QUIC_8051
Datasheet

Core release 1.0
Document release 0.9

Contents

1 Available documents

This datasheet describes the functionality of the QUIC_8051 embedded controller core. The structure of the core and the meaning of all ports are explained in detail. Information about the usage can be found in the users manual. 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

QUIC_8051 is compatible to the original 8051 micro controller that was designed by Intel in 1979. However, it is not identical. The main difference is the fact that QUIC_8051 executes the instructions six times faster than the original 8051 when both controllers are clocked at the same frequency. The peripherals, however, are not speeded up.

2.1 Pipeline Architecture

The CPU of QUIC_8051 is built up in modern pipeline architecture with three stages. Because of that, it is possible to achieve an operating frequency of 20 MHz without the need to use the fastest FPGA available. So, a speed equivalent of 120 MHz, based on the original 8051, can be reached easily.

2.2 Compatibility

Despite this optimization, the CPU is very close to the original Intel 8051 micro controller concerning the behavior of instruction execution and interrupts. Because of that, an FPGA with a QUIC_8051 can be connected to a standard 8051 In Circuit Emulator. So, very powerful debug tools can be used for the test of the software running on a QUIC_8051 based FPGA.
Great verification effort has been spent to assure the compatibility to the original Intel 8051 micro controller. For that, a lot of test programs have executed on an Intel 8031-AH derivative and a QUIC_8051 based FPGA. Such test programs even included somewhat insane instructions like "POP SP".

2.3 Usage

The module can be integrated into an FPGA easily, thus getting access to the external code and data segment memory that must be connected to the FPGA. The internal data memory is built up of RAM modules of the FPGA. As a special feature, QUIC_8051 offers the possibility to connect self-built peripheral modules to the internal data bus of the core. This, in turn, makes the modules accessible via internal data segment addresses, giving much more flexibility to the programmers. It is even possible to connect external peripherals to the FPGA in a way that they can be accessed through internal data segment addresses.

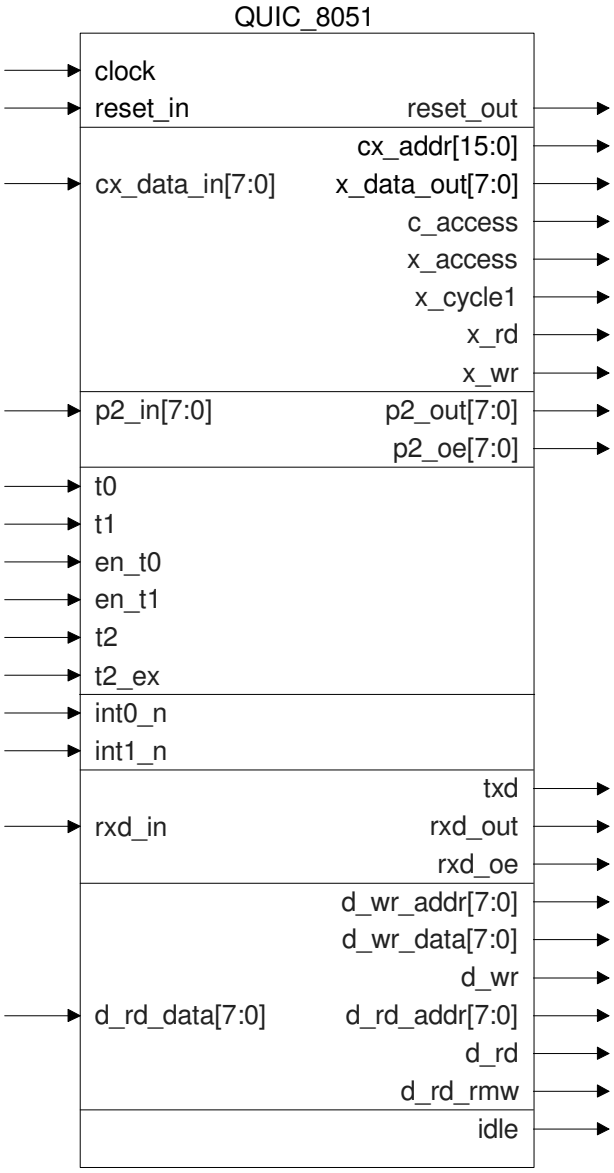
The interrupt and timer inputs are available separately and can be driven from internal or external sources, just as needed.

