ARM® Cortex™-M0 DesignStart™

r0p0-00rel0

Release Note

4 August 2010

Release Note - ARM Cortex-M0 DesignStart r0p0-00rel0

Document confidentiality status

This document is confidential except disclosure permitted to “Designers”.

Product status

The information in this document is Final.

Web address

http://www.arm.com

Feedback

ARM limited welcomes feedback on both the product and the documentation.

Feedback on the Cortex-M0 DesignStart processor

Support is not provided with the Cortex-M0 DesignStart processor. However, if you have a question you can post it on the ARM forums at http://forums.arm.com.

Feedback on this document

If you have any comments about this document, please send email to errata@arm.com giving:

- The document title
- The document number
- The page number(s) to which your comments refer
- A concise explanation of your comments.

General suggestions for additions and improvements are also welcome.

Contents

1 PRODUCT DELIVERABLES

1.1 Product Release Status

This is the Full release of the ARM Cortex-M0 DesignStart processor at revision r0p0. These deliverables are released under the terms of the agreement between ARM and each licensee (the "Agreement"). Use by recipient of the deliverables is subject to the terms and conditions of the Agreement.

NOTE:

- These deliverables may only be used as described in the terms of your legal agreement.

1.2 About the Cortex-M0 DesignStart Processor

The Cortex-M0 DesignStart processor is a fixed configuration of the Cortex-M0 processor, enabling low cost access to Cortex-M0 processor technology by offering a subset of the full product.

The Cortex-M0 DesignStart processor is delivered as a pre-configured and obfuscated, but synthesizable, Verilog version of the full Cortex-M0 processor. As such the Cortex-M0 DesignStart processor is a fully working version of the Cortex-M0 processor, and can be used as the basis for production hardware and software designs. Details of the Cortex-M0 processor configuration used to construct the Cortex-M0 DesignStart processor are provided in section 5.1.1.

The Cortex-M0 processor is a highly deterministic, low gate count, 32-bit processor that implements the ARMv6-M architecture with zero deviation instruction determinism in zero wait-state memory systems. The Cortex-M0 processor's three-stage pipeline allows for very low area implementation whilst still being capable of achieving performance figures around 0.9 DMIPS/MHz. The Cortex-M0 processor's programmers' model is fully upwards compatible with the Cortex-M1, Cortex-M3 and Cortex-M4 processors for portability.

Further details of the Cortex-M0 processor are contained in the Technical Reference Manual (TRM) document and other technical publications documents, see section 6.1.

2 INSTALLATION

2.1 Introduction

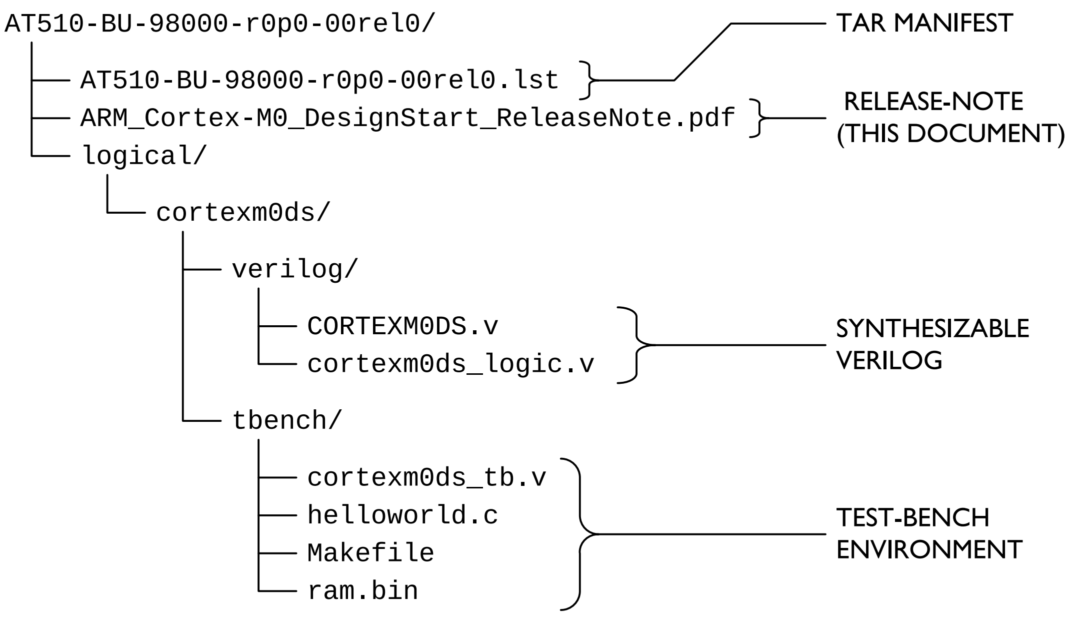
The Cortex-M0 DesignStart intellectual property (IP) deliverables are provided as a single zipped tar file. The following instructions cover Unix-like operating systems only.

