# **Application Note 199**

## ARM11 core simulation guidelines

Document number: ARM DAI 199A Issued: 6th December, 2007 Copyright ARM Limited 2007



#### Application Note 199 ARM11 core simulation guidelines

Copyright © 2007 ARM Limited. All rights reserved.

#### **Release information**

		C	hange history
Date	Issue	Change	
December 2007	А	First release	

#### **Proprietary notice**

Words and logos marked with  $\bigcirc$  and  $^{TM}$  are registered trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

#### **Confidentiality status**

This document is Open Access. This document has no restriction on distribution.

#### Feedback on this Application Note

If you have any comments on this Application Note, please send email to errata@arm.com giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

#### ARM web address

http://www.arm.com

### **Table of Contents**

1.	Need for initialization sequence	1
2.	Content of initialization code	1
3.	ARM1136/1156 initialization code	2
4.	ARM1176 initialization code	4

## 1. Need for initialization sequence

ARM cores are implemented using non-reset flip-flops, on logic which does not need to be reset, for area saving. This sometimes leads to problems of X-propagation in simulations, particularly at the gate level. Our recommendation for gate-level simulations is to run two-state simulation using the +2state switch available in vcs simulator. However, not all customers have access to a vcs simulator. Another alternative is to force some registers to reset at the start of the simulation.

This application note contains routines for initialisation. By taking care to initialise as much logic as possible, we can minimise gate-level simulation issues. One routine is provided for ARM1136 and ARM1156 cores and a separate routine is provided for the ARM1176 core, which has an additional secure monitor mode. This initialization sequence is only necessary in the simulation world for rtl, DSM and gate-level simulations and is not needed in the real world.

## 2. Content of initialization code

The programmer's registers are not reset and they can affect simulations in some circumstances (depending on the instruction sequence), so the recommended approach is to initialize these registers in software. The routine enters each mode and resets the programmer's registers of that mode, leaving the CPU in supervisor mode at the end, which is the reset default mode.

The two read ports of the ALU are uninitialized until used by an ALU instruction, so an ADD instruction is used to reset the A and B read ports between the register bank and the ALU. This prevents propagation of X-states into the load-store unit (LSU).

The return stack is three entries deep and is not reset. The routine sets the Z bit and initialises the return stack entries by the use of three BL instructions.

## 3. ARM1136/1156 initialization code

```
;* The confidential and proprietary information contained in this file may
;* only be used by a person authorised under and to the extent permitted
;* by a subsisting licensing agreement from ARM Limited.
;*
;*
           © COPYRIGHT 2007 ARM Limited.
;*
               ALL RIGHTS RESERVED
;*
;* This entire notice must be reproduced on all copies of this file
;* and copies of this file may only be made by a person if such person is
;* permitted to do so under the terms of a subsisting license agreement
;* from ARM Limited.
;*
;
; Code to initialize ARM1136 or ARM1156
;
; --- Standard definitions of mode bits and interrupt (I & F) flags in PSRs
Mode_USR
             EQU
                    0x10
Mode_FIQ
             EQU
                    0x11
Mode_IRQ
             EQU
                    0x12
                    0x13
Mode_SVC
             EQU
                    0x17
Mode ABT
             EQU
Mode UND
             EOU
                    0x1B
Mode_SYS
             EQU
                    0x1F
I_Bit
             EOU
                    0x80 ; when I bit is set, IRQ is disabled
F_Bit
             EOU
                    0x40 ; when F bit is set, FIQ is disabled
        Header Code
                      , CODE
AREA
       ENTRY
          в
               Start
reset
; Exception vectors should be added here
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
Start
; On reset the ARM core is in SVC mode
          MOV
               r0, #0
          MOV
               r1, #0
          MOV
               r2, #0
          MOV
               r3, #0
          MOV
               r4, #0
          MOV
               r5, #0
          MOV
               r6, #0
          MOV
               r7, #0
          MOV
               r8, #0
          MOV
               r9, #0
          MOV
               r10, #0
          MOV
               r11, #0
               r12, #0
          MOV
          MOV
               r13, #0
```

```
MOV
                  r14, #0
; Enter each mode in turn and initialize the registers specific to it
                  CPSR_c, #Mode_FIQ:OR:I_Bit:OR:F_Bit
            MSR
            MOV
                  r8, #0
            MOV
                  r9, #0
                  r10, #0
            MOV
            MOV
                  r11, #0
            MOV
                  r12, #0
            MOV
                  r13, #0
            MOV
                  r14, #0
                  CPSR_c, #Mode_IRQ:OR:I_Bit:OR:F_Bit
            MSR
            MOV
                  r13, #0
            MOV
                  r14, #0
                  CPSR_c, #Mode_ABT:OR:I_Bit:OR:F_Bit
            MSR
                  r13, #0
            MOV
                  r14, #0
            MOV
            MSR
                  CPSR_c, #Mode_UND:OR:I_Bit:OR:F_Bit
                  r13, #0
            MOV
                  r14, #0
            MOV
; System mode shares user mode registers
            MSR
                  CPSR_c, #Mode_SYS:OR:I_Bit:OR:F_Bit
            MOV
                  r13, #0
            MOV
                  r14, #0
; Leave core in SVC mode (if necessary)
            MSR
                  CPSR_c, #Mode_SVC:OR:I_Bit:OR:F_Bit
; Initialise ALU paths
            ADD
                  r0, r1, r2
; Initialise return stack - requires branch prediction enabled (Z=1)
            MRC
                  p15, 0, r0, c1, c0, 0 ; read control reg
                  r0, r0, #0x800
                                          ; set Z bit
            ORR
                  p15, 0, r0, c1, c0, 0 ; write control reg
            MCR
                  call1
            BL
call1
            BL
                  call2
call2
                  call3
            ΒL
call3
; Restore Z bit (if necessary)
                                          ; clear Z bit
            BIC
                  r0, r0, #0x800
                  p15, 0, r0, c1, c0, 0 ; write control reg
            MCR
            END
```

