

Customized 8051 processors – suitable for a small quantity of units

by Harald Flügel, Hitex Development Tools

THIS ARTICLE SHOWS THAT IT IS NOT AT ALL DIFFICULT TO DEVELOP ONE'S OWN 8051 DERIVATIVE, WHICH CAN ALSO BE PROFITABLE FOR A SMALL QUANTITY OF PIECES.

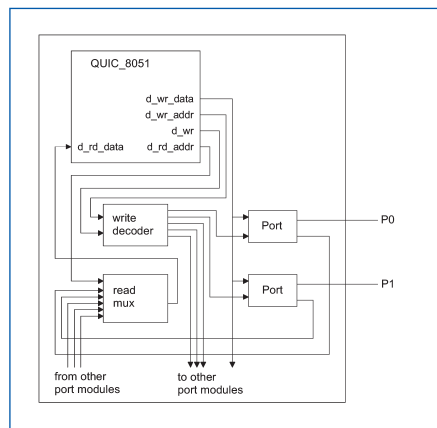


Figure 1. Block diagram for the connection of the port module

Developers often think that although a customized design for their project would be quite interesting, this is only economically justifiable for a large quantity of pieces. The majority of electronic developments are only produced in batches for which an ASIC design would not be affordable. However, there is a possibility of implementing SoCs into an FPGA, and this solution is suitable for a small quantity of units.

Today, modern FPGAs provide sufficient space to incorporate an SoC. Low-priced soft CPUs are offered by FPGA manufacturers, which one can customize oneself, and place into one's FPGA. Indeed, software developers are faced with an unknown processor architecture with the resulting adjustment effort in the CPU and the development environment. The debugging possibilities of such CPUs are also rather limited. Furthermore, hand-optimized software components often have to be taken over from earlier developments. From a development point of view, it is preferable that the SoC includes a processor that is already well known. For this reason, the choice of embedded processors and associated development and test tool kits is a critical issue.

The 8051 architecture is the most common for simple control applications. The following sample designs show how one can create one's

own 8051 derivative and fit it into an FPGA. There, a special emphasis has been placed on the use of widespread and effective tools. Development environments such as Altera's Quartus and Keil's μ Vision, as well as Hitex's DProbeHS emulator are used for debugging. The cost aspects are also taken into consideration. The license costs for the soft CPU per assembly for a quantity of 1000 pieces account for less than a quarter of the costs of the FPGA.

If an external address/data bus and the secondary functions of the port pins are required, only a few port pins are left over on most 8051 derivatives. If one requires many I/O signals, then one has to attach one's own port expansion to the micro controller. It is disadvantageous that these external ports are only accessible through MOVX commands, thus eliminating elegant and effective programming with bit instructions (SETB, JB).

The specifications of the sample design described herein state that an 8051 derivative in a FPGA with 56 bidirectional pins available in addition to all interrupt and timer inputs should be designed. The required seven 8-bit-ports should all be located on bit-addressable addresses. In addition, the derivative should have 4 KB of internal XRAM. The processor's maximum frequency should be 60 MHz based on 8051 standard timing with 12 clocks per

machine cycle. Finally, the derivative should be emulatable, in order to guarantee a high functionality when testing software.

For the FPGA design, the 8051 module from Quickchip has been selected, because it fulfills important requirements for emulatability. This module includes only one port (P2), so additional six port modules must be added. For that, the processor module, which is available as an EDIF net list, is integrated into a top-level design that contains an address decoder for write access, and a data multiplexer for the read data. In addition, six instances of the standard port modules are attached to the micro-controller core. The ensemble is then compiled in the Quartus development software, and mapped in an FPGA from Altera's Cyclone device family. The QUIC_8051 processor module, the port module, and XRAM use 2654 logic elements and 34816-bit memory, so that the entire circuit is suitable for the smallest possible FPGA cyclone (EP1C3) that is available. One requires a TQFP144 casing due to the quantity of port pins. Quartus calculates a maximum frequency of over 25 MHz for the design for the FPGA with the lowest speed rate (-8). This is equivalent to a standard 12 clock 8051 running at 150 MHz and thus surpasses the speed requirement of the specification by far.

