QUIC_8051
Users Manual

Core release 1.0
Document release 0.8

Contents

1 Available documents

This users manual describes the usage of the QUIC_8051 embedded controller core. It explains how to instantiate the module and where to connect the ports. A detailed information about the structure of the core and the function of the ports is given in the datasheet. Application notes with example designs complete the literature for the QUIC_8051.

2 Overview

QUIC_8051 is an 8051 compatible processor core for FPGA implementation. It is available in the form of EDIF net lists for different FPGA families. The core consists of:

- CPU
- Registers
- RAM (256 Byte)
- Timer 0, 1, and 2
- UART
- Interrupt Controller
- Port 2

3 Interfaces

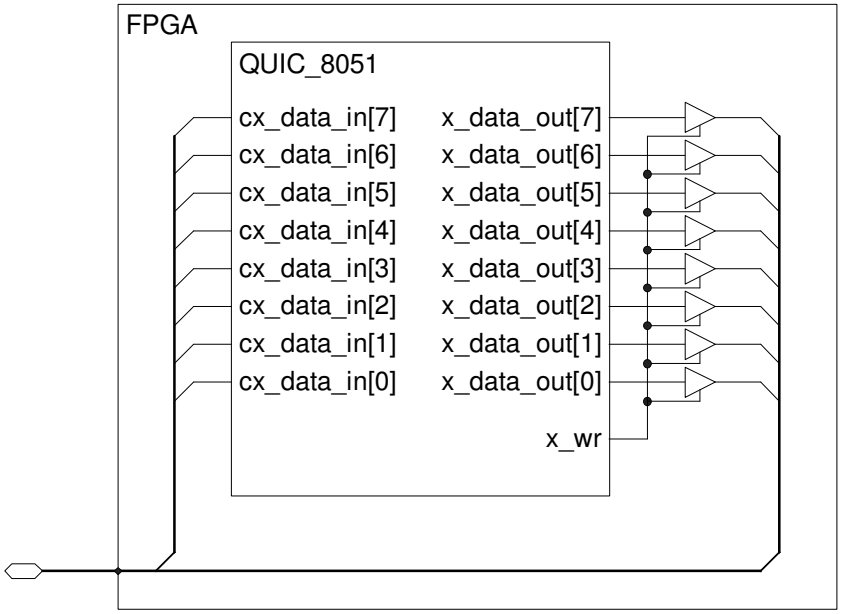
3.1 Global signals

Signal	Usage
clock	The usual recommendations for a clock signal of an FPGA apply for this clock signal, of course. Use a global clock signal so that there is only minimal skew between clocked registers.
reset_in	Because of the internal synchronization there are no special requirements for the reset input. It is advisable to have a proper reset signal with low rise an fall time, though.
reset_out	The synchronized version of the reset signal can be used for other modules in the FPGA that need to be reset. It can also be routed to an output pin and used on the board.

3.2 Address and data interface

Signal	Usage
cx_addr[15:0]	The usage of the address lines depends on the type of external bus used. If the address lines are not multiplexed with the data lines (and it is definitely recommend to have these busses separated), these address lines can simply be routed to the appropriate output pins.
x_data_out[7:0]	The data output lines are only of concern if there is an external RAM for the X-segment variables or if there is X-segment memory mapped IO. If no external X-segment is used. the data output lines can simply be left unconnected. Otherwise, the two busses cx_data_in and x_data_out must be routed to the same pins of the FPGA. The x_data_out lines need to be buffered using a tri-state driver. The enable signal for this buffer can be derived from the signal x_wr. See figure 1 on page 5.
cx_data_in[7:0]	The data input lines must simply be connected to the input pins where the external memory is connected to. This is independent of the type on external bus used. If no external bus is used, tie these pins to low level.
c_access (output)	Invert this signal to generate a low active chip enable signal for a external standard EPROM.
x_access (output)	Invert this signal to generate a low active chip enable signal for a external standard RAM, if there is any. If there is no, the signal can be left unconnected. In that case it is still advisable to have the external EPROM be controlled by c_access because this signal is inactive during reset and idle mode.
x_cycle1 (output)	This signal is only necessary if a multiplexed external address/data-bus must be built. It is valid in the first clock cycle of a X-segment access. In this cycle the ALE signal must be strobed high during the high phase of the clock.
x_rd (output)	The external read strobe can be derived from this signal by just inverting it.
x_wr (output)	The external write strobe can be derived from this signal by just inverting it. I case there is no external X-segment, both the x_rd and the x_wr signal can be left unconnected.

Figure 1: Buffers for x_data_out



3.3 Port 2

The input and output lines for Port 2 must be routed to the same pins of the FPGA. The p2_out lines need to be buffered using a tri-state driver. So, the wiring scheme is nearly the same as for the busses cx_data_in and x_data_out. There is only one difference: Every buffer needs its own enable signal. Note that the port pins need pull-up resistors at board level.

