

# Design Platform for Quick Integration of an Internet Connectivity into System-on-Chips

Bartosz Wojciechowski, Tomasz Kowalczyk  
Virtual Component Lab, Evatronix SA  
Dubois 16  
Gliwice, Poland  
Bartosz.Wojciechowski@evatronix.pl  
Tomasz.Kowalczyk@evatronix.pl

Wojciech Sakowski  
Institute of Electronics  
Silesian University of Technology  
Akademicka 16  
Gliwice, Poland  
Wojciech.Sakowski@polsl.pl

**Abstract** — The paper describes pre-integrated subsystem consisting of a configurable 8-bit microcontroller and an Internet connection solution. The latter integrates Ethernet Media Access Controller, hardware accelerator that implements TCP/IP checksums generation and analysis and the third party TCP/IP stack adapted to take advantage of the available accelerator. A reference design based on the presented subsystem is also described. It enables interaction with the object controlled by microcontroller by means of a web browser.

## I. INTRODUCTION

### A. Use of virtual components (IP cores)

Development of more and more complex systems-on-chip would not be possible if the reuse of predefined functional blocks known as IP cores or electronic virtual components had not become a common design practice.

However, an important part of the functionality of such systems is implemented in software and integration of numerous IP cores is still time consuming due to the continuous progress of semiconductor technology governed by Moore's Law that enables integration of higher and higher number of gates per sq.mm every year.

### B. Platform based design

Therefore the natural next step in the systematic reuse of predefined components is to offer combine a set of IP cores and low level software (often including embedded operating system) to form a skeleton of a system suitable to implement many different applications. Such approach is known as platform based design [1].

The definitions of the platforms differ in literature. In this article we will use a term *subplatform* to denote a set of IP cores and associated software that were integrated to implement complex function, and to make it easy for SoC designer to integrate this function in his/her chip.

While in most cases the platforms are built around 32-bit RISC architectures (with ARM processor playing a leading role), there are still many embedded control applications that are well served by 8-bit microcontrollers. Revival of Z80 instruction set with new families of microcontrollers (Zilog's eZ80) and maintained popularity of 8051 architecture prove this clearly.

In System-On-Chip designs 8-bit microprocessor or microcontrollers may be a central part of a design for applications where not much processing power is required (which may be a case for large class of control applications). They may also complement the main 32-bit processor in a multiprocessor SoC and implement "house-keeping" tasks off-loading the main processor. One of such tasks may be off-chip communication.

Developing a platform around 8-bit architecture requires maximum flexibility in shaping its peripheral set.

### C. Reasons for embedding Internet connectivity in System-On-Chips

Over the last few years, the interest for connecting computers and embedded systems to Internet network has steadily increased. It is due to the fact that Internet is seen more and more as the most cost-effective way of remotely monitoring and controlling of any object from any place in the world.

Therefore ways of fast and low cost integration of Internet connection are of interest of many System-on-chip designers that develop applications that may take advantage of easy access to Internet.

## II. R8051X-C CONFIGURABLE MICROCONTROLLER IP CORE

### A. 8051 instruction set architecture and its improvements

The major improvements to 8051 instruction set architecture were implemented as early as in 1999, during development of R80515 microcontroller core. It implemented peripheral set that was a mixture of peripherals from 80535 and 80537 chips once manufactured by Siemens Semiconductors. It was presented briefly in (Bandzerewicz and Sakowski, 1999). One year later the same CPU was used as a basis for R8051 microcontroller core that differed from the R80515 by peripheral set which matched exactly that of original Intel chip.

The improvements were based on the fundamental reduction of the number of clock cycles necessary to execute each instruction. A single clock cycle FETCH phase is interleaved with the last clock cycle of the execution phase of the previous instruction. The number of clock cycles necessary to implement each instruction is only limited by the number of

memory read/write cycles. Performance improvement close to the factor of eight over original architecture running at the same clock speed is achieved this way. A multiplying dividing unit (MDU) may be added to enhance some arithmetic operations on 16 bit data.

R8051XC did not introduce major improvements to this architecture, although execution of a few instructions was further enhanced. More advanced is expected from a pipelined version of 8051 ISA which is under development now and should bring 50% performance increase when compared to R8051XC.

### B. Rich peripheral set

Apart from optimization of instruction execution major extensions to the original 8051 architecture include configurable interrupt and DMA controllers as well as advanced power management options. Configurable interrupt controller enables the user to choose (with certain restrictions) a number of available priority levels and interrupt sources. One of the power management options is based on separation of CPU and peripheral clock domains with a handshake-based synchronization between them.