## 4. ARM1176 initialization code

```
;* The confidential and proprietary information contained in this file may
;* only be used by a person authorised under and to the extent permitted
;* by a subsisting licensing agreement from ARM Limited.
;*
;*
           © COPYRIGHT 2007 ARM Limited.
;*
               ALL RIGHTS RESERVED
;*
;* This entire notice must be reproduced on all copies of this file
;* and copies of this file may only be made by a person if such person is
;* permitted to do so under the terms of a subsisting license agreement
;* from ARM Limited.
;*
;
; Code to initialize ARM1176
;
; --- Standard definitions of mode bits and interrupt (I & F) flags in PSRs
Mode_USR
             EQU
                    0x10
Mode_FIQ
             EQU
                    0x11
Mode_IRQ
             EQU
                    0x12
                    0x13
Mode_SVC
             EQU
Mode MON
             EQU
                    0x16
Mode ABT
             EOU
                    0x17
Mode UND
             EOU
                    0x1B
Mode_SYS
             EQU
                    0x1F
I Bit
             EOU
                    0x80 ; when I bit is set, IRQ is disabled
             EOU
                    0x40 ; when F bit is set, FIQ is disabled
F_Bit
                 | Header Code
                               , CODE
          AREA
ENTRY
               Start
reset
          В
; Exception vectors should be added here
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
Start
; On reset the ARM core is in SVC mode
          MOV
               r0, #0
          MOV
               r1, #0
          MOV
               r2, #0
          MOV
               r3, #0
          MOV
               r4, #0
          MOV
               r5, #0
               r6, #0
          MOV
          MOV
               r7, #0
          MOV
               r8, #0
          MOV
               r9, #0
          MOV
               r10, #0
          MOV
               r11, #0
               r12, #0
          MOV
          MOV
               r13, #0
```

```
MOV
                  r14, #0
; Enter each mode in turn and initialize the registers specific to it
            MSR
                  CPSR_c, #Mode_FIQ:OR:I_Bit:OR:F_Bit
            MOV
                  r8, #0
            MOV
                  r9, #0
            MOV
                  r10, #0
            MOV
                  r11, #0
            MOV
                  r12, #0
            MOV
                  r13, #0
            MOV
                  r14, #0
            MSR
                  CPSR_c, #Mode_IRQ:OR:I_Bit:OR:F_Bit
            MOV
                  r13, #0
            MOV
                  r14, #0
                  CPSR_c, #Mode_ABT:OR:I_Bit:OR:F_Bit
            MSR
                  r13, #0
            MOV
                  r14, #0
            MOV
            MSR
                  CPSR_c, #Mode_UND:OR:I_Bit:OR:F_Bit
                  r13, #0
            MOV
                  r14, #0
            MOV
; System mode shares user mode registers
            MSR
                  CPSR_c, #Mode_SYS:OR:I_Bit:OR:F_Bit
            MOV
                  r13, #0
                  r14, #0
            MOV
            MSR
                  CPSR_c, #Mode_MON:OR:I_Bit:OR:F_Bit
            MOV
                  r13, #0
            MOV
                  r14, #0
; Leave core in SVC mode (if necessary)
            MSR
                  CPSR_c, #Mode_SVC:OR:I_Bit:OR:F_Bit
; Initialise ALU paths
            ADD
                  r0, r1, r2
; Initialise return stack - requires branch prediction enabled (Z=1)
                  p15, 0, r0, c1, c0, 0
                                         ; read control req
            MRC
                  r0, r0, #0x800
            ORR
                                          ; set Z bit
            MCR
                  p15, 0, r0, c1, c0, 0
                                         ; write control reg
            ΒL
                  call1
call1
                  call2
            BT.
call2
                  call3
            BL
call3
; Restore Z bit (if necessary)
            BIC
                  r0, r0, #0x800
                                          ; clear Z bit
            MCR
                  p15, 0, r0, c1, c0, 0 ; write control reg
            END
```