AN2551
Application note

Configuring the STR91xFA MCU for optimum CPU performance

Introduction

The STR91xFA series of Flash MCUs is based on an ARM966E CPU core which executes code directly from its internal Flash memory at a rate of up to 96 MHz. To allow flexibility for a variety of applications and power management schemes, there are many configuration settings available to firmware during the initialization of the STR91xFA at start-up. This application note outlines the necessary steps to ensure the STR91xFA is configured for optimum performance for those applications which require the STR91xFA to operate at full speed and deliver the highest performance from the CPU core and highest bandwidth of data movement.

The related software is packed into a zip file associated with this application note and available from http://www.st.com/.

Contents

1 About STR91xFA

The STR91xFA family of MCUs combines a powerful 32-bit ARM966E RISC processor core, dual-bank Flash memory reaching 544 Kbytes and a vast 96 Kbyte SRAM for data or code storage. It includes a rich peripheral set to form an ideal embedded controller for a wide variety of applications.

This microcontroller has a custom memory accelerator, consisting of a Pre-fetch Unit and Branch Cache coupled with the Instruction Tightly Coupled Memory (I-TCM) of the CPU core, to accelerate the performance of the Flash memory system and lower Interrupt Latency. The job of the PFQ is to keep the ARM core continuously fed with instructions. It performs asynchronous pre-fetch cycles to the Flash memory during idle bus cycles to keep the Pre-fetch queue full. When instruction addresses are sequential, the burst Flash memory will supply the CPU core with 32-bit instructions at a continuous rate of 96 MHz.

However, when instructions have non-sequential addresses, such as during a branch in code execution, the PFQ must be flushed and refilled, which imposes a stall to the CPU core. The role of the BC is to minimize the occurrences of these stalls. The BC will remember the first eight 32-bit instructions of each of the previous 15 branch destinations taken by the firmware. Anytime one of these 15 branches is taken again, the BC will immediately supply the first eight instructions to the PFQ, which significantly reduces the amount of time the CPU is stalled. While the CPU consumes these first eight instructions, the PFQ has a chance to refill itself and be ready to supply instructions again at 96 MHz on the 9th instruction.

In order to take full advantage of the memory accelerator, the system configuration registers of the STR91xFA must be initialized correctly.

For more about STR91xFA, refer to http://www.st.com.

2 Configuring STR91xFA for best performance

The following items must be configured as indicated to achieve optimum CPU performance:

2.1 System clock configuration

1. The system clock can be boosted using the PLL to 96 MHz by updating the PLL configuration register SCU_PLLCONF register. This clock is generated for internal use by selecting the appropriate multiplier and divider.
2. Select the appropriate Clock dividers in the Clock control register (SCU_CLKCNTR):
   a) The Flash memory interface clock (FMICLK) should have the same frequency as the RCLK clock (96 MHz) this means no FMICLK dividers used. As a result since Flash has a Sequential Burst read up to 96 MHz, we reduce execution time from there.
   b) If code is executed from an external memory, it’s recommended that the External memory interface clock (BCLK) to be used in its highest frequency depending on read access time of the external memory.
   c) For correct operation, the APB Peripheral Clock (PCLK) frequency should be less than or equal to 48 MHz. So, to run APB bus at its maximum frequency, choose 2 as the PCLK divider.

2.2 Wait states insertion

Maximum memory performance is achieved by specifying the correct number of wait states in relation to the CPU and FMI operating frequency.

1. Wait states may be specified for non-sequential-address read accesses of Flash memory (no wait states are required for sequential-address read requests). For best performance, specify in the Flash configuration register, bits 11 and 12 (see STR9 Flash Programming Manual), one wait state for an FMI frequency of 66 MHz or less, and 2 wait states when FMI frequency is above 66 MHz.
2. Wait states may be specified for writing Flash memory. Best performance is achieved when 1-cycle writes (zero wait states) are specified for Flash write operations in the FMI Control Register, FMI_CR, bit 8 (see STR91xFA Reference Manual).
3. Wait states may be specified for accessing the SRAM from either the CPU's Data Tightly Coupled Interface (DTCM) or from the AHB bus. The DTCM and the AHB both share access to the SRAM through the SRAM arbiter circuitry. Best performance is achieved when zero wait states are specified for SRAM access in the System Configuration Register0, SCU_SCR0 (see STR91xF Reference Manual), bit 1 for DTCM access, and bit 2 for AHB access.

2.3 Buffered writes

The CPU can access SRAM and the peripherals on the AHB/APB using either buffered or non-buffered writes. Buffered writes operations are implemented by writing to SRAM and peripherals in one address range, or standard non-buffered writes are implemented by writing to SRAM and peripherals at a different address range (see STR91xFA datasheet for memory map). When the CPU makes a buffered write operation through the DTCM to the SRAM, there will be zero delay, even if SRAM arbitration is currently granted to the AHB at the time of the DTCM write operation. A similar benefit can be enjoyed when the CPU writes to the peripheral on the AHB/APB, decoupling the CPU from waits associated with a busy bus or peripheral. However, care must be taken using buffered writes to preserve data coherency. For example, if the AHB has just written a value to SRAM while a buffered write is pending from the DTCM for the very same SRAM address, then when arbitration is granted to back the DTCM, the most current value just written by the AHB will be overwritten by the DTCM. If this situation would cause a problem in your system, then non buffered writes are required.

2.4 Pre-Fetch Queue (PFQ), Branch Cache (BC) accelerator

The PFQ/BC memory accelerator can be enabled and disabled by firmware. By default, the PFQ/BC is enabled, but it's important to ensure that your initialization firmware has not disabled the PFQ/BC unintentionally (bit0 of register SCU_SCR0).

Early versions of STR91x firmware library from ST contained a mistake in the initialization code which disabled the PFQ/BC.

Additionally, in the first version of production silicon, STR91xF (RevD), the depth of the PFQ is set to four instructions by default, and firmware must write to bit 0 of the FMI control register FMI_CR to specify a PFQ depth of 8 instructions to achieve optimum CPU performance. However, in the current production silicon STR91xFA (RevG), the depth of the PFQ is fixed at eight instructions and it cannot be changed, ensuring optimum performance.

2.5 Instruction-set mode

32-bit architectures have a higher performance when manipulating 32-bit data. For applications needing best performance, STR91xFA should be used in ARM (32-bit) mode.

3 Implementation

The guidelines given above could be implemented in “C” code or even in the assembly startup file. The current application note provides a special assembly startup file "91x_init.s" for EWARM, RVDK, RIDE and RVMDK tools chain which covers all these guidelines.

This start-up file configures the PLL@96 Mhz to be the default clock source. It offers also the ability to select either the OSC clock or the RTC clock.

The zip file associated with this application note contains:

- The STR91x firmware library v1.2 from ST (source and header files).
- A standard template project program that compiles all library files and also all the user modifiable files needed to create a new project:
  - 91x_conf.h: The configuration header file with all peripherals defined by default.
  - 91x_it.c: The source file containing the interrupt handlers (the function bodies are empty in this template).
  - main.c: The main program body.
  - EWARM, RVDK, RIDE, RVMDK sub-directories for each toolchain and contains the modified start-up file "91x_init.s"

When booting from bank1:

- Use OSC as the default clock source.
- If you want to run the CPU @ 66 MHz or higher, the PLL configuration can be done in the application code (“C” code) instead of in the start-up assembly file.
- The Flash wait state selection instructions must be executed from SRAM as it is not possible to read while writing in the same bank.

4 Revision history

Table 1. Document revision history

Date	Revision	Changes
29-05-2007	1	Initial release.