3 Top level view

This block diagram shows all input and output ports of the QUIC_8051 module. The ports are grouped into the classes:

- Global signals
- Address and data interface
- Port 2
- Timer
- Interrupt
- UART
- Internal data bus
- Miscellaneous

Figure 1: Top level view



4 Interfaces

The following tables describe all input and output signals of the QUIC_8051. Low active signals are indicated with the suffix "_n". For a functional timing information see the timing diagrams in chapter 6 on page 10.

While the original Intel 8051 has numerous double and triple used pins, QUIC_8051 has not. Every signal has only one meaning. If the user wants to have INT0 and the enable signal for timer 0 driven by the same FPGA input pin, then the two QUIC_8051 signals int0_n and en_t0 must be connected to the same FPGA pin in the custom top-level design.

4.1 Global signals

Signal	Direction	Description
clock	in	Global clock for CPU and peripherals. The module is clocked on the rising edge of this clock signal.
reset_in	in	Global reset for CPU and peripherals. The reset input is synchronized inside of the QUIC_8051 module. So it can be driven from an asynchronous source.
reset_out	out	This is the synchronized version of the reset signal.

4.2 Address and data interface

Signal	Direction	Description
cx_addr[15:0]	out	Address lines for all C and X-segment accesses to the external memories.
x_data_out[7:0]	out	Data output lines for all write accesses to the external X-segment RAM.
cx_data_in[7:0]	in	Data input lines for all C and X-segment accesses to the external memories.
c_access (output)	out	Access to the C-segment is in progress.
x_access (output)	out	Access to the X-segment is in progress. During normal program execution, one of the two signals c_access or x_access is active. However, during reset or idle mode, both signals are inactive.
x_cycle1 (output)	out	The first of the two clock cycles of a X-segment access.
x_rd (output)	out	Read strobe for the X-Segment. Note that this signal is high active and separated from P3.7.
x_wr (output)	out	Write strobe for the X-Segment. Note that this signal is high active and separated from P3.6.

4.3 Port 2

Signal	Direction	Description
p2_in[7:0]	in	Input lines for port P2.
p2_out[7:0]	out	Output lines for port P2.
p2_oe[7:0]	out	Individual enable signals for the output driver for the p2_out lines.

4.4 Timer

Signal	Direction	Description
t0	in	Input for timer 0 in counter mode. On the original 8051, this input is combined with P3.4.
t1	in	Input for timer 1 in counter mode. On the original 8051, this input is combined with P3.5.
en_t0	in	Enable input for timer 0. On the original 8051, this input is combined with INT0_N and P3.2.
en_t1	in	Enable input for timer 1. On the original 8051, this input is combined with INT1_N and P3.3.
t2	in	Input for timer 2 in counter mode. On the original 8052, this input is combined with P1.0.
t2_ex	in	Reload or capture input for timer 2. On the original 8052, this input is combined with P1.1.

4.5 Interrupt

Signal	Direction	Description
int0_n	in	External interrupt 0. On the original 8051, this input is combined with P3.2. The signal is low active so that the original 8051 documentation on interrupt level and edge is still valid.
int1_n	in	External interrupt 1. On the original 8051, this input is combined with P3.3. Also a low active signal.

4.6 UART

Signal	Direction	Description
txd	in	External interrupt 0. On the original 8051, this input is combined with P3.2. The signal is low active so that the original 8051 documentation on interrupt level and edge is still valid.
rxd_in	in	Input of the UART. On the original 8051, this input is combined with P3.0.
rxd_out	out	In UART mode 0, the pin RXD acts as a bidirectional pin. The signals rxd_in and rxd_out are usually connected to the same pin of the FPGA. The rxd_out line need to be buffered using a tri-state driver.
rxd_oe	out	Enable signal for output driver for the rxd_out line.