2.1.1 Unpacking the Deliverables

The Cortex-M0 DesignStart processor is distributed as a single zipped tar file bundle containing two sub-deliverables, AT510-DC-80001 (this release note) and AT510-MN-80001 (the synthesizable Verilog and test-bench). Once the zipped tar file has been downloaded, it should be unpacked using the GNU gtar utility via the following command:

```
% gtar xzvf AT510-BU-98000-r0p0-00rel0.tgz
```

The content of the zipped tar file is as follows:



```
AT510-BU-98000-r0p0-00rel0/
  ├── AT510-BU-98000-r0p0-00rel0.lst
  ├── ARM_Cortex-M0_DesignStart_ReleaseNote.pdf
  └── logical/
        └── cortexm0ds/
              ├── verilog/
              │     ├── CORTEXM0DS.v
              │     └── cortexm0ds_logic.v
              └── tbench/
                    ├── cortexm0ds_tb.v
                    ├── helloworld.c
                    ├── Makefile
                    └── ram.bin
```

Once unpacked, the deliverables are ready to use.

3 IMPLEMENTATION AND INTEGRATION

3.1 Cortex-M0 DesignStart Processor

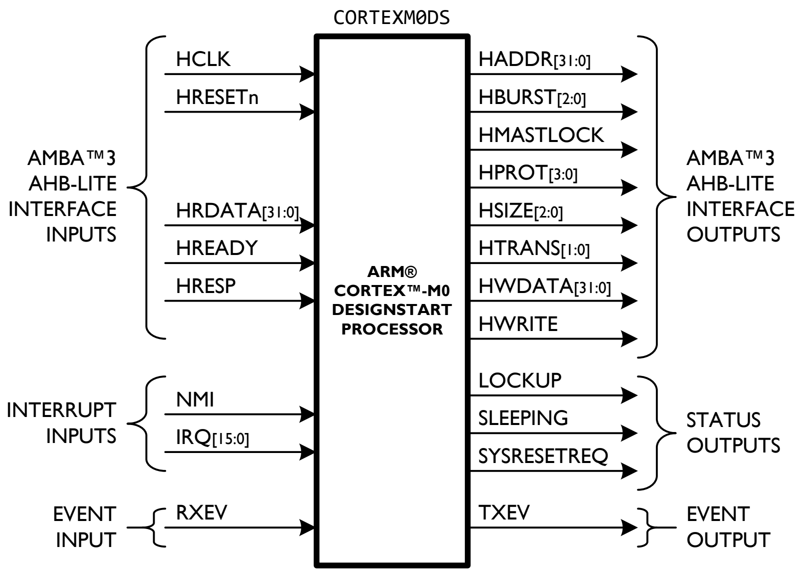


Figure 1 - CORTEXM0DS schematic

3.1.1 Overview

The Cortex-M0 DesignStart processor is contained within a top-level macro-cell module named “`CORTEXM0DS`”, and its obfuscated sub-module “`cortexm0ds_logic`”. The top-level macro-cell implements ports for a single *AMBA™3* AHB-Lite interface, interrupt and event inputs, three status outputs and an event output. A brief summary of each of the groups is provided below. Further details are included in the port list given in section 3.1.2.

AHB-Lite interface

The Cortex-M0 DesignStart processor implements a primary memory and system bus interface compatible with the AMBA™3 AHB-Lite specification. The Cortex-M0 DesignStart processor additionally uses the AHB-Lite reset and clock signals as its clock and reset source. All signals are sampled and driven relative to positive clock edges on the AHB-Lite HCLK signal.

The Cortex-M0 DesignStart processor is capable of generating four basic transaction types. These are listed in the following table.

Transaction	Access	Description
`HTRANS[1:0] = 2’b00`	IDLE	The processor does not wish to perform any transaction.

`HTRANS[1:0] = 2’b10`
`HPROT[0]    =  1’b0`
`HSIZE[1:0]  = 2’b10`
`HWRITE      =  1’b0`
	FETCH	The processor wishes to perform an instruction fetch. Instructions are fetched 32-bits at a time from memory. If required, the processor internally buffers and manages the extraction of two 16-bit instructions.

Transaction	Access	Description

`HTRANS[1:0] = 2'b10`
`HPROT[0] = 1'b1`
`HSIZE[1:0] = 2'b00`
	BYTE	The processor wishes to perform an 8-bit data access resulting from an LDRB, LDRSB or STRB instruction. Load instructions will drive the HWRITE signal LOW; store instructions will drive the HWRITE signal HIGH.

`HTRANS[1:0] = 2'b10`
`HPROT[0] = 1'b1`
`HSIZE[1:0] = 2'b01`
	HALF-WORD	The processor wishes to perform a 16-bit data access resulting from an LDRH, LDRSH or STRH instruction. Load instructions will drive the HWRITE signal LOW; store instructions will drive the HWRITE signal HIGH.

`HTRANS[1:0] = 2'b10`
`HPROT[0] = 1'b1`
`HSIZE[1:0] = 2'b10`
	WORD	The processor wishes to perform a 32-bit data access resulting from an LDR, LDM, POP, STR, STM or PUSH instruction, or as part of exception entry or return. Loads will drive the HWRITE signal LOW; stores will drive the HWRITE signal HIGH.