R8051XC enables to tune the peripheral set to user needs. In addition to the “standard” peripherals (timers, uarts, watchdog timer) SPI and I2C controllers can be added too.

### C. Configurability

Every second sale of our products from 8051 family involved a customization which required removing some standard peripherals and / or adding some that extended original architecture to match current needs (like I2C, SPI or – in some cases – USB device controllers or Ethernet MAC).

The idea of adding configurability to the microcontroller core was introduced pretty early (Bandzerewicz and Sakowski, 1999), but it was not available as an automated solution till recently, with a launch of a configurable microcontroller core based on 8051 instruction set architecture. We named it R8051X-C. It is based on our best selling R8051 and R80515. These two shared the same CPU design and differed with peripheral set that was derived from Intel and Siemens chips respectively.

Along with peripheral set quite a few other important features of the 8051 architecture are configurable in R8051X-C. The configurable capabilities cover:

- size of external program memory (64kB up to 8 MB)
- number of DPTR registers: 1, 2 or 8
- two types of interrupt controller with different number of sources and priority levels
- MUL, DIV, DA instructions – implemented or not
- separate Multiplication-Division unit: 0 or 1
- number of 8-bit I/O ports: 0 ... 4
- number of 16-bit timers: 0, 1, 2 or 3
- number of serial ports: 0, 1 or 2
- Watchdog timer: 0 or 1
- I2C and SPI master-slave interface: 0 or 1 for each
- On chip debug support options

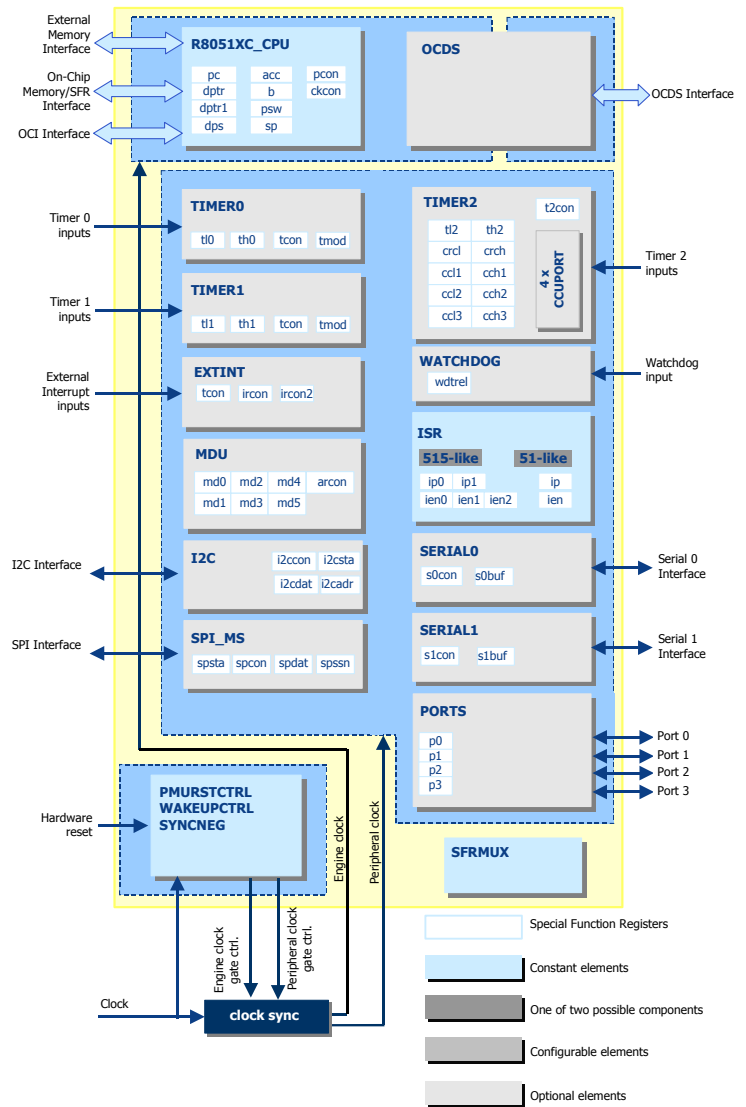


Figure 1. Architecture of the R8051XC – configurable microcontroller IP core

### D. Towards connectivity subsystems

In most applications some form of exchange of data over some standard is required. An example of this may be just I2C bus. As it was mentioned I2C Master Slave controller is an optional peripheral of R8051XC. However, enhanced performance of R8051XC core makes possible to use it for interaction with connectivity controllers of much higher data throughput like USB or Ethernet [2].

