LeventOzturk.com

MD5 Generator
Hardware Design

Version 1.0
May 31, 2003

Levent Ozturk
leventozturk.com

Table of Contents

List of Tables

1 Introduction

This document describes the MD5 message-digest algorithm implemented in Verilog / VHDL RTL for Hardware. The MD5 message-digest algorithm module takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It provides a direct interface for other hardware modules and an additional CPU interface to communicate directly with the CPU.

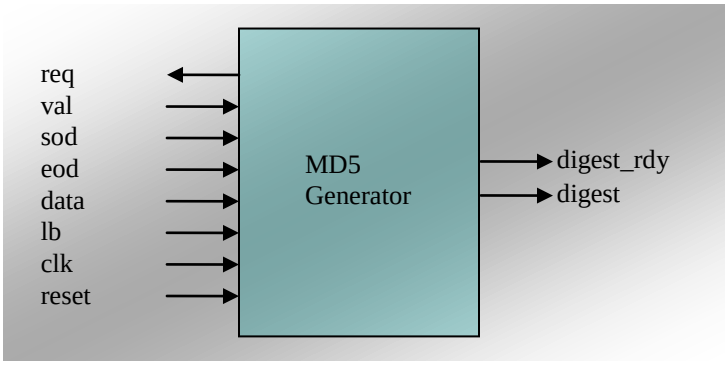

Figure 1 MD5 Generator Block Diagram

1.1 MD5 Overview

MD5 is used in many applications, including GPG, Kerberos, TLS / SSL, Cisco type 5 enable passwords, and RADIUS.
The difficulty of creating two files with the same MD5 hash should be approximately 2^{64}. The difficulty of creating a file with a specific MD5 hash should be approximately 2^{128}.

1.2 Product Summary

1.2.1 Intended Use

- MD5 message digest generation.

1.2.2 Key Features

- RFC 1321 compliant.
- Available as fully synchronous Verilog and VHDL synthesizable RTL supporting major FPGA vendor libraries.
- Supports virtually unlimited data size in bit granularity.
- Supports input handshaking for interrupted data input.
- Supports variable input data width.

1.2.3 Supported Families

- Altera® Cyclone III, Stratix III
- Xilinx® Virtex
- Generic ASIC libraries

1.2.4 Core Deliverables

- Full Version.
  Compiled RTL Simulation Model, Compliant with the Altera® Quartus II (IDE)

- Netlist Version
  - Structural VHDL and Verilog Netlists
- RTL version
  - VHDL or Verilog Core Source Code
  - Synthesis Scripts
- Verification Testbench
  - Verilog
- User Testbenches
  - Modelsim Compatible
  - VHDL and Verilog

1.2.5 Development system

- Complete MD5 (RFC1321)