The block diagram for the connection of the

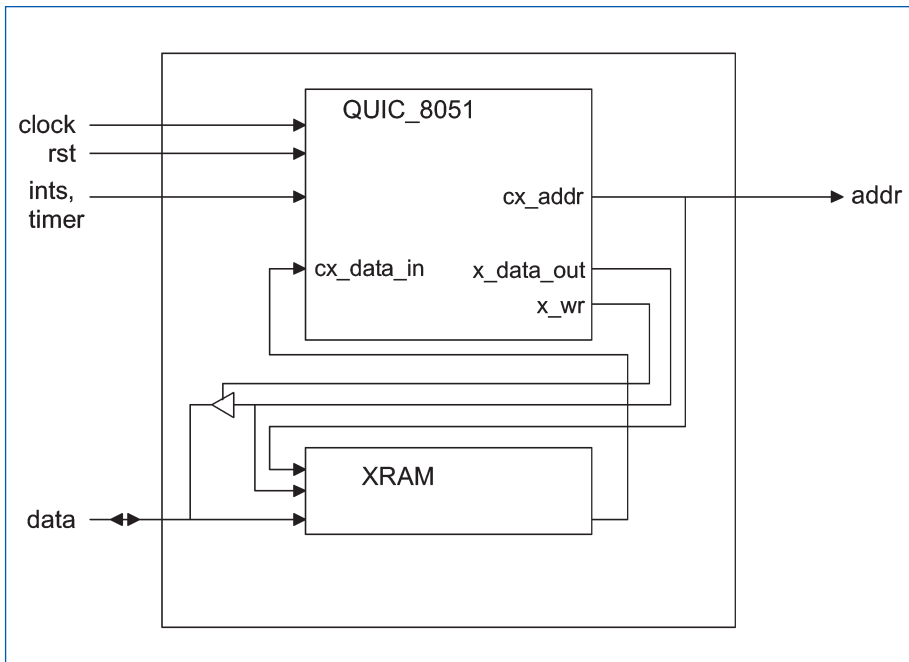


Figure 2. Block diagram for the connection of the internal XRAM and the external memory

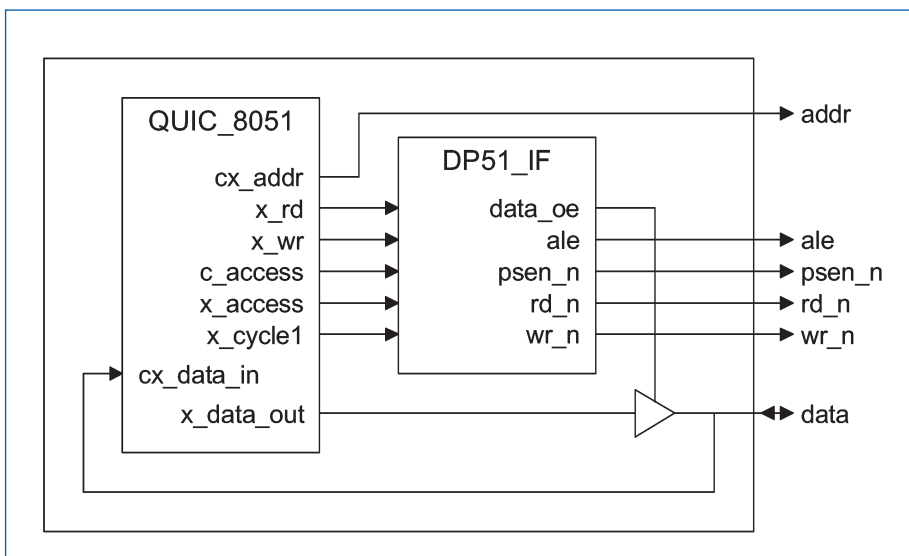


Figure 3. The interface for the emulator

port module (figure 1) shows the processor module and port modules, the address decoder for the write address, and the multiplexer for the read data. The selected port module receives a signal from the write decoder when the processor writes to the D Segment addresses (MOV P0, A). The selected port module then accepts the data from the processor. The read back multiplexer selects the data of the port module it should give to the processor during the read-access process.

The block diagram for the connection of the internal XRAM and the external memory (figure 2) shows that the 16-bit address for the code and Xdata access generated from the processor is simply connected to the

FPGA pins. The address also goes to the on-chip XRAM module, which decodes the address and the access signals (x_rd, x_wr) and thus detects the accesses to the on-chip memory, to which it responds. The data-mapping module is located between the external memory and the processor module's data input during read access. If the read accesses the on-chip XRAM module, the RAM data is fed to the processor module, otherwise the data from the external memory is fed to the processor module. The XRAM module accepts the data during the write cycles; however, it does not prevent the data in the external data bus from being visible. Thus, you can see the written data in the trace's emulator and also trigger it.

