Design Notes

Thinking of changing from an 80C51 to a 16 bit microcontroller? Why not consider the Philips XA?

The 80C51 family has been used extensively for close to 20 years, and there are many silicon suppliers who offer 80C51 derivatives. For this reason, many companies who have designed with this family of processor, have a tremendous investment in their software. As the need arises to develop faster and more sophisticated products, designers who have used 80C51's will need to use 16 bit processors. Ideally they would also like to utilise their existing software with little or no change. Philips has developed a processor called the XA which is geared for the 80C51 user. The processor's core is totally new, but it offers features which allow existing 80C51 code to run virtually unchanged.

This issue of Design Notes investigates some of the pros and cons of the XA processor and compares them to the 80C51. This article is not intended to be a summary of the existing XA literature. Instead, it outlines features the author found interesting when designing the XA into a recent project, and also discusses some things to be aware off which are not made obvious in the data sheets.

Brief Outline of the XA

The XA is a true 16 bit processor which offers 16 data lines and up to 24 address lines. Data manipulation and mathematical instructions can operate on 8 or 16 bit operands and some can even manipulate 32 bit long words. The processor can operate from DC to 30MHz and a typical instruction is 4-6 clock cycles duration. The processor offers a wide range of interrupts for UARTs, timers etc, as well as software and exception interrupts to flag stack overflows and divide by zero errors. The processor can also access up to 16MB of code and data memory in separate spaces. Data is accessed in 64k

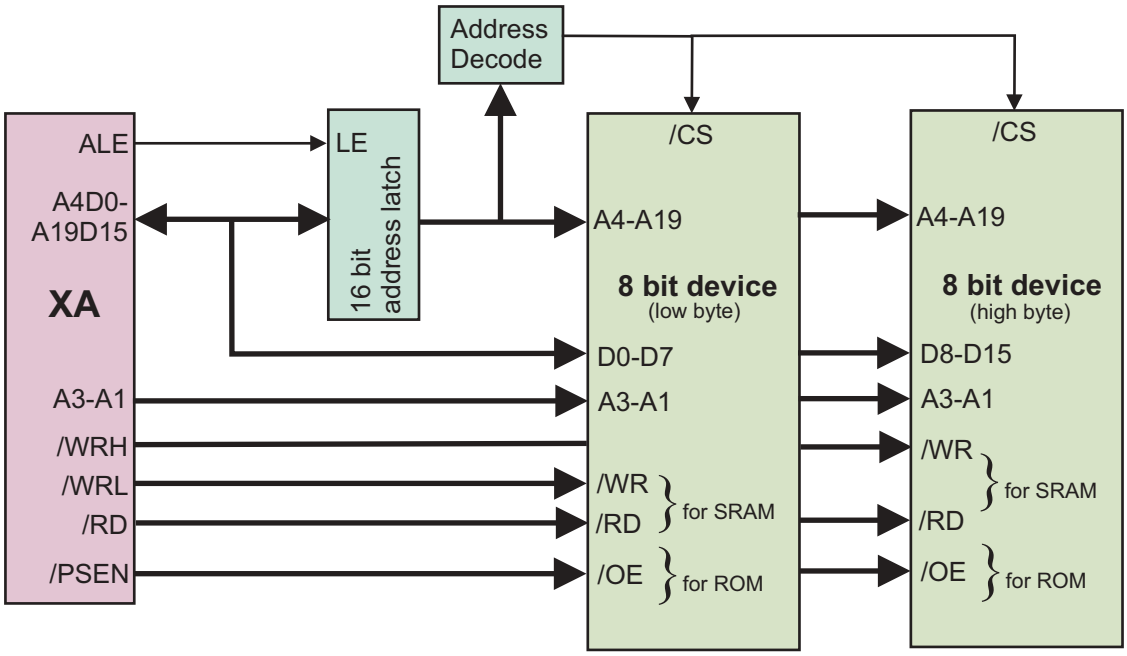

Figure 1 : Typical 16 bit XA Bus Connections to External ROM and RAM

Electronics By Design

Suite 25, 1-7 Jordan St Gladesville 2111
Phone : (02) 9816-3965
Fax: (02) 9816-3967
Web: www.electronicsbydesign.com.au

pages, allowing the processor to multitask by keeping the data of each task in a separate data page. Further information is available in references 1 and 2. There are currently only two derivatives available called the G3 and S3. The S3 includes the XA CPU and additional peripherals including an A/D converter and I2C bus. The G3 only offers the XA core with timers, dual UART's, watchdog and I/O ports.

Hardware

The XA is a processor that can emulate an 80C51 by operating in 8 bit mode, or can function as a native 16 bit device. Depending on the chosen mode, data paths for code and external RAM access will be 8 or 16 bit. The choice is configurable in software in conjunction with the bus width (BUSW) pin.

When writing to SRAM, 8 bit or 16 bit devices can be used. If 8 bit SRAMs are used, they can be written separately because they have separate write strobes, /WRL and /WRH. This occurs with instructions which have a byte operand. The architectural design and interfacing of the processor in 16 bit mode is shown in figure 1.