1.2.6 Synthesis and Simulation Support

1.2.7 Verification and Compliance

1.3 File Structure

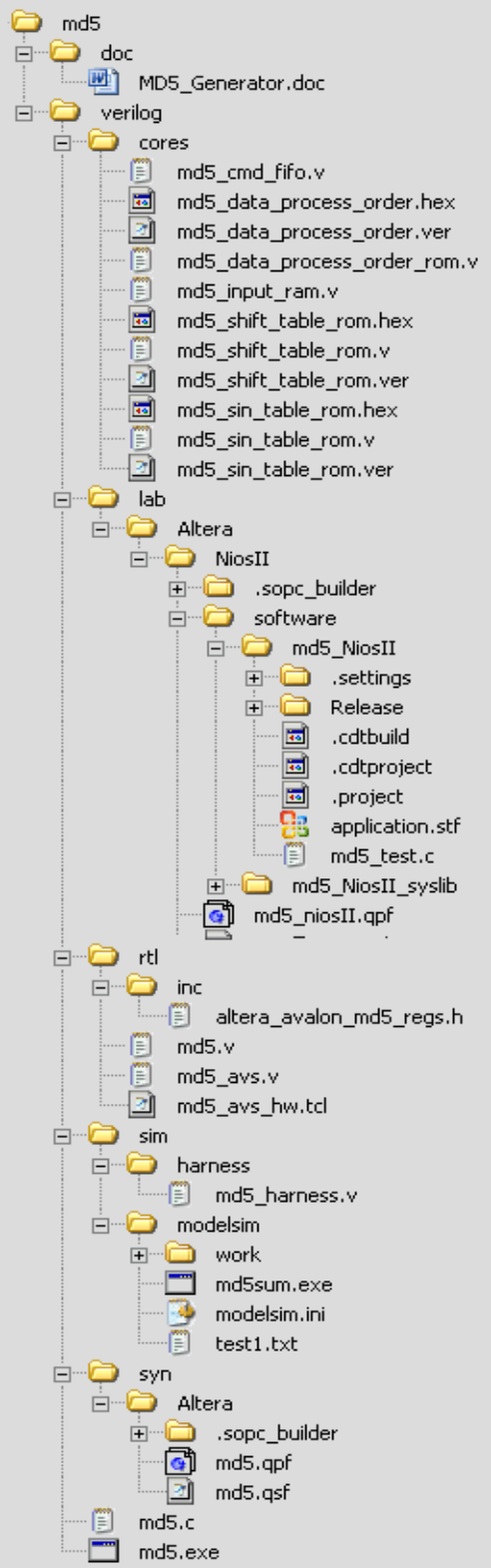

Figure 2 MD5 Generator File Structure

2 Functional Description

The MD5 Generator sub module functionality is described in this section. The MD5 Generator functional block diagram is shown in Figure 4.