The interface for the emulator (figure 3) differs only slightly from the standard bus interface. Just like the standard bus interface, the strobe signals psen_n, rd_n and wr_n as well as the enable signal for the data bus driver are generated from the processor module's signals. The only enhancement is the ALE signal, which is necessary for the DProbeHS emulator. Actually such a signal makes no sense, because the address/data bus is not multiplexed. However, the emulator requires a signal with which it can latch the address, because it must extract information about the internal state of the processor out of the progression of the addresses issued.

The Cyclone FPGA family components offer pull-ups for all pins. You can activate these pull-ups by making an entry in the esf file. By doing so, the ports behave like original 8051 ports with a strong pull-up during 0-to-1 transitions and a weak pull-up afterwards.

```

OPTIONS_FOR_NODES_AND_ENTITIES
{
    p0 : WEAK_PULL_UP_RESISTOR = ON;
    p1 : WEAK_PULL_UP_RESISTOR = ON;
    p2 : WEAK_PULL_UP_RESISTOR = ON;
    p3 : WEAK_PULL_UP_RESISTOR = ON;
    p4 : WEAK_PULL_UP_RESISTOR = ON;
    p5 : WEAK_PULL_UP_RESISTOR = ON;
    p6 : WEAK_PULL_UP_RESISTOR = ON;
}
    
```

The soft-core processor used is compatible with 8032 and thus one may select the 8032 generic processor derivative in the µVision development environment (Project -> Select Device). The addresses for the new P4 to P6 ports have to be broadcast to the environment, should it be a .h-file in C coding or an .inc file by the assembler.

Excerpt of .h file

```

sfr P4 = 0xC0;
sfr P5 = 0xD8;
sfr P6 = 0xE8;
    
```

Excerpt of .inc file

```

P4 DATA 0C0H
P5 DATA 0D8H
P6 DATA 0E8H
    
```

After broadcasting the addresses, one can access the new ports in the same manner as the common ports.

According to the specification, the maximum frequency of the embedded processor should be 60 MHz, based on the standard 8051's 12-clock scheme. As the processor core used is a 2-cycle machine, the FPGA must be clocked at 2/12 of

the target frequency, thus 10 MHz. The emulator, however, must be capable of running at 60 MHz. This is the reason why only very fast emulators like the DProbeHS can be used to emulate such a SoC.

Otherwise, there is not much to consider concerning emulation. The DprobeHS emulator offers a universal interface for the adaptation of any 8051 derivative. A 60-pin header must be placed onto the component assembly, and the emulator must be connected with this connector. In the HiTOP user interface, one can select the 8032, the simplest one from the multitude of possible derivatives. If one would like to display the additional ports in HiTOP, one must assign a SFR file; otherwise, no further data is required.

Excerpt from my_8051.sfr (based on 8032.sfr)

```
;Windows for all ports
```

```
REGION : BASE "D:0x00", LENGTH 256
```

```
P0      0x80,   1,    RW
P1      0x90,   1,    RW
P2      0xA0,   1,    RW
P3      0xB0,   1,    RW
P4      0xC0,   1,    RW
P5      0xD8,   1,    RW
P6      0xE8,   1,    RW
```

```
ENDREGION
```

```
WINDOW 1 "Ports "
```

```
| |
| [P0 : ##]      [P1 : ##] |
| [P2 : ##]      [P3 : ##] |
| [P4 : ##]      [P5 : ##] |
| [P6 : ##]      |
| |
ENDWINDOW
```

Contrary to what might be assumed, it is not problematic to develop one's own SoC, which optimally fulfils one's own specifications. The advantages of a SoC design are complex. They include a direct cost advantage for components and production costs. Furthermore, there are better EMC characteristics and more effective peripheral interfaces to the CPU for these designs. SoCs in an FPGA have the serious advantage of having the easiest convertibility, which even extends to previously released products. Such a chip can never be discontinued, because one can "re-pot" the design into another FPGA device if required. The 8051 derivative presented here offers suitable additional testing possibilities. The fact that the sample design illustrated here fits into the smallest member of a modern FPGA family illustrates the possibilities that accompany this technology.

You can find the sample design on the internet at www.quickchip.de as Application Note Altera 004. ■