Signal	Usage
p2_in[7:0] p2_out[7:0]	The input and output lines for Port 2 must be routed to the same pins of the FPGA. The p2_out lines need to be buffered using a tri-state driver. So, the wiring scheme is nearly the same as for the busses cx_data_in and x_data_out. There is only one difference: Every buffer needs its own enable signal. Note that the port pins need pullup resistors at board level.
p2_oe[7:0]	Enable signals for the output driver for the p2_out lines.

3.4 Timer and Interrupt

The inputs of the timer and the interrupt module must be connected to their internal or external sources according to the application. All input signals are synchronized inside the QUIC_8051, so there are no precautions. The level of these signals is the same as on the original 8051 (or 8052, respectively).

3.5 UART

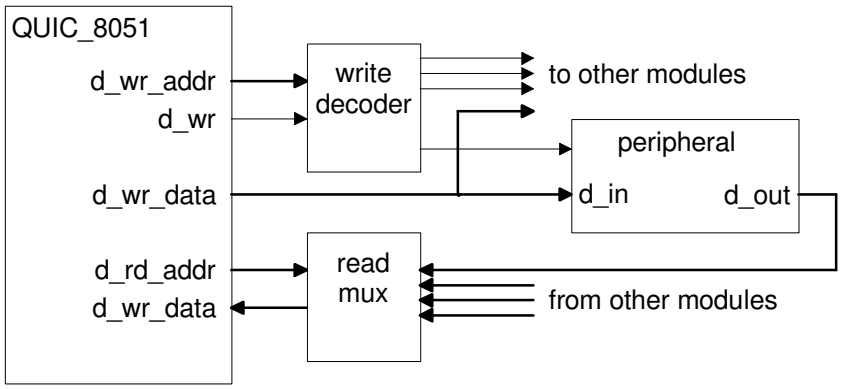
Signal	Description
txd	The output of the serial shift register can just be connecter to the approptiate pin of the FPGA. Note that for a RS-232 compliant transmission there must be an inverting level shifter connected to this FPGA pin.
rxd_in rxd_out rxd_oe	Because of the mode 0 of the serial interface there are three signals associated with the rxd pin of the FPGA. Just like the port P2 pins, the rxd_in signal is connected directly to the pin of the FPGA. The rxd_out signal drives a buffer the enable of which is driven by the rxd_oe signal. In case the mode 0 of the serial interface is not used in an application, the signals rxd_out and rxd_oe can be left unconnected. This saves a few logic cells.

3.6 Internal data bus

The handling of the dataflow differs for the write and the read data path. For the write data path an address decoder is used. This read data path needs a multiplexer.

Signal	Description
d_wr_addr[7:0] d_wr	The write address d_wr_addr and the write enable signal d_wr must be connected to the write address decoder. This decoder generates the individual write enable signals for the SFRs by simply comparing the address with the fixed address values of the SFRs and checking that the write enable is valid.
d_wr_data[7:0]	The write data bus must be connected to the data inputs of all peripherals, thus their SFRs.
d_rd_addr[7:0]	The read address d_rd_addr must be connected the select input of the read multiplexer.
d_rd_data[7:0]	D-segment read data.
d_rd d_rd_rwm	These signals are only used if there are user added SFRs that must be notified that the have been read (like a handshake SFR) of read for read-modify-write (like the port SFR).

Figure 2. Write decoder and read multiplexer



3.7 Internal state

These signals can be used by user added peripheral modules to determine the state of the processor execution.

Signal	Usage
idle	This signal can be used to disable user added peripheral modules in idle mode. The signal can also be routed to external devices to reduce the power consumption during idle mode

4 Recommendations

In this chapter some hints for a successful FPGA implementation are given

4.1 Synchronous design

It can't be said often enough: Keep the design synchronous. Do not design-in flip flops that are clocked with another signal that the global clock. In case a peripheral logic module needs another clock, keep this peripheral synchronous and use double flip flop synchronization logic between different clock domains.

4.2 Separate control and status signals

It is highly recommended that the control, mode, and command signals that are set by the software are separated from the status and result signals that are read by the software. For a proper design, always implement separate SFR for these two information directions and do not mix them up.

The original 8051 has such mixed SFR and these may cause trouble. The SFR TCON, for example, consists of four SFR bits that are mode signals for the timer and the interrupt logic: TR1, TR0, IT1, and IT0. The other four SFR bits (TF1, TF0, IE1, and IE0) can be modified set by the timer or interrupt controller as well as by the software. Now, imagine that the CPU executes the instruction:

```
ANL     TCON, #5Fh
```

This instruction clears both timer overflow TF1 and TF0 bits in one single instruction. For that, the CPU reads the content of the SFR in one clock cycle and writes the modified value back to the SFR in the subsequent clock cycle. If an external interrupt that sets one of the interrupt bits occurs just between these two clock cycles that interrupt would be lost if there was no special circuitry in the QUIC_8051 that prevents this. You can eliminate the need for such a special circuitry by separating the information directions.