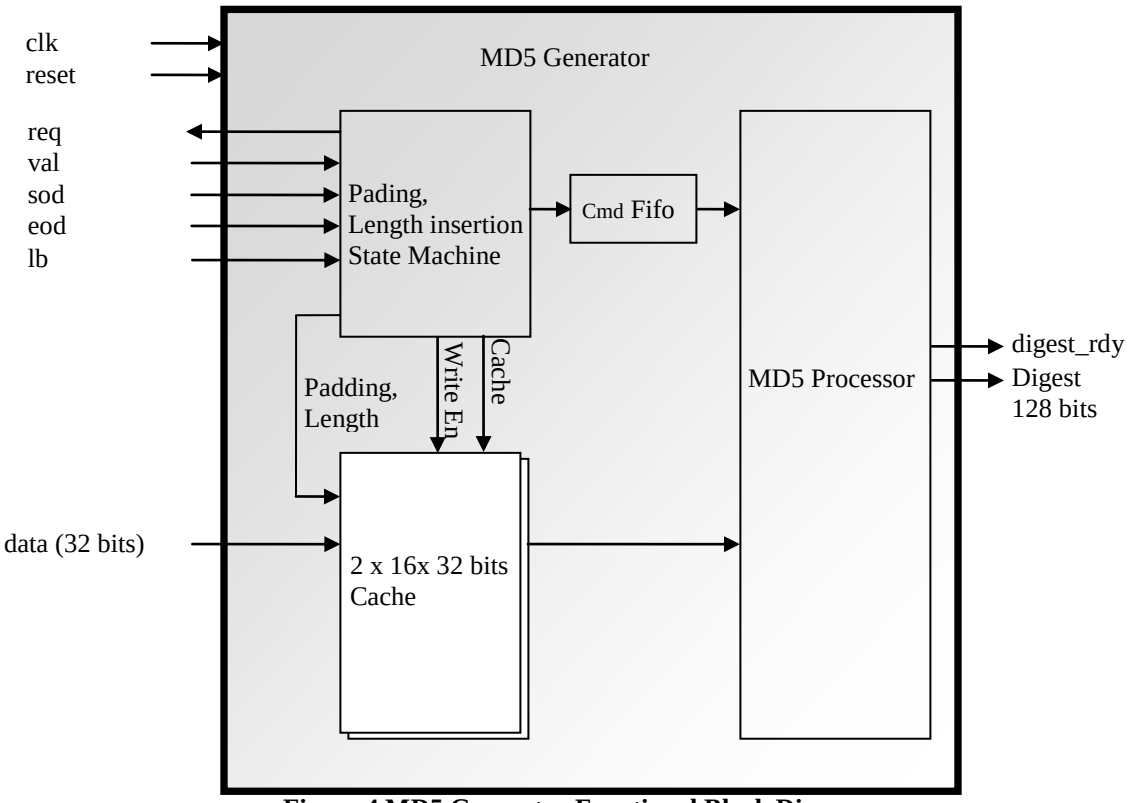

Figure 4 MD5 Generator Functional Block Diagram

The MD5 generator contains two input cache to allow parallel processing to accelerate the overall speed. This also allows the input data stream no to be forced to be continuous. The internal logic only starts to process a block when it is complete.

The input state machine adds the padding required by the RFC MD5, adds the computed size for the input data to the end of data and selects the input cache for storing the input data.

Command FIFO allows the output processor module to function independent of input data flow.

MD5 processor implements the MD5 algorithm functionality to compute A, B, C, D words defined by the RFC 1321. The final result is reported as the output of the module.

For every 16 word, 66 clocks of processing time are required.

SOD has priority over EOD. It resets the input and the initial values of words A, B, C, D.

Lb input is valid only when val and eod are asserted and indicates the last valid bit within the last word. The MD5 generator waits for the val signal to start processing. When the val signal is asserted, req signal is asserted to indicate the data is used and a new data is expected. The MD5 generator resets its internal status with every asserted sod signal.

3 Detailed Design

3.1 Input State Machine

This state machine controls the input data flow. It selects the available cache which is not being used by the MD5 processor and writes the input data into this cache. When 16 word is complete, a command is sent to the MD5 processor through the FIFO. If the other cache is released by the MD5 processor, the state machine switches the cache and continues accepting new data from the input.

If the eod is asserted, the sate machine includes the padding bits required by the RFC 1321. The padding bits completes the last bock to be 16 word length leaving room for the length field to e inserted.

After the padding, the length field is inserted.

3.2 Input Cache

His cache is used to store input data and process the previous 16 word block in parallel. While one cache is used by the input state machine, data in the other cache is processed. Once the processing is complete the cache is released. The state machine switches the input cache, when the other is released.

3.3 MD5 Processor

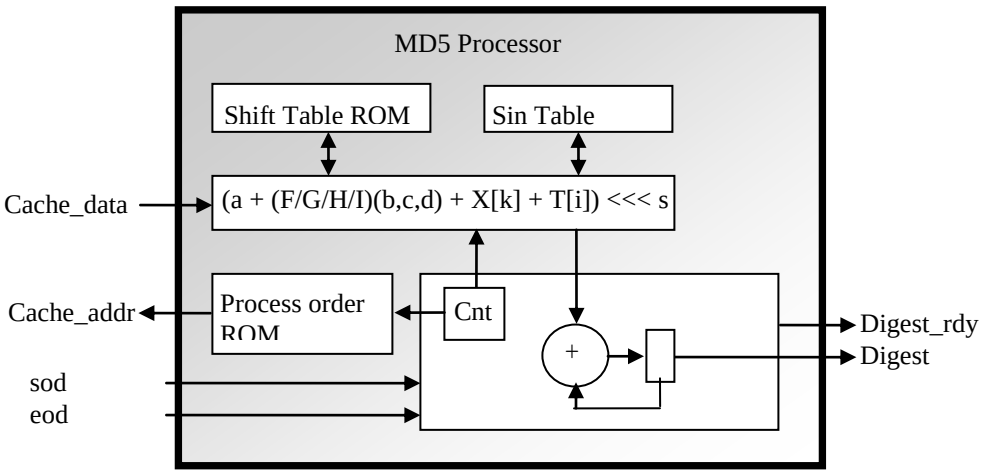

Figure 5 MD5 Processor Block Diagram

The MD5 processor receives the padded and length added input data and applies the algorithm defined in the RFC 1321 to generate the four A, B, C, D words as output.
The MD5 processor checks for the Cmd FIFO and when there is a command in the FIFO, it starts processing the corresponding cache. If sod signal is asserted the processor resets its computation values. If an eod is detected the last computed values are sent to the output. The digest value is valid when the digest_rdy is asserted.