The reader will notice the XA does not multiplex A1-A3 lines. This increases speed by not requiring an ALE bus cycle when code is fetched within a 16 byte block. An ALE cycle will only occur when the 16 byte boundary is crossed. This typically reduces the number of ALE cycles by 50% or more.

The XA also has a wait state pin which allows it to stretch a bus cycle for an indefinite period when the pin is held high. This allows the processor to accommodate slow I/O. The XA's bus timing can also be modified by writing to registers BTRH and BTRL to stretch or shrink write cycles, read cycles, read or write strobe widths, address setup times and many other timing parameters. This is a great feature because the processor can accommodate slow memory or other I/O.

Interfacing to the XA is similar to an 80C51 except data transfers are 16 bit rather than 8 bit. I/O interfacing is TTL levels, program and data memory bus cycles are also similar. Due to the bus configuration registers BTRH and BTRL which control the timing of the bus cycles, most of the AC timing parameters in the data sheet are given as formulas dependant on crystal frequency and the values written in these registers. To check if an I/O device will interface to the bus, it is suggested all timing parameter formulas are entered in an Excel spreadsheet as functions of the clock frequency and BTRH and BTRL. This allows the design engineer to play around with the values written in BTRH and BTRL and optimise the timing to be as fast as possible to suit the target I/O device.

XA bus timing is similar to other Intel style processors such as those used on PCs and has a lot of similarity with bus cycles found on the ISA bus. This allows the processor to interface to a wide range of memory mapped peripherals thus offering the XA designer access to a whole host of cheap peripheral ICs such as UARTs, A/D convertors etc, which are normally used in PCs. The author recently interfaced the XA to an ethernet ISA bus controller chip without requiring any glue logic or modifications of any kind.

The XA also has a non maskable interrupt (NMI) input pin. This is typically used for signifying a power shutoff, and allows the processor to perform any required housekeeping before the VCC rail collapses.

Unlike the 80C51 ports, the XA's ports are configurable as quasi bidirectional, push-pull, open drain or high impedance inputs. This can be done on a pin for pin basis. I/O mode setup is done using the port configuration registers in the SFR space of the XA.

The reset line on the XA also differs from the 80C51 because it's active low similar to Motorola processors.

Software

Like an 80C51, the XA has a Harvard architecture i.e. separate address and data spaces. The data space can extend up to 16MB depending on the XA derivative. Memory is paged in 64k blocks using the DS or ES registers. XA has paged data memory to facilitate multitasking. The processor can

operate in system or user modes. In system mode, software can access all memory and resources on the chip. System mode is usually run only by system software residing in page 0 of code space. The system software also controls the execution of the multitasking software. To access a page of data memory the DS or ES register has to be written with the required page number. By using the ES register and running the processor in user mode, it is impossible for the current task to stray outside it's intended 64k data bank and corrupt another task's data memory. The page registers are under the control of the system software which resides in page 0 and cannot be changed by the multitasking software.

The downside of this occurs when multitasking is not required. Since the data memory space is not linear i.e. it's paged, it is cumbersome to access data across page boundaries because the DS or ES register has to be modified each time a page boundary is crossed.

256 pages of data memory are available if the full 16MB is utilised. Access within each 64k page is possible using direct and indirect addressing up to 3FFH and indirect only addressing from 400H to FFFFH. Data memory configuration is shown in figure 2.

Unlike the 80C51, the XA addresses off chip RAM using indirect addressing through the registers R0-R6. While MOVX still exists in the instruction set, it is used only for compatibility with 80C51 code and only works on page 0 of data memory space.

Another difference when compared to an 80C51 is the versatility of the processor's register file. There are eight word wide registers R0-R7. R7 is used as the stack pointer. R0-R3 are bank switchable to one of four banks. R4-R6 are not switchable. Unlike the 80C51, the XA does not have an accumulator. Instead, any of R0-R6 are used for data transfer and manipulation instructions making it very flexible and fast since everything does not have to go through the accumulator. Register pairs can also be concatenated to form a 32 bit wide registers. In addition any of the registers can be used as pointers with or without auto increment, making the processor ideal for transferring data between pages where multiple pointers are required.

With 16 bit processors comes the issue of where the low byte is stored relative to the high byte in ROM or RAM. The XA uses the "little endian" configuration, i.e. the least significant byte of a word is stored in the lower memory location and the most significant is in the higher address. The low byte is stored at an even address (A0=0), and the high byte at an odd address (A0=1).

Interrupt priority on the XA is very versatile. Each interrupt can be separately assigned a priority level between 0 and 15. Once the interrupt service routine (ISR) is entered, it's priority can be re-set so it can also be interrupted, i.e. nested interrupts are allowed.