Table 1: Processor AHB-Lite transaction types

The AHB-Lite interface of the Cortex-M0 DesignStart processor always operates in little-endian format. All transactions are always naturally aligned. The active byte lanes for HRDATA and HWDATA and their correspondence to the source/destination register (Rd) within the Cortex-M0 processor are given in the following table. Byte lanes marked with "-" should be ignored for write-transactions and will be ignored by the processor for read transactions.

Address-phase:		**Data-phase:**			
HSIZE [1:0]	**HADDR [1:0]**	**HxDATA [31:24]**	**HxDATA [23:16]**	**HxDATA [15:8]**	**HxDATA [7:0]**
00	00	-	-	-	Rd[7:0]
00	01	-	-	Rd[7:0]	-
00	10	-	Rd[7:0]	-	-
00	11	Rd[7:0]	-	-	-
01	00	-	-	Rd[15:8]	Rd[7:0]
01	10	Rd[15:8]	Rd[7:0]	-	-
10	00	Rd[31:24]	Rd[23:16]	Rd[15:8]	Rd[7:0]

Table 2: Processor AHB-Lite read/write data byte lanes

The memory properties for the Cortex-M0 processor are specified by the ARMv6-M architecture and are fixed for a given address. The following table lists the HPROT[3:2] mapping from HADDR and indicates the recommended usage of each location.

HADDR[31:0]	Type	HPROT[3:2]	Recommended usage
`32'hF0000000 – 32'hFFFFFFFF`	Device	01	None.
`32'hE0000000 – 32'hEFFFFFFF`	Reserved	-	Maps to the processor's internal peripherals, for example NVIC.
`32'hA0000000 – 32'hDFFFFFFF`	Device	01	Peripherals.
`32'h80000000 – 32'h9FFFFFFF`	Normal (write-through)	10	Off chip RAM.
`32'h60000000 – 32'h7FFFFFFF`	Normal (write-back write-allocate)	11	Off chip RAM.
`32'h40000000 – 32'h5FFFFFFF`	Device	01	Peripherals.
`32'h20000000 – 32'h3FFFFFFF`	Normal (write-back write-allocate)	11	On chip RAM.
`32'h00000000 – 32'h1FFFFFFF`	Normal (write-through)	10	Program code.

Table 3: Processor memory map attributes

Interrupts

The Cortex-M0 DesignStart processor provides support for one non-maskable interrupt (NMI) and up to sixteen configurable-priority interrupts. All interrupt signals are active high and the processor supports pending of interrupts via both level and single-cycle (pulse) signaling.

NMI is always the highest priority exception and will cause the processor to pre-empt anything currently running, excluding another NMI handler.

IRQ[15:0] map to the prioritizable interrupts fifteen down to zero. Each of these interrupts can be individually enabled and prioritized with respect to each other via software.

All interrupt inputs are considered to be synchronous by the processor. If connecting to asynchronous interrupt sources, care should be taken to ensure that appropriate synchronization is used so as to address metastability issues.

Status Outputs

The Cortex-M0 DesignStart processor provides three additional status outputs.

LOCKUP indicates that the processor is in the Lockup state as defined in the ARMv6-M Architecture Reference Manual. System designers can choose to use this signal to inform a watchdog timer that the processor is in a likely undesirable state.

SYSRESETREQ is an active high signal that mirrors the state of the software accessible SYSRESETREQ bit of the Application Interrupt and Reset Control Register (AIRCR). This signal is designed to inform the system reset controller that software wishes to reset the entire system. Care should be taken to ensure that this signal is registered before use; driving the HRESETn signal LOW will cause the SYSRESETREQ signal to become asynchronously reset LOW.

SLEEPING is an active high signal that indicates the processor is in an idle state. This can occur as a result of executing either a wait-for-event (WFE) or wait-for-interrupt (WFI) instruction, or by making use of the Cortex-M0 processor's SLEEP-ON-EXIT functionality. The processor will not perform any AHB-Lite transactions, or execute any instructions whilst it is driving SLEEPING HIGH.

Event signals

TXEV is an active high signal that is pulsed HIGH for a single HCLK cycle whenever the processor executes a SEV instruction. This signal can be used by other processors or system components as a simple communication mechanism indicating that the processor has performed some operation that may be of interest, for example releasing a memory-based semaphore.

RXEV is an active high signal that the processor uses as one of the potential sources for indicating that it should wake-up from a WFE instruction.

3.1.2 Port List

The following table describes each of the ports on the CORTEXM0DS macro-cell.