The processing of every block takes 64 clock cycles.

3.4 Resets

There is only one asynchronous reset input used by the module

3.5 Interrupts

There is no interrupt used by the module. However digest_rdy signal can be used as an interrupt source.

leventozturk.com
(323) 988-0345

4 I/O Signal Descriptions

4.1 MD5 Generator Module

Table 1 MD5 Generator I/O Signal Descriptions

Name	Direction	Description
reset	I	Global reset
clk	I	Global clock
data[31:0]	I	Data
val	I	Data valid
sod	I	Start of Data
eod	I	End of Data
lb[4:0]	I	Last Bit
req	O	MD5 module request
digest_rdy	O	Digest result is ready
digest[127:0]	O	Digest result.

4.2 MD5 Generator Avalon Slave Module

This module contains the MD5 Generator module and registers to allow SW to control the MD5 Generator module through Avalon Slave standard interface.

Table 2 MD5 Generator Avalon Slave I/O Signal Descriptions

Name	Direction	Description
avs_s0_reset_n	I	Global reset
avs_s0_clk	I	Global clock
avs_s0_address[3:0]	I	Register Address
avs_s0_read	I	Register Read command
avs_s0_write	I	Register write command
avs_s0_writedata	I	Register write value
avs_s0_readdata	O	Register read value
avs_s1_irq	O	MD5 slave interrupt

5 Resource Usage

Table MD5 Generator resource utilization

Device	Logic Cells			Memory		DSP	PLL	Utilization
	Combinatorial	Sequential	Total	Bits	Blocks			%
Cyclone III EP3C25	3725	623	3890	3664	5 M9K	0	0	15

5.1 Clocks

The MD5 Generator module runs on a single cock domain. The clk input provides the clock source. The timing report for various vendors and parts are listed in Table 14.

Table 3 MD5 Generator Clocks

Name	Frequency	Resource	Description
clk	130 MHz	Global	Common clock

5.2 LE

The resource usage report for various vendors and parts are listed in Table xxx.

6 AC Characteristics

Power
Load
Timing (setup, hold)

leventozturk.com
(323) 988-0345

7 Software Interface

This section describes the register map of the MD5 module provided as a software interface.
Reserved bit fields return '0' in a read operation. A read operation to an undefined address will return '1' in every bit. Supported Register Types are listed below.

R/O: Read Only. This type of register can be read only. A write operation will have no affect regardless of the write value.
R/WC: Read / Write to Clear. This type of register can be read. A write operation with '1' in any bits will clear the corresponding bit in the register. The bits that are set '0' will have no affect in the corresponding bits on a write.
R/W: Read / Write. This type of register can be written and read.

7.1 Generic Register Interface

The MD5 Generator does not contain any configuration, control or status registers.

7.2 NiosII Avalon Interface

NiosII is a soft CPU provided by Altera® for Altera® FPGA family. The MD5 Generator can be used as a peripheral through the NiosII Avalon MM Slave interface.

The MD5 Generator module provides a container module to provide a software interface for any processor. It implements Altera® Avalon Memory Mapped Slave type interface. This interface can automatically be integrated to any NiosII application using SOPC tool. The SW can use MD5 HW module to generate the MD5 result of any bit stream. This reduces the SW processor load and accelerates the MD5 Generation.