## III. EMBEDDED INTERNET CONNECTIVITY

### A. Introduction to Internet Protocol

The important reason of the popularity of use an Internet technology in embedded system industry was fact that TCP/IP protocols stack has proven itself flexible enough to incorporate the changing network environments. Besides this, the large connectivity of the global Internet is also a very strong

incentive. Today Internet technology runs over a large spectrum of link technologies with vastly different characteristics in terms of bandwidth and bit error rate.

Among all of the protocols used in Internet communication, the most important is Ethernet protocol. It is commonly used in embedded systems. It is a medium of choice because of its established infrastructure, relatively low price of implementation, competitive performance, and interoperability. Ethernet is also easy to use, widely available, and very scalable. These features are crucial for using TCP/IP protocol stack in embedded system. Since embedded systems, often working as a small devices, are usually required to be physically small and inexpensive, an implementation of the Internet protocols will have to deal with having limited computing resources and memory.

#### *B. Restrictions on TCP/IP implementation in 8-bit systems*

The basic problem in the implementation of TCP/IP on 8- and 16-bit microcontroller systems [3] is that an available system memory is usually too small for traditional TCP/IP implementation in terms of code size and RAM requirements. TCP protocol which is the most widely used of the transport protocols in the TCP/IP stack is also the most complex. In the traditional network stack implementations, TCP uses sequence numbering and retransmission in order to achieve reliable, ordered data transfer.

In embedded systems, fragmented packets are usually not handled. Also TCP options are often ignored there. Using such mechanisms, TCP/IP stack implementations may make it possible to run TCP/IP in small 8-bit systems. Furthermore, existing TCP/IP implementations for embedded systems assume that the embedded device always will communicate with a full-scale TCP/IP implementation running on a workstation-class machine. Thus, TCP/IP implementations running on embedded systems have to be optimized a particular application (e.g. an embedded web server), and are not suited for handling generic TCP/IP protocols.

#### *C. CMX-Micronet TCP/IP stack architecture [4]*

One of the best available networking software solutions optimized for 8- and 16-bit embedded processors is CMX-Micronet. This is a highly configurable TCP/IP stack that enables the developer to further minimize resource usage. It supports over than 15 the most common used network protocols, including such advanced features like sending emails and serving JAVA applets.

It requires 3 to 20kB of program code, depending upon processor, protocols used and options selected. Generally, CMX-MicroNet provides the full TCP/IP protocol stack with a few limitations. The most important are that it does not handle fragmented packets (IP fragmentation) and ignores all TCP options.

From application programmer viewpoint the TCP/IP stack in CMX-MicroNet can be seen as a "black box" that takes data from the application side and send them over Ethernet as transmission packets formatted according to the protocol, or

get them from network and make them available for the application specific function for handling the incoming data. Like many other network solutions for embedded systems, CMX-MicroNet uses callback function mechanism for handling user data. In this way, all details relevant to TCP/IP protocols are invisible for the user. By using callback functions, user may handle receive and transmit packets.

### IV. ETHERNET MAC IP CORES FAMILY DEVELOPED AT EVATRONIX SA

#### *A. Flexible Architecture Ethernet Media Access Controller*

The first Ethernet Media Access Controller (MAC) developed at Evatronix was build to control transfers over Ethernet (10 MHz) and fast Ethernet network. Its architecture is not tied to any particular host processor and can be configured to work with 8 bit, 16 bit, or 32 bit systems and either big or little endian. Great care was taken during its design over details of internal DMA, FIFO, and interrupt controller architecture. The details of these solutions have the greatest impact on the whole system performance.

For example - there are many Ethernet MAC controllers that implement scatter-gather DMA. But only few of them allow to make use of "zero-copy" functionality in the software TCP/IP stack. The difference here is data buffer alignment. If the controller requires some particular data buffer alignment (usually word alignment is required), then the software is forced to do excess data copying. If the controller allows free buffer alignment, then the copying is not necessary. Another example of flexibility – Ethernet controllers usually have configurable internal FIFO sizes. One can choose the appropriate size for your application before going into ASIC. But what about the number of frames that can reside in the FIFO at the same time? In most cases it is not configurable.