Port	Dir	Description
HADDR[31:0]	OUT	AHB-Lite address-phase interface address. These signals provide the 32-bit address used to identify the memory or device being accessed by a transaction. The value on this bus is only valid if HTRANS[1] is HIGH.
HBURST[2:0]	OUT	AHB-Lite address-phase burst information. The processor only generates SINGLE type transfers. The value of this bus is always driven as 3'b000, indicating a SINGLE type transfer.
HCLK	IN	AHB-Lite interface and processor clock. All signals in and out of the processor are processed on the positive/rising edge of this clock.
HMASTLOCK	OUT	AHB-Lite address-phase locked-transaction information. The processor never generates locked transactions. The value of this signal is always driven LOW, indicating that the transaction is not part of a locked sequence.
HPROT[3]	OUT	AHB-Lite address-phase transaction cacheable information. The value of this bit is derived from the address of the transaction. The value on this signal is only valid if HTRANS[1] is HIGH.
HPROT[2]	OUT	AHB-Lite address-phase transaction bufferable information. The value of this bit is derived from the address of the transaction. The value on this signal is only valid if HTRANS[1] is HIGH.
HPROT[1]	OUT	AHB-Lite address-phase transaction privilege information. The processor always operates in a Privileged mode. The value of this signal is always driven HIGH, indicating that the transaction is PRIVILEGED.
HPROT[0]	OUT	AHB-Lite address-phase transaction data-vs-opcode information. The processor uses this signal to differentiate between instruction fetches and data-accesses generated by loads and stores. The value of this signal is driven LOW for instruction fetches and HIGH for load/store data transactions. The value on this signal is only valid if HTRANS[1] is HIGH.

Port	**Dir**	**Description**
HRDATA[31:0]	IN	AHB-Lite data-phase transaction read-data. This bus carries the read-data associated with a previous AHB-Lite address-phase from the system into the processor. The read-data is constructed of four 8-bit byte-lanes, the number and location of the valid lanes is dependent on the values of HADDR[1:0] and HSIZE[1:0] used in the address-phase, see Table 2 for details. The value on this bus must be valid for a read-transaction data-phase cycle where HREADY is HIGH and HRESP is LOW.
HREADY	IN	AHB-Lite address and data-phase ready signal. This signal indicates the acceptance of an address-phase from the processor and the completion of any current data-phase. The value of this signal must always be valid and driven HIGH to indicate that the AHB-Lite bus is either idle and accepting an address-phase, or is completing a data-phase and accepting an address-phase. After accepting an address-phase, HREADY can be driven LOW to extend the current data-phase.
HRESETn	IN	AHB-Lite interface and processor reset. This is an active low signal which, when LOW, asynchronously resets all state in the processor. This signal should be held LOW for at least two HCLK cycles, and should be de-asserted (driven HIGH) synchronously to HCLK. After reset, this signal should be held HIGH to permit normal execution.
HRESP	IN	AHB-Lite data-phase error-response signal. This signal indicates that the current data-phase is to a location which does not wish to provide valid read-data, and that the processor should take an exception if it should try to execute instructions from or perform a load or store to this location. Note that HRESP operation must comply with the requirements of AHB-Lite across two HCLK cycles. It is permissible to continuously drive the HRESP signal LOW in a system which does not wish to generate any errors.
HSIZE[2:0]	OUT	AHB-Lite address-phase transaction size information. The processor is capable of generating transactions for bytes (8-bits), halfwords (16-bits) and words (32-bits) for load/store data-transactions, and for words (32-bits) for instruction fetches. The value of this bus will be driven to 3’b000 indicating a BYTE, 3’b001 indicating a HALFWORD, or 3’b010 indicating a WORD transaction. The value on this bus is only valid if HTRANS[1] is HIGH.
HTRANS[1:0]	OUT	AHB-Lite address-phase transfer type information. The processor uses this bus to initiate an AHB-Lite transaction. All transfers are issued as non-sequential transfers (no guaranteed relationship to previous transfer address). The value on this bus is driven to 2’ b00 to indicate an IDLE cycle or to 2’ b10 to indicate a NONSEQ transfer.
HWDATA[31:0]	OUT	AHB-Lite data-phase transaction write-data. This bus carries the write-data associated with a previous AHB-Lite address-phase. The write-data is constructed of four 8-bit byte-lanes, the number and location of the valid lanes is dependent on the values of HADDR[1:0] and HSIZE[1:0] used in the address-phase, see Table 2 for details. The value on this bus is only valid for a write-transaction data-phase.