4.7 Internal data bus

The internal data bus of the QUIC_8051 consists in fact of two address busses, two data busses, and the appropriate enable signals. One address/data bus pair is for writing the SFRs, the other address/data bus pair is for reading the SFRs.

Signal	Direction	Description
d_wr_addr[7:0]	out	D-segment write address.
d_wr_data[7:0]	out	D-segment write data. A SFR connected to the write data bus must register the data on the rising edge of the global clock when the address of the SFR is hit and when the signal d_wr is active.
d_wr	out	D-segment write enable signal.
d_rd_addr[7:0]	out	D-segment read address.
d_rd_data[7:0]	in	D-segment read data.
d_rd	out	D-segment read signal. This signals is only necessary if a peripheral has to know that one of its SFRs has been read.
d_rd_rmw	out	D-segment read signal for read-modify-write-instructions. This signal is issued instead of d_rd if a SFR is read with a read-modify-write instruction like ANL or INC. This signal is necessary to properly reconstruct the behavior of the ports of the original 8051.

4.8 Internal state

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

Signal	Direction	Description
idle	out	Shows the state of the bit IDL in the SFR PCON. If the signal is 1, the CPU is in idle mode. If a peripheral supports such a mode as well, it can be switched to it. The signal is set to 0 when an interrupt or reset ends the idle mode.

5 Details and deviations

The following paragraphs explain very special aspects of the QUIC_8051.

5.1 Instruction execution

The number of machine cycles taken for the execution of instructions is identical to that in the original 8051. However, on the QUIC_8051 a machine cycle takes 2 clocks. Note that the peripherals are not speeded up. For example, the prescaler factor for timers 0 and 1 is 12.

5.2 Clock edges

The design of the QUIC_8051 is strictly synchronous using only the rising clock edge. However, the two output signals x_rd and x_wr are generated with the falling edge of the clock in order to maintain functional timing compatibility with the original 8051. In case of an X-segment read, the data read is captured with the falling clock edge in the mid of the second clock cycle.

5.3 SFRs

In the core of QUIC_8051, all SFRs of the Intel micro controller derivative 8052 are implemented with the exception of P0, P1, and P3.

5.4 PCON

In the SFR PCON, all 8 bits are implemented and can be written and read. However, only the bits IDL und SMOD have a real function associated to them. The bit PD has no function as FPGAs usually do not have the possibility to switch into a power down mode.

5.5 PSW.1

The bit PSW.1 is implemented as a simple read/write bit as it is in other actual 8051 implementations. In the original 8051, however, this bit is reserved.

5.6 Disable interrupts on accesses to SFRs IE or IP

In the original Intel 8051 the interrupts are disabled for one instruction after all instructions that access the SFRs IE or IP, even if the registers are read. In the QUIC_8051, however, the interrupts are disabled only if the instruction that accessed the SFRs is a two cycle instruction, i.e. MOV IE, #data.

5.7 MOV A, ACC

The original Intel 8051 cannot execute the instruction MOV A, ACC properly. It does not cause the processor to hang but the result is unpredictable. Most of the newer derivatives do what the instructions says: nothing. But the instruction itself was defined by Intel to be a invalid instruction and so it should not be used at all.

In the QUIC_8051 the instruction MOV A, ACC causes the accumulator to reverse the bit order of all 8 bits. Bit 0 exchanges its value with bit 7, bit 1 with bit 6, and so on. This should not confuse any application, because the instruction should not be used at all.

This behavior was implemented because of two reasons. First, it is a very useful instruction. And second, a program written for an 8051 processor can test whether it runs on an QUIC_8051 by executing the instruction and checking the result.