The final "dimension" of flexibility of this architecture is the interrupt subsystem. While its main role seem very simple: just to generate the interrupt for RX frame, and for TX frame. But for a large network traffic, servicing thousands of interrupts becomes a serious problem. In such case the feature called "interrupt mitigation control" come to the rescue. With interrupt mitigation control, the Ethernet MAC can generate single interrupt for multiple packets, thus reducing the total CPU overhead.

#### *B. Media Access Controller for 1 GHz Ethernet*

There were two directions of evolution of this base product. One path was towards the higher performance, i.e. towards support of 1 GHz Ethernet. Along with support of three Ethernet transmission speeds (10/100/1000 MHz), it has a few additional features important for high bandwidth systems: Flow control, set of Statistical Counters, and GMII management. What is important, the base architecture remains the same, so for the software developer, an upgrade of the software driver is very easy.

#### *C. Ethernet MAC for low end systems*

The second evolution path was towards the simplicity. Many applications require simple and low area Ethernet

controllers. Not only 8-bit controllers require simplicity, but also products with multiple instances of an Ethernet MAC, sharing single DMA engine, like for example switches. The product developed to address such applications was named MAC-L (L for Lite). When properly configured it may be used in high performance switches, but its relatively low gate count makes it perfect for applications that require Ethernet-enabled embedded systems based on 8-bit architecture.

The optimization of the base Ethernet MAC architecture towards low gate count required removal of sophisticated DMA, of large register set, and of interrupt controller from the base MAC 10/100. Instead of master DMA, the user gets the direct interface to the FIFO memories inside MAC-L, independently for RX and TX data paths. From the host side the MAC-L is seen as a slave with just a few control and status signals. On the other hand, many features remains unchanged from the base MAC 10/100 architecture. The same flexible FIFO buffering, the same Ethernet address filtering, including 512-bit hash table, and the same set of sixteen full 48-bit addresses.

And one additional feature that makes this solution exceptional: a built-in hardware TCP/IP acceleration. This feature, important in 8-bit systems, reduces the CPU work in computing TCP/IP/ICMP checksums.

## V. CHECKSUM HARDWARE ACCELERATION

### A. Hardware acceleration of the TCP/IP protocol

Basic activity of TCP/IP protocols resolved to transmission and reception of data frames. In both situations, preparing a transmission frame and checking a received frame use a mechanism of computing CRC. It is used for guarantee that a given frame is valid. All commonly used protocols like TCP/IP/UDP/ICMP use their own checksum mechanisms.

Computing the checksum often requires taking into account all bytes of data of a given layer, and for some protocols even taking data from other layers. It causes that microprocessor in embedded system is preoccupied for a significant amount of time with CRC computing. The longer frame the more time to compute it gets.

One of the solutions for this problem is use of a hardware CRC accelerator. This module can automatically compute a checksum for a given layer and insert it into the transmitted frame. For a received frame, it may check if the all checksums in the frame are valid, and give the information about it to the software or discard such frame. It would be good, if all settings referring to work of such module could be set from software level.

### B. Integration of the checksum accelerator with MAC-L

The module that support checksum acceleration and a number of other functions of TCP/IP stack was developed at Evatronix company and became an optional part of the MAC-L IP core. Ethernet controller module. It is suitable for automatic inserting and checking checksums for commonly used TCP/IP protocols. It handles TCP/IPV4/UDP/ICMP, both for transmit and receive frames. Setting for each protocol can be set

individually. Checksums for all protocol are computed on-the-fly.

For both directions: transmission and reception, the computing is realized before writing data to a given FIFO. It results in that processor does not spend a lot of time on computing and checking frames. In small systems, like the one with MAC-L, all protocol settings are enabled in hardware and can be "overwritten" during compilation of software with TCP/IP stack. In much complicated systems, it is possible to use more advanced schemes of checksum acceleration..