Port	Dir	Description
HWRITE	OUT	AHB-Lite address-phase transaction write-vs-read information. This signal indicates whether the processor wishes to read or write the address specified by HADDR. This signal is driven LOW for a READ transaction or HIGH for a WRITE transaction. The value on this signal is only valid if HTRANS[1] is HIGH.
IRQ[15:0]	IN	Prioritizable interrupt inputs. Each signal in this bus corresponds to an individually prioritizable interrupt. Each signal must be driven synchronously to HCLK. If a source is asynchronous, then it must be synchronized before being driven onto its associated IRQ input. If two interrupts are assigned the same priority and asserted at the same time, then the one lowest in the bus will win. Driving signal IRQ[n] HIGH for at least on HCLK cycle will cause interrupt[n] to become pending within the processor. Each signal within the bus must be valid every cycle and should be held LOW unless an interrupt should be pended.
LOCKUP	OUT	Processor in Lockup state indicator. The processor uses this signal to indicate that two or more unrecoverable errors have occurred, for example due to bus-faults, attempted execution of an unaligned memory transaction or execution of an UNDEFINED instruction, and has therefore entered Lockup state. This signal is driven LOW for cycles in which the processor is executing normally, and is driven HIGH for every cycle in which the processor is waiting in the Lockup state. This signal is valid on every cycle. Typically this signal would be connected to a watchdog type device.
NMI	IN	Non-maskable interrupt input. This signal must be synchronous to HCLK. If the source is asynchronous, for example a general-purpose input, then it must be synchronized before being driven onto the NMI signal. The NMI signal is active high. Driving this signal HIGH for at least one HCLK cycle will cause the processor to take an NMI exception. This signal must be valid on every cycle and should be held LOW unless an NMI should be generated.
RXEV	IN	Event reception port. Driving this signal HIGH for a single HCLK cycle causes the ARMv6-M Event Register inside the processor to become set. This signal can be used by the system to indicate to software that some event has occurred. Software can use the WFE instruction inside a loop to reduce system polling if the event being polled is capable of driving RXEV.
SLEEPING	OUT	Processor is idle. The processor drives this signal HIGH if it is idle and will drive this signal LOW before performing any operation. This signal is valid on every cycle.

Port	Dir	Description
SYSRESETREQ	OUT	Processor request for entire system reset. This signal is mapped to the SYSRESETREQ register inside the processor and is driven LOW during normal operation. The processor will drive this signal HIGH in response to a validated software write to the internal reset request register. This signal is reset LOW by the asynchronous HRESETn signal. The signal must therefore not be connected via combinatorial logic to the HRESETn port. This signal is valid on every cycle.
TXEV	OUT	Event transmission port. This signal will be driven high by the processor each time it executes an SEV instruction. This signal can be used by software to indicate to the system that some operation has been performed. This signal is valid on every cycle.

Table 4: Cortex-M0 DesignStart processor port descriptions

3.2 Key Integration Tasks

Integration of the CORTEXM0DS module into a system is summarized in the following seven steps:

1. Generate and connect the HCLK clock signal.
2. Generate and synchronize the HRESETn active-low reset signal.
    a. The HRESETn signal should be held for at least two HCLK cycles and de-asserted synchronously.
    b. If desired, integrate SYSRESETREQ into the reset synchronization/control circuitry.
    c. If desired, integrate LOCKUP into the reset synchronization/control circuitry.
3. Build and attach the rest of the AHB-Lite system.
    a. Note that Cortex-M0 requires accesses to 0x0 and 0x4 to provide the initial stack-pointer value and address of the reset handler.
4. Connect appropriate interrupts from peripherals to appropriate IRQ pins.
    a. Tie any unused IRQ signals LOW.
5. If required, connect the NMI input to an appropriate source.
    a. Note that NMI is capable of preempting before software initialization of the processor or C-library has completed, as such it can be desirable to add appropriate masking to the NMI signal.
    b. If the NMI signal is not used, it should be tied LOW.
6. Connect external event stimulus to the RXEV pin.
    a. Tie RXEV LOW if unused.
7. Validate that your design is functional.

3.3 Implementation

Scripts and methodology for producing a physical implementation of either the pre- or post-integrated CORTEXM0DS macro-cell are not provided as part of the Cortex-M0 DesignStart processor distribution. Successful implementation in silicon, on FPGA, or using other physical means, is the responsibility of the licensee. The CORTEXM0DS macro-cell should be compatible with most Verilog synthesis flows. The synthesizable processor is contained within the two Verilog files found in the "`logical/cortexm0ds/verilog/`" directory, namely "`CORTEXM0DS.v`" and "`cortexm0ds_logic.v`".

