market segments with particular expertise in multimedia and communication applications. Our customers range from start-ups to blue-chip companies. EnSilica can provide the full range of front-end IC design services, from System Level Design, RTL coding and Verification through to either a FPGA device or the physical design interface (synthesis, STA and DFT) for ASIC designs. EnSilica also offer a portfolio of IP, including a highly configurable 16/32 bit embedded processor called eSi-RISC and the eSi-Comms range of communications IP.

⁷ In a 0.18 μm process, typical.

⁸ http://www.dolphin.fr/flip/logic/logic_core.php