### C. Adaptation of CMX TCP/IP stack

To use CRC accelerating feature of R8051XC-MAC-L module, the number of modifications had to be done with CMX-MicroNet TCP/IP stack. Generally, these modifications rested on identification the part of the code, responsible for calculating particular TCP/IP checksum and adding a set of conditional preprocessor directives #if, #else and #endif, e.g.:

```
#if (!defined(R8051XC_MAC_L) ||
    !USE_HW_IPv4_CRC_STUFF)
    ip_csum_hb = HIGHBYTE(csum);
    ip_csum_lb = LOWBYTE(csum);
#else
    ip_csum_hb = 0x00;
    ip_csum_lb = 0x00;
#endif /* !R8051XC_MAC_L ||
        !USE_HW_IPv4_CRC_STUFF */
```

In this fragment if both macrodefinition R8051XC\_MAC\_L and USE\_HW\_IPv4\_CRC\_STUFF are defined, no software CRC calculation for IP frames is done and 0x0000 value of IPv4 CRC is inserted into the IPv4 frame. If not, formerly calculated CRC of IPv4 frame is inserted.

Additionally, in the mnconfig.h - main configuration header file, a section with settings of R8051XC-MAC-L was added. It contains detailed settings for all protocols, e.g.:

```
#define USE_HW_IPv4_CRC_STUFF X
#define USE_HW_IPv4_CRC_CHECK X
```

where X is 0 or 1, and denotes that a feature is disabled or enabled respectively.

Thus from the perspective of a user of Embedded Internet Subplatform (see section VI below) taking advantage of the checksum accelerator of R8051XC-MAC-L module requires only a setting appropriate features in mnconfig.h configuration file of the adapted CMX-MicroNet TCP/IP stack.

## VI. EMBEDDED INTERNET SUBPLATFORM

Integrating R8051XC and MAC-L enhanced with hardware accelerator for checksum calculations with third party TCP/IP steck (CMX-Micronet) created an Embedded Internet Subplatform.

It enables to develop quickly System-On-Chips with moderate performance that can handle Internet connectivity with limited burden for the CPU. It also allows designers of

more complex SoCs that require access to Internet with moderate throughput to offload main processor by integrating Embedded Internet Subplatform as autonomous subsystem in their SoC.

### VII. REFERENCE DESIGN

A reference design was developed to demonstrate usefulness of the Embedded Internet Subplatform in implementing control systems with Internet connectivity.

HTTP server built on top of the CMX Micronet TCP/IP stack runs on R8051XC, connecting to Ethernet network with MAC-L IP core (enhanced with hardware checksum accelerator).

R8051X-C interfaces a set of sensors (thermistor, photodiode, push buttons) and objects (set of LEDs and fan). It reads the temperature, level of light to which photodiode is exposed and position of push buttons. It may also switch on or off any of eight light emitting diodes as well as turn on or off a fan and set one of three speeds with which fan rotates.

The state of sensors may be seen with web browser that runs on PC connected to the design over Ethernet cable (after proper configuration of its TCP/IP connection). The web page managed by R8051XC is updated periodically or on demand of the web browser operator. The operator may control the state of the diodes and rotation of the fan by clicking corresponding graphic representation of these objects on the web page. Such command is transmitted over Internet protocol to application that runs on R8051XC and controls the diodes and the fan.

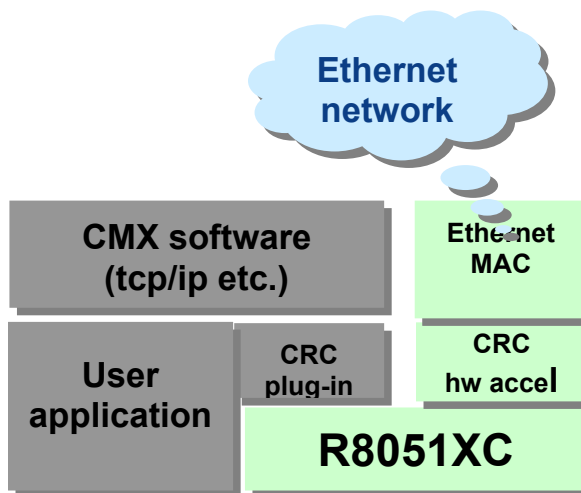


Figure 2. Reference design integrating R8051XC core with Ethernet media access controller and supporting software

### REFERENCES

- [1] Altizer B., Cooke L., Martin G. "Platform Based-Design: What is Required for Viable SoC Design ?", D&R Industry Articles, 2003.
- [2] M.Pyka, W.Sakowski "Configurable 8-Bit Microcontroller IP Core as a Basis for Effective System On Chip Implementation", Proceedings of the DesDes 2006 Workshop, Rydzyna, 2006
- [3] C. L. Stephens "TCP/IP - An Introduction for 8 & 16 bit Microcontroller Engineers", issued by Computer Solutions Ltd., 2002