5.8 Write to SBUF during the transmission of a byte

Under normal circumstances this should not occur. Usually, firmware checks that the transmit buffer is empty before a new byte is written into SBUF. But in case that it happens, QUIC_8051 behaves different than the original 8051. In the original 8051 the byte that is currently sent is destroyed because the shift register is really written. In the QUIC_8051, however, the byte in transmission is sent correctly. Note that in both cases only on byte is sent.

5.9 Read-modify-write-instructions

In order to be compatible to the original 8051 it is evident for the port SFRs to distinguish between read accesses and read-modify-write-accesses. Read accesses read the port pins while read-modify-write write accesses read the port registers. The instructions below are considered to be read-modify-write-instructions:

- ANL
- ORL
- XRL
- JBC
- CPL
- INC
- DEC
- MOV Px.y, C
- CLR Px.y
- SETB Px.y

Note the excerpt from the original Intel Microcontroller Handbook: It is not obvious that the last there instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

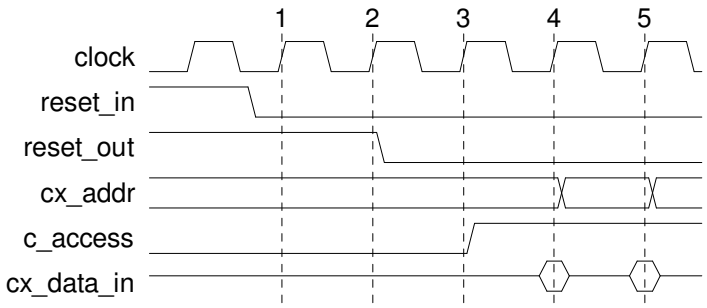
6 Functional timing diagrams

These diagrams show the behaviour of the QUIC_8051 core module at its input and output ports. The real timing of the signals depends on the FPGA used and the clock frequency.

6.1 Start of execution after reset

As already mentioned, the reset_in input is synchronized internally. The CPU starts to fetch the first instruction three clock cycles after the release from reset.

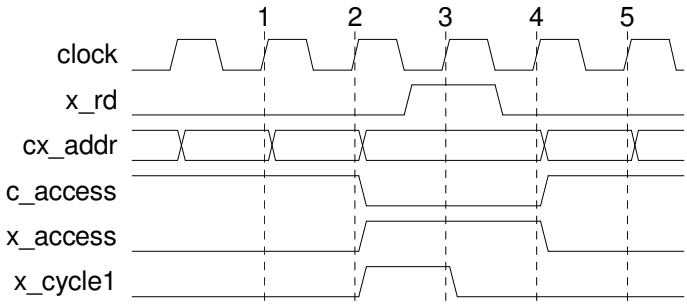
Figure 2: Start of execution after reset



6.2 MOVX instruction

For compatibility with the original 8051, the read and write strobes are generated in a special way. In the diagram below, a MOVX A,@DPTR instruction is read by the CPU on the clock edge labelled 1. After this clock edge, the cx_addr changes and the CPU pre-fetches the next instruction. This instruction is read on the clock edge labelled 2. Now a time frame of two clock cycles (12 on the original 8051) is taken to apply the X-segment address, activate and deactivate the read strobe, and switch back to the C-segment address. Note that x_rd changes after the falling edge of clock.

Figure 3: MOVX instruction

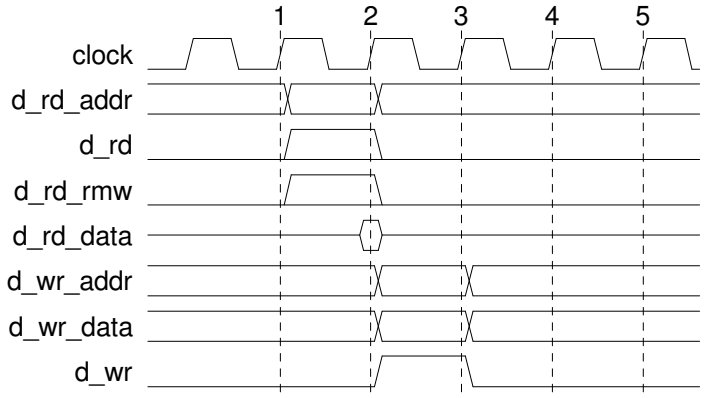


The same diagram applies to the write instruction MOVX @DPTR,A and the x_wr strobe, respectively. So, the accesses to the X-segment memory takes two clock cycles while accesses to the C-segment memory take only one clock cycle even though both accesses take place on the same busses.