The order of the operation I listed below.
The SW checks for the Digest_req field of the STATUS register is set. This indicates the HW is ready to process new 16 blocks of 32 bit word data.
The SW clears the history register.
The SW sets the SOP field of the control register.
The SW writes the first 32 bit word.
The SW clears the SOP field of the control register.
The SW starts writing 32bit words into the Data register. Every write will initiate the HW to latch new data and store into its buffers.
Very time a 16 block of 32 bit data is written, the SW should check the Digest_req field of the STATUS register is set. If it is not set, the SW should wait until it is set again.
When the last 32 bit is reached, The SW sets the EOP and LB fields in the control register, and then writes the last 32 bit word in to the data register. This last 32 bit is not required to be a complete 32 bit data. LB filed determines how many bits of the last 32 bit are valid.
The SW clears the EOP and LB field, and checks for the Digest_done field in the history field..
When the Digest_done field is asserted, the SW can read the Digest0-3 registers to fetch the MD5 result.
The SW can either loop reading the history register or set an interrupt to be notified when the Digest_done field is set.

Table 4 MD5 Avalon Slave Register Map

Name	Type	Address	Description
MD5_HISTORY	R/WC	0x0000	History of MD5 events
MD5_INT_ENB	R/W	0x0001	Enables the status register bits to cause interrupt
MD5_STATUS	R/O	0x0002	Current event status
MD5_CTRL	R/W	0x0003	Control register
MD5_DATA	R/W	0x0004	Data input register
MD5_DIGEST0	R/W	0x0008	Digest output. Most significant 32 bits
MD5_DIGEST1	R/W	0x0009	Digest output
MD5_DIGEST2	R/W	0x000a	Digest output

MD5_DIGEST3	R/W	0x000b	Digest output Least significant 32 bits

7.2.1 HISTORY Register

This register holds the historical values of the status register.

Table 5 HISTORY Register

Field	Bits	Reset	Type	Description
Reserved	31:2			
Digest_done	1	0x0	R/WC	MD5 digest output has a valid result.
Digest_req	0	0x0	R/WC	MD5 Module is ready to receive new data.

7.2.2 INT_ENB Register

This register enables or disables the corresponding fields in the status register to cause an interrupt through avs_s1_irq output.

Table 6 INT_ENB Register

Field	Bits	Reset	Type	Description
Reserved	31:2			
Digest_done	1	0x0	R/W	MD5 digest output has a valid result.
Digest_req	0	0x0	R/W	MD5 Module is ready to receive new data.

7.2.3 STATUS Register

This register holds the current status of the MD5 module.

Table 7 STATUS Register

Field	Bits	Reset	Type	Description
Reserved	31:2			
Digest_done	1	0x0	R/O	MD5 digest output has a valid result.
Digest_req	0	0x0	R/O	MD5 Module is ready to receive new data.

7.2.4 CTRL Register

This register allows the SW to control the MD5 Generator module.

Table 8 CTRL Register

Field	Bits	Reset	Type	Description
Reserved	31:7			
Sop	6	0x0	R/W	Start of message.
Eop	5	0x0	R/W	End of message. Last word to be processed.
Lb	4:0	0x0	R/W	Valid with EOP. Points the last bit of the message in the last word.

7.2.5 DATA Register

This register is used by SW to send message data to MD5 Generator. Every write operation to this register causes the data in this register and the control register values to be latched by the MD5 module.

Table 9 DATA Register

Field	Bits	Reset	Type	Description

Data	31:0	0x0	R/W	Message input to the MD5.

7.2.6 DIGEST0 Register

This register stores the 32bits of the digest result value.

Table 10 DIGEST0 Register

Field	Bits	Reset	Type	Description
Digest0	31:0	0x0	R/W	Bits [127:96] of the MD5 result.

7.2.7 DIGEST1 Register

This register stores the 32bits of the digest result value.

Table 11 DIGEST1 Register

Field	Bits	Reset	Type	Description
Digest1	31:0	0x0	R/W	Bits [95:64] of the MD5 result.

7.2.8 DIGEST2 Register

This register stores the 32bits of the digest result value.

Table 12 DIGEST2 Register

Field	Bits	Reset	Type	Description
Digest2	31:0	0x0	R/W	Bits [63:32] of the MD5 result.

7.2.9 DIGEST3 Register

This register stores the 32bits of the digest result value.

