Instruction Execution Timing (Cont.)

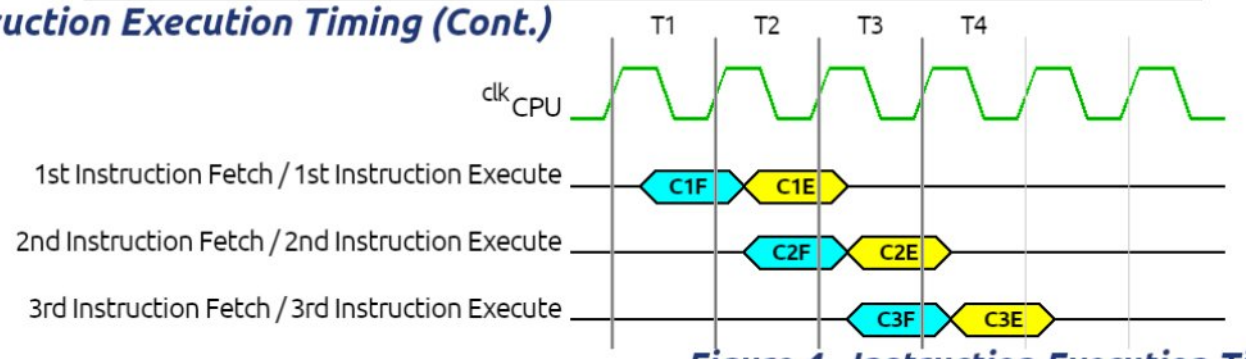

Figure 4. Instruction Execution Timing

As can be seen in the figure above, the second instruction will be fetched during the execution of the first instruction. When the second instruction enters execution, the third instruction is fetched. This eliminates the need for an additional CPU clock cycle to fetch the instruction before that instruction can be executed.

The following figure shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

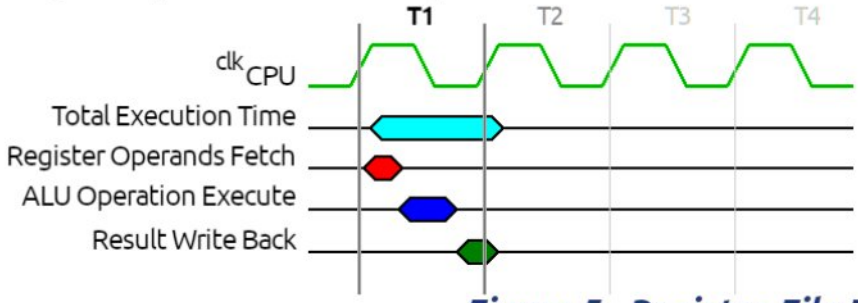

Figure 5. Register File Timing

Reset and Interrupt Handling

The following figure shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

The LGT8XM supports multiple interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. In general, all interrupts are assigned separate control bits. When this control bit is set and the Global Interrupt Enable bit of the Status Register is enabled, the CPU can respond to this interrupt.

The lowest addresses in the program memory space are reserved by default as the Reset and Interrupt Vector area. For a complete list of interrupts supported by the LGT8FX8P, please refer to the Interrupts section. This list also determines the priority of different interrupts. The lower the address, the higher the corresponding interrupt priority. RESET has the highest priority, then INT0 – External Interrupt Request 0. The start address of the interrupt vector table (except the RESET vector) can be moved to the beginning of any 256-byte alignment, via the MCU Control Register's IVSEL bit in (MCUCR) and the IVBASE vector base address register implementation.

When the CPU responds to an interrupt, the Global Interrupt Enable flag (I) is automatically cleared by hardware effectively disabling interrupts. The user can nest interrupts by enabling the I-bit in software. If this is done in software, any subsequent interrupts will interrupt the current interrupt service routine. The Global Interrupt Enable I-bit is automatically set after the execution of the interrupt return instruction (RETI), so that it can respond normally to subsequent interrupts.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For this kind of interrupt, the Program Counter (PC) value is directly replaced with the actual interrupt vector address. This enables the PC to execute the interrupt handling routine. Hardware automatically clears the interrupt flag bit. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If another interrupt condition occurs while interrupts are disabled the new interrupt flag will be remembered until the interrupt enable (I-bit) is set. Then this new interrupt will be serviced. This flag can also be cleared by software to bypass the new/waiting interrupt. Similarly, if the global interrupt enable bit (SERG.I) is cleared when an interrupt occurs, the corresponding interrupt flag bit will also be set to record the interrupt event, wait until the global interrupt enable bit is set, and will then be executed by order of priority.