4 SIMULATION TEST-BENCH

4.1 Cortex-M0 DesignStart Test-Bench

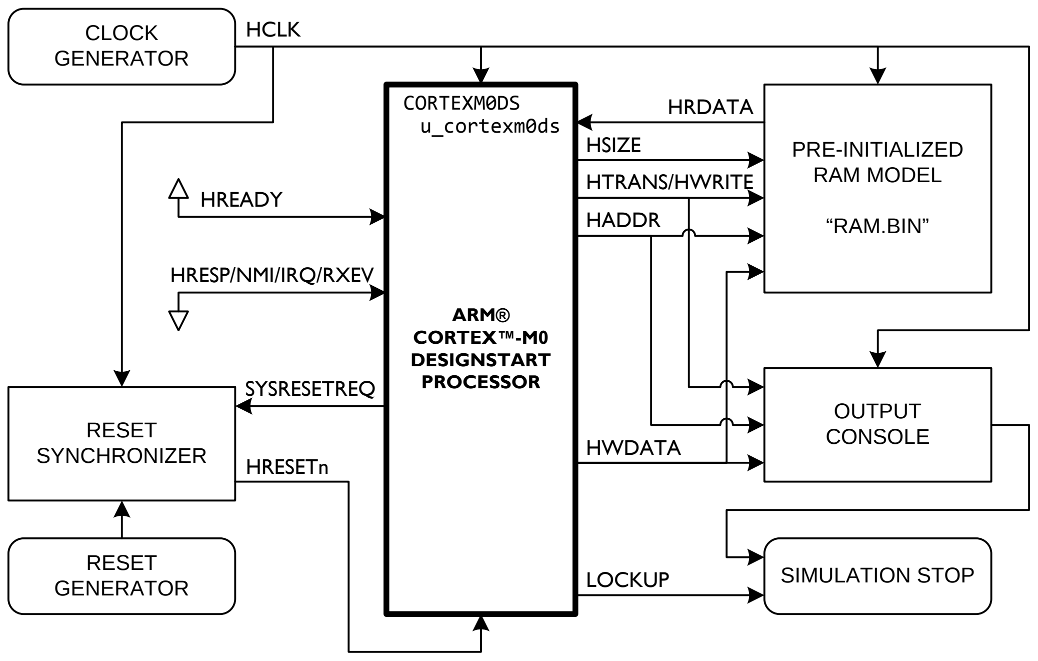


Figure 2 - Cortex-M0 DesignStart test-bench schematic

4.1.1 Overview

The Cortex-M0 DesignStart deliverables include a lightweight simulation test-bench intended to minimize the time from downloading the deliverables to being able to execute code on the processor. The test-bench deliverables consist of: a Verilog test-bench, an example program and corresponding memory-image, and a simple Makefile.

Verilog test-bench

The test-bench, shown in Figure 2 above, instantiates the CORTEXM0DS module and connects it in a minimal way to a memory model and clock and reset generators. It also provides a means of outputting information from the processor to the Verilog simulator's console output. Note that the components within the test-bench are intended for simulation purposes only, and should be replaced with synthesizable counterparts before performing any synthesis.

Simulating the test-bench is simply a matter of compiling the test-bench, "`cortexm0ds_tb.v`" with the Cortex-M0 DesignStart processor synthesizable Verilog using a Verilog simulator.

Commands for performing this operation using some commercial EDA tools under Linux are provided in the Makefile, and can be invoked simply by running "`make`". The Cortex-M0 DesignStart processor does not include a Verilog simulator, however, it should be possible to simulate the provided RTL files using any EDA tool compliant with the Verilog-2k standard (IEEE Std 1364-2001) or later. Such tools are available both commercially and from the open-source community.

Example memory image

An example memory image for the processor is provided by the “`ram.bin`” file. This is loaded by the memory model in the Verilog test-bench at the beginning of simulation. The provided memory image is for a simple “hello world” program, which uses the test-bench to write a message to the simulator’s console, and then to terminate the simulation.

The C source code for the program can be found in “`helloworld.c`”. The commands required to re-compile this using ARM’s RealView compiler are included in the Makefile, however, use of the source file, or a compiler tool-chain is not required to execute the provided program.

The Cortex-M0 DesignStart deliverables do not include a compiler tool chain. Either an evaluation or a full version of an appropriate tool-chain is required should you wish to recompile or modify the example program. Evaluation versions of ARM RealView or Keil MDK are available from www.arm.com or www.keil.com respectively. Alternative compiler tool-chains for the Cortex-M0 processor are available both commercially and from the open-source community; however, these can require modification of the provided source code.

Makefile

The Makefile contains example commands for both compiling the memory image from its C source file, and additionally for performing a simulation of the test-bench and processor RTL using a number of commercial Verilog simulators running under the Linux operating system.

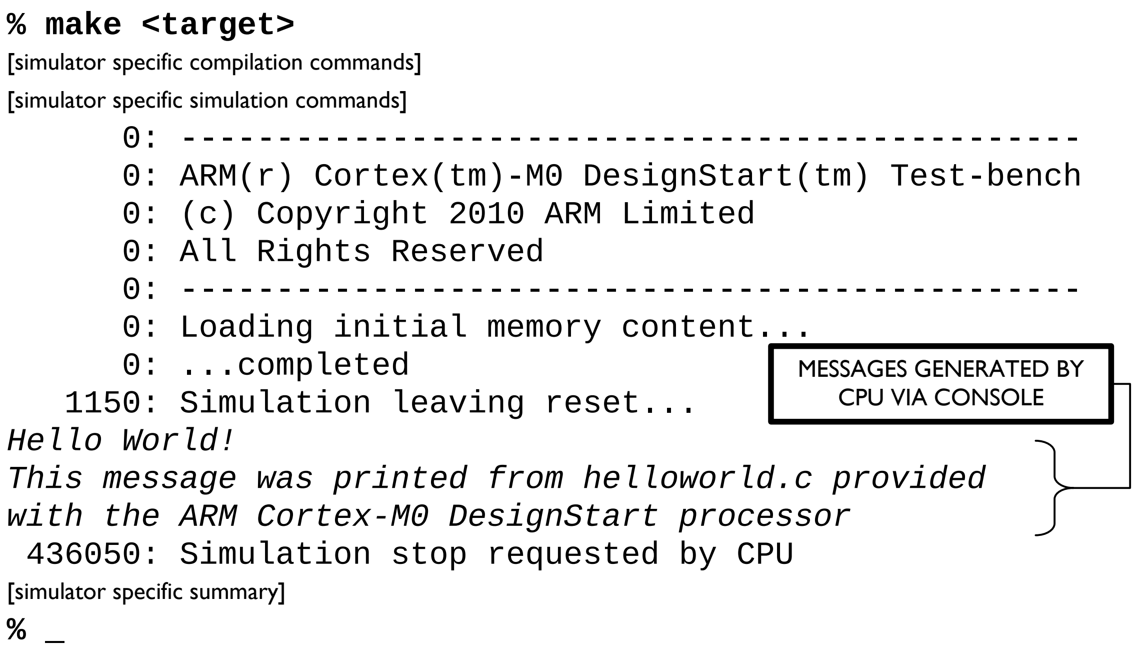
Users of alternative operating systems or Verilog simulators, or those wishing to perform simulation using a graphical-user-interface, should refer to the documentation provided by their EDA tool vendor. The relevant files for simulation are “`cortexm0ds_tb.v`”, “`CORTEXM0DS.v`” and “`cortexm0ds_logic.v`” with the top-level Verilog module being “`cortexm0ds_tb`”.