6.3 Internal data bus interface

The interface for user-added D-segment peripherals is very simple. As already noted, there are separate address and data busses for write and read accesses. An access always takes one clock cycle.

Figure 3: Internal data bus interface



Note that only one of the two read enable signals d_rd or d_rd_rmw is active during a read access depending on the type of instruction. The appendix _rmw is for read-modify-write.

Second Note: Do not rely on the fact that read and write accesses don't occur in the same clock cycle. This may change in future releases.

An access on this internal data bus interfaces in run every time

- the CPU executes an instruction that accesses a direct internal address (called D-segment address)
- AND this address is greater than or equal to 0x80
- AND the address does not match one of the SFRs implemented inside the QUIC_8051 itself.

The following tables list all SFRs implemented inside the QUIC_8051 core.

SFR name	SFR address
SP	0x81
DPL	0x82
DPH	0x83
PSW	0xD0
ACC	0xE0
B	0xF0
P2	0xA0
TCON	0x88
TMOD	0x89
TL0	0x8A
TH0	0x8C
TL1	0x8B
TH1	0x8D

SFR name	SFR address
IE	0xA8
IP	0xB8
T2CON	0xC8
TL2	0xCC
TH2	0xCD
RCAP2L	0xCA
RCAP2H	0xCB
SCON	0x98
SBUF	0x99
PCON	0x87

7 Peripherals

Since the QUIC_8051 is compatible to the original Intel 8052 micro controller, the best description of the peripheral modules can be found in the original Intel Microcontroller Handbook. Numerous other books also describe the CPU and the peripheral modules very detailed. So, this datasheet focuses on these aspects the QUIC_8051 differs from the original 8052.

As already mentioned, the QUIC_8051 executes the instructions 6 times faster than the original Intel 8051 micro controller. The peripherals, however, are not speeded up. Nevertheless, there are some enhancements in the QUIC_8051's peripherals.

7.1 Port 2

The original 8051 employs some kind of open drain transistor for the port pins. High levels are output actively only for a short time and then a pull-up resistor holds the pin high. These pull-up resistors are not part of the QUIC_8051 and must be added by the user either on board level or by FPGA internal pull-up. Some FPGA families offer such a feature.
When a "1" is written to a port register that contained a "0", the original 8051 drives an active high level for two oscillator periods. The QUIC_8051 drives this high level for 1 clock period.

7.2 Timers

The inputs to the timer module are the signals:

- t0
- t1
- en_t0
- en_t1
- t2
- t2_ex

Because these signals may change asynchronously, they are synchronized inside QUIC_8051 using a two stage flip flop clocked with the rising edge of the global clock.

In the counter mode, the counter register is incremented in response to a 1-to-0 transition at the corresponding input pin. In the original 8051, the timer input signals are sampled once each machine cycle. Because one machine cycle takes 12 clocks, it takes 24 clocks to recognize a 1-to-0 transition. Thus, the maximum count rate is 1/24 of the oscillator frequency. For the QUIC_8051, the maximum count frequency is 1/2 of the clock frequency because the counter inputs are sampled every clock.

The same applies to the enable signals of timers 0 and 1 and the capture/compare input of timer 2. All signals are sampled every clock and thus lead to the appropriate action every clock.

7.3 Interrupt

The same guideline "sample every clock" applies to the interrupt signals as well. While the original 8051 needs a high level for 12 clock cycles followed by a low level for 12 clock cycles, the QUIC_8051 need these levels for only one clock cycle to detect a transition. So, combined with the accelerated instruction execution, the response time for interrupts is much shorter than in the original 8051.