Table 13 DIGEST3 Register

Field	Bits	Reset	Type	Description
Digest3	31:0	0x0	R/W	Bits [31:0] of the MD5 result.

8 Simulation

MD5 simulation environment is implemented using Modelsim tools and scripts.

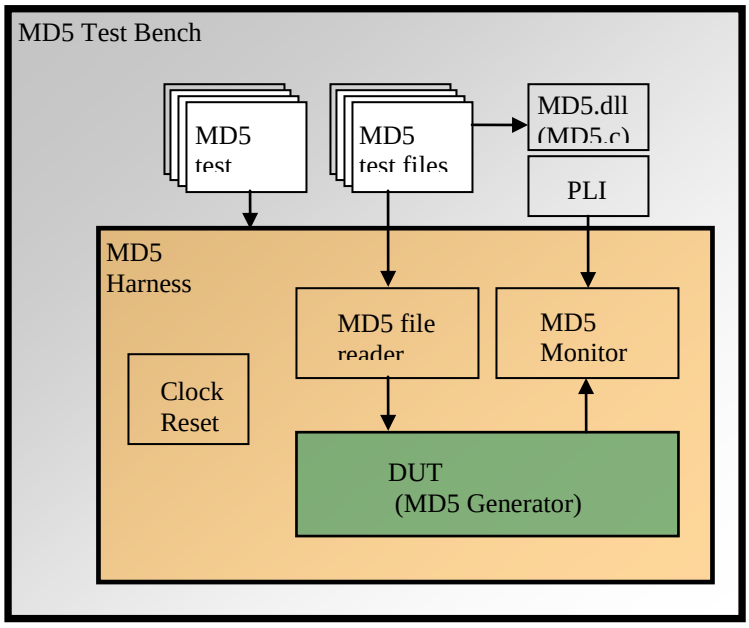

Figure 6 MD5 Testbench Block Diagram

8.1 Test Bench

Testbench module is the global container for test cases required external utilities and the harness.

8.2 Harness

Harness module provides the required hardware inputs for the DUT and provides utilities to access and monitor the DUT.

MD5 File reader reads a specified file and feeds it to the DUT as a data stream driving the entire necessary signals such as val, sod, eod, lb and data.

MD5.dll is compiled externally as a PLI source. MD5.c is the C source that generates MD5 sum in software and returns it to the MD5 monitor through a callback function.

MD5 Monitor uses external MD5 software to get MD5 result of the specified file and compares it to the MD5 result received form the DUT and reports any error.

8.3 Test Cases

Test cases are individual functions that test a specific feature or feature set of the DUT. The test cases for MD5 Generator are listed in Table 14 Simulation Test Cases.

Table 14 Simulation Test Cases

#	Test Case	Description
1	RFC 1321 sample messages	Verifies the MD5 generator against the messages where the MD5 output is known. MD5 ("") = d41d8cd98f00b204e9800998ecf8427e MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661 MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72 MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0 MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") = d174ab98d277d9f5a5611c2c9f419d9f MD5 ("12345678901234567890123456789012345678901234567890123456789012345678901234567890") = 57edf4a22be3c955ac49da2e2107b67a
2	n bit input message smaller than 1-block in size	Verify the generator can handle any small size and correct pad and length insertion in small size messages. Also if the length of the message is bigger than a block after the pad and length insertion, the generator can correctly handle to add a new complete block.
3	1 block input message	
4	Message that is between 1 -2 bock sizes.	Verifies the MD5 generator for boundary conditions.
5	Large files	Verifies the MD5 generator for large files.

8.4 Behavioral Models

8.4.1 MD5 Checksum SW

The md5 checksum (md5.dll) is the software that is used to generate the alternative MD5 result to be compared with the MD5 hardware module output. This software can be created by the following commands:
C:\gcc\gcc-3.3.1-mingw32\bin\g++ -c -IC:\Altera\71\modelsim_ae\include md5_checksum.c

C:\gcc\gcc-3.3.1-mingw32\bin\g++ -shared -o md5_checksum.dll md5_checksum.o -LC:\Altera\71\modelsim_ae\win32aloem –lmtipli
This dll file is included in the simulator as PLI extension.