For those using Linux, and with the relevant EDA tools, compiler tools and GNU Make available (and locatable via the PATH environment variable), the Makefile provides a number of targets. The targets are available by executing “`make <target>`”, where <target> should be substituted for one of the items in the following table:

<target>	**Description**
ram.bin	Rebuilds the “`ram.bin`” file from the “`helloworld.c`” source file using ARMCC. This target can only be built if “`armcc`”, “`armlink`” and “`fromelf`” are on the PATH.
ncverilog	Builds and simulates the Verilog test-bench using Cadence NC Verilog. This can only be used if “`ncverilog`” is on the PATH.
questasim	Builds and simulates the Verilog test-bench using Mentor Graphics QuestaSim or ModelSim. This can only be used if “`vlib`”, “`vlog`” and “`vsim`” are on the PATH.
vcs	Builds and simulates the Verilog test-bench using Synopsys VCS. This can only be used if “`vcs`” is on the PATH.

Table 5: Cortex-M0 DesignStart test-bench Makefile targets

Using any of the three included simulator targets from within the “`logical/cortexm0ds/tbench`” directory will build, and then simulate, the Verilog test-bench. This should produce output similar to that shown in Figure 3 below.



```
% make <target>
[simulator specific compilation commands]
[simulator specific simulation commands]
      0: ------------------------------------------------
      0: ARM(r) Cortex(tm)-M0 DesignStart(tm) Test-bench
      0: (c) Copyright 2010 ARM Limited
      0: All Rights Reserved
      0: ------------------------------------------------
      0: Loading initial memory content...
      0: ...completed
   1150: Simulation leaving reset...
Hello World!
This message was printed from helloworld.c provided
with the ARM Cortex-M0 DesignStart processor
 436050: Simulation stop requested by CPU
[simulator specific summary]
% _
```

Figure 3 - Cortex-M0 DesignStart processor test-bench simulation output

4.1.2 Debugging

Whilst the Cortex-M0 DesignStart processor does not offer any hardware debug capabilities, the provided Verilog RTL does offer visibility of key architectural state for the purposes of simulation waveform based debugging.

Several observation signals are routed from within the Cortex-M0 DesignStart processor logic level into the CORTEXM0DS level for this purpose. Table 6 below lists the signals to be found in the CORTEXM0DS level.

Signal	Description
cm0_r00[31:0] . . . cm0_r12[31:0]	ARMv6-M general-purpose register r0. . . . ARMv6-M general-purpose register r12.
cm0_msp[31:0]	ARMv6-M main stack pointer (MSP).
cm0_psp[31:0]	ARMv6-M process stack pointer (PSP).
cm0_r14[31:0]	ARMv6-M link register.
cm0_pc[31:0]	Address of instruction currently in execute.
cm0_xpsr[31:0]	ARMv6-M processor status register (combined A/E/IPSR).
cm0_control[31:0]	ARMv6-M CONTROL (stack-pointer selection).
cm0_primask[31:0]	ARMv6-M PRIMASK (prioritizable interrupt mask).

Table 6: Cortex-M0 DesignStart processor observation signals

5 LIMITATIONS OF THIS RELEASE

5.1 Introduction

To enable low cost access, the ARM Cortex-M0 DesignStart processor has limitations in comparison to the full Cortex-M0 processor. Neither the processor technology nor supporting deliverables should be used as an indicator of what is received under a full technology license of the ARM Cortex-M0 processor. This section gives an overview of the differences.

5.1.1 Cortex-M0 DesignStart Processor Configuration

The Cortex-M0 DesignStart processor offers a single configuration of that possible from the full Cortex-M0 processor. Table 7, below, illustrates the key differences, further details can be found in section 5.