Software interrupts (SWI) are also a feature of

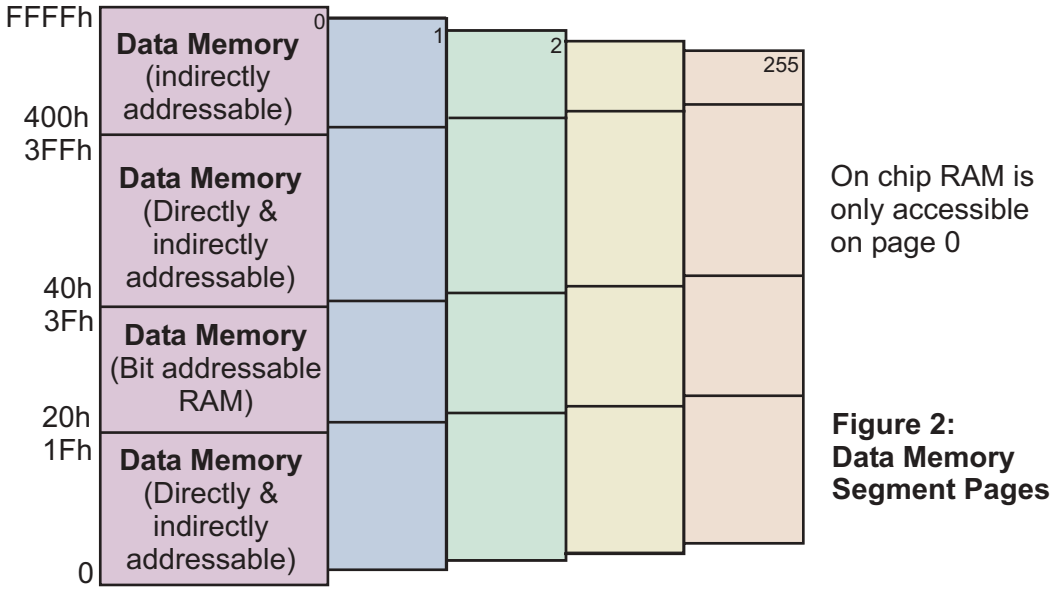

Figure 2: Data Memory Segment Pages

the XA. If an interrupt occurs for a UART (say), the ISR will be executed. This will usually perform some minor housekeeping tasks which then need to be followed up with further processing. The housekeeping is done in the ISR, while the further processing is assigned a software interrupt with a jump vector and priority. After the ISR is finished, code execution will return to the highest pending interrupt which could be the SWI, or any other interrupt. The SWI will be executed when it's priority is the highest pending one at a particular time. SWI allows semi urgent tasks to be executed in a reasonable time frame without having to be specifically called by the software and avoids them being included inside critical ISR's such as timers or UARTs thus making these interrupts very fast.

The XA also offers a reset command which generates a warm reset under software control. This is useful if the software detects a fault condition such as a task gone haywire or a stack crash, and cannot rectify it unless it resets the processor.

Interrupt vectors on the XA are very interesting. In the 80C51 the interrupt vector location where code execution begins usually has a JMP instruction or a very short ISR. In the XA, the vector location must hold the data to be written into the PSW when the interrupt occurs. This is followed by the address location which is a label corresponding to the start of the ISR, i.e. there is no JMP instruction.

The PSW in the XA plays a much more significant role than it does in the 80C51. PSW high byte holds flags for setting the mode of operation (system or user mode) the trace mode and also sets the priority of the currently executing ISR which will determine which other interrupt (if any) will be allowed to interrupt it. PSW low byte is similar to the 80C51 and holds the usual mathematical result flags such as C, Z, N etc.

The above requirements for an interrupt also apply to the reset vector and the start of code execution. The PSW value must reside at address 0000 on page 0 (low byte first) followed by the label which corresponds to the start of code execution. The XA User Guide in chapter 4 shows the correct code initialisation routine. The reader is also encouraged to initialise SCR and BCR immediately as the first steps of code execution.

The XA also has exception interrupts for handling exceptional circumstances including divide by 0, stack overflow and a breakpoint amongst others. The stack overflow exception occurs when the stack crosses the 80H data memory location. Since the XA stack grows downwards, the author recommends it be set at some location above 80H, such that under normal conditions the stack will not spill over the 80H barrier. If a problem occurs and 80H is crossed, the interrupt will occur and the condition can be rectified by software.

The XA also offers a very rich instruction set. Additional instructions which are very useful include conditional branches such as BEQ, BNE, BGT and others similar to those offered by Motorola processors. There are also extra rotate and shift instructions with extra flexibility which allow the required amount of shifting to be specified.

The usual bit test and branch instructions are also available. Unfortunately one instruction that has been removed is the bit complement (CPL) instruction.

The XA allows the clock source for the timers and UARTs to be pre-scaled at osc/4, osc/16 or osc/64 speeds. This allows the use of a fast crystal in conjunction with slow timer and baud rates.

Acknowledgments
The following are registered names and trademarks:
Philips, Intel, ISA, 80C51, XA, Motorola,

References
(1) XA User Guide - 1997
Philips Semiconductors
(2) XA data sheets for XA-G3 and XA-S3

Author
Ef Misoyannis - Electronics By Design