8.5 Simulator Settings

The settings for various simulators and FPGA vendors are listed below.

8.5.1 Modelsim

8.5.1.1 Altera®

Parametes:
-y C:/Altera/71/quartus/eda/sim_lib
-Lf C:/Altera/71/quartus/eda/sim_lib/Altera_mf.v
+incdir+C:/levent/projects/md5/verilog/cores
+incdir+C:/levent/projects/md5/verilog/rtl
+define+MD5_SIM=1
+define+VENDOR_ALTERA=1

Commands:

```
>vlog -work work -vlog01compat \
 -y C:/Altera/71/quartus/eda/sim_lib \
 -Lf C:/Altera/71/quartus/eda/sim_lib/Altera_mf.v \
 +incdir+C:/levent/projects/md5/verilog/cores \
 +incdir+C:/levent/projects/md5/verilog/rtl \
+define+MD5_SIM=1 \
+define+VENDOR_ALTERA=1 \
 C:/levent/projects/md5/verilog/sim/harness/md5_harness.v

>vsim –c –pli hello.dll work.md5_harness

>run -all
```

9 Lab Verification

The MD5 Generator module is tested using a test board that populates an Altera Cyclone III FPGA. An application that runs on a soft NiosII processor provides the communication between the console and FPGA. The MD5 Generator lab test environment configuration is shown in Figure 7.

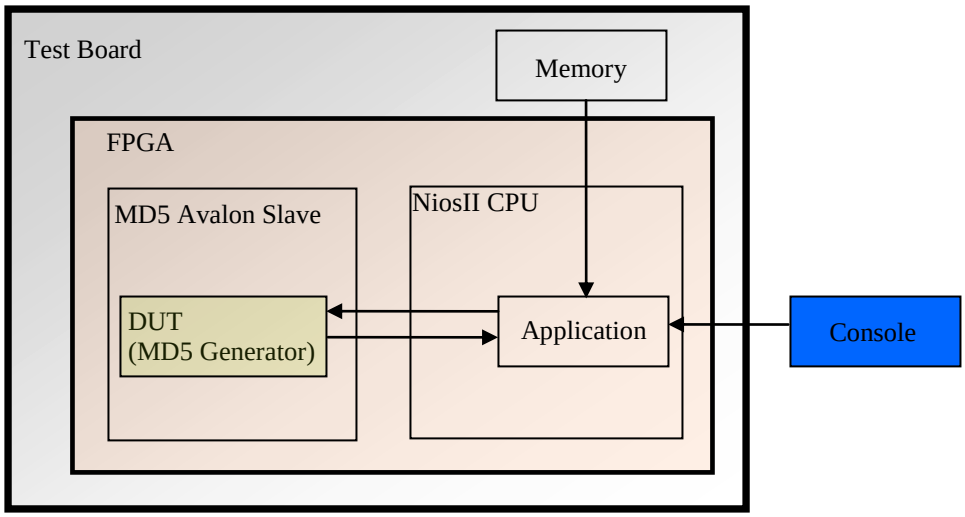

Figure 7 MD5 Lab Test Environment Block Diagram

9.1 Lab settings

Console communicates with the NiosII application over a USB cable. The user stores several test files in the memory using console. The NiosII application fetches these files and feeds to the MD5 Generator through the MD5 Avalon Slave interface. The NiosII application also calculates the MD5 results of these files with an MD5_sum function that I implemented as SW within the application. The NiosII application compares the results of both MD5 Generator and the MD5_sum function. The MD5 results and the comparison results are printed on the console.

9.2 Test Cases

Test Cases defined in Table 14 Chapter 8.3 is executed in Hardware.

10 References

RFC 1321, R. Rivest, MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992
Quartus II Handbook, v7.1, Altera®
Cyclone III Device Handbook, version 1.1, Jul 2007, Altera®
Nios II Processor Reference Handbook, *version 7.1, May 2007,* Altera®
Xilinx®