Feature	Full Cortex-M0 processor	Cortex-M0 DesignStart processor
Verilog code	Commented plain-text RTL.	Flattened and obfuscated RTL.
AMBA™3 AHB-Lite interface	Master and optional slave ports.	Master port only.
ARMv6-M instruction set	— Identical ARMv6-M instruction set support. —	
Fast/small multiplier options	Fast single or small 32-cycle.	Small 32-cycle multiplier only.
Nested vectored interrupt controller (NVIC)	1 to 32 interrupt inputs.	16 interrupt inputs only.
Wake-up Interrupt Controller	Optional.	None.
Architectural clock gating	Optional.	None.
24-bit system timer (SysTick)	Optional reference clock.	Core clock source only.
Hardware debugger interface	Optional Serial-Wire or JTAG.	None.
Hardware debug support	Optional single step with up to four breakpoints, up to two watchpoints and PC sampling.	None.
Low-power signaling and domains	Optional state-retention power domains and power control signaling.	SLEEPING, TXEV and RXEV signaling only.

Table 7: Cortex-M0 product feature comparison

5.1.2 Benefits of the Full Cortex-M0 Processor

The full Cortex-M0 processor provides additional capabilities above that of the Cortex-M0 DesignStart processor.

Low power optimizations

The full Cortex-M0 processor is designed for deployment in a multi-power domain system to maximize static power efficiency, featuring a minimal Wake-up Interrupt Controller. It also makes extensive use of architectural clock gating to minimize dynamic power. For simplicity these are amongst the technologies not included in the Cortex-M0 DesignStart processor.

SysTick timer source

The full Cortex-M0 processor supports a SysTick timer clocked via the processor clock source or via an alternative external reference clock. This offers the opportunity to provide more accurate timing and/or lower power consumption. The Cortex-M0 DesignStart processor variant only offers clocking of the SysTick timer from the processor clock.

Debug support

The full Cortex-M0 processor supports the use of an external hardware debugger to facilitate the development of applications. Connection is possible via either Serial-Wire or JTAG interfaces and provides a host of debug functionality. Both connections provide the ability to access all AHB-Lite connected slaves, including RAM, whilst the processor is running, as well as providing full halting-mode debug. Halting-mode debug allows all processor registers to be examined and modified, and can be configured to provide up to four hardware breakpoints and two hardware watchpoints. Unlimited software breakpoints are possible via the BKPT instruction.

The Cortex-M0 DesignStart processor implementation does not provide any debug capabilities.

Hardware multiplier options

The ARMv6-M architecture provides a MULS instruction capable of performing a 32-bit x 32-bit multiply, generating a 32-bit result. The full Cortex-M0 processor product allows implementation time selection between a fast single-cycle implementation and a low-area 32-cycle implementation. The Cortex-M0 DesignStart processor only provides the low-area option.

Jitter-free interrupt handling

The full Cortex-M0 processor provides the capability to optionally regulate the time between an interrupt-signal being asserted and having the associated exception handler being enter, thus providing zero-jitter interrupt entry. The Cortex-M0 DesignStart processor always handles interrupts as fast as possible.

5.2 Migrating to the Full Cortex-M0 Processor

The Cortex-M0 DesignStart processor is derived from the full ARM Cortex-M0 AT510-MN-22100-r0p0-03rel1 deliverable. The DesignStart version of the ARM Cortex-M0 processor offers a preconfigured subset of the features, interfaces, capabilities, power-domains, documentation and synthesis support of the full ARM Cortex-M0 processor. Licensees of the full Cortex-M0 processor have access to the plain-text Verilog RTL with the ability to configure the processor via Verilog parameters.

Licensees wishing to mirror the parameters used for the DesignStart Cortex-M0 processor should set the RAR, SMUL and SYST parameters to 1, the NUMIRQ parameter to 16 and set all other parameters to zero.

The full Cortex-M0 processor includes additional ports, not present in the Cortex-M0 DesignStart processor. The full Cortex-M0 processor can be configured to match the Cortex-M0 DesignStart processor by assigning SCLK from HCLK, tying STCALIB[25:24] and SLEEPHOLDREQn HIGH, and tying all other unused inputs LOW.

6 DOCUMENTATION

6.1 Cortex-M0 Technical Publications Documents

The following technical publications are available from http://infocenter.arm.com:

1. Cortex-M0 Technical Reference Manual - ARM DDI0432C

   The Technical Reference Manual provides details for programming the Cortex-M0 processor.
2. Cortex-M0 User Guide Reference Material - ARM DUI0467B

   The User Guide Reference Material provides a generic device user level reference document for software development on a Cortex-M0 device.

6.2 ARM Architecture Documents

The following architecture documents are available from http://infocenter.arm.com:

1. ARMv6-M Architecture Reference Manual - ARM DDI 0419B

   This is the ARM architecture which the Cortex-M0 processor complies with and provides details of the instruction set and exception model.
2. AMBA™3 AHB-Lite Specification - ARM IHI 0033A

   This is the AMBA bus standard which the Cortex-M0 processor's AHB-Lite master interface implements.

7 REVISION HISTORY

Date	Revision - Version	Description
4 August 2010	r0p0-00rel0	First Cortex-M0 DesignStart release