MicroCore Labs

MCL51
Application Note

Lockstep

Quad Modular Redundant System

Introduction:

This application note describes a Lockstep Quad Modular Redundant (QMR) system that employs the microsequencer-based MicroCore Labs MCL51, an 8051 compatible soft-processor core.

The Lockstep QMR system consists of four MCL51 cores running in lockstep which can detect and recover from a variety of soft errors. Each module contains independent voting logic to detect errors and silently remove affected modules from the lockstep. If the module can successfully rebuild itself it can rejoin the lockstep where it will resume running synchronously with the other modules. The modules in this application can detect errors, rebuild themselves, and rejoin the lockstep in around 700 microseconds.

Peripherals such as UARTs and timers connect to the QMR system via a proxy addressing system and can independently select which healthy module's data to use.

To demonstrate the Lockstep QMR System's capabilities we have a YouTube video of it playing the "Flight of the Bumblebee" while being subjected to a variety of errors. The errors cause modules to fail and silently drop out of the lockstep after which they rebuild themselves and then rejoin the lockstep. Meanwhile, the remaining healthy cores continue to play the music without skipping a single note.

Lockstep Quad Modular Redundant System Description

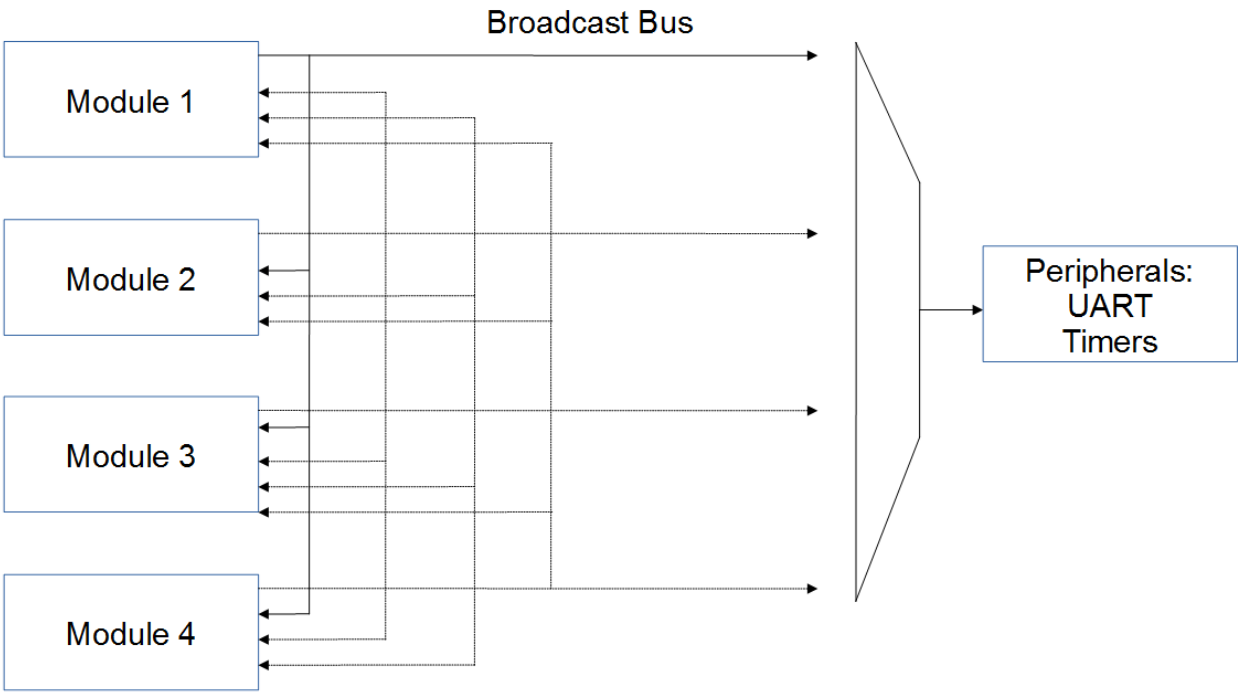
Each module continuously broadcasts its register, RAM, and ROM contents to the three neighboring modules over a "Broadcast Bus". The data on all Broadcast Busses will be identical and synchronous across all modules. The independent voting logic within each module compares its broadcast information with that of the neighboring modules and upon a discrepancy will silently take that module out of the lockstep and enter "Rebuild Mode".

When a module enters Rebuild Mode it uses the information received from the Broadcast Bus of neighboring modules to update its local registers, RAMs, and ROMs. Register and memory updates initiated by the 8051 program itself are interleaved into the broadcast stream and integrated by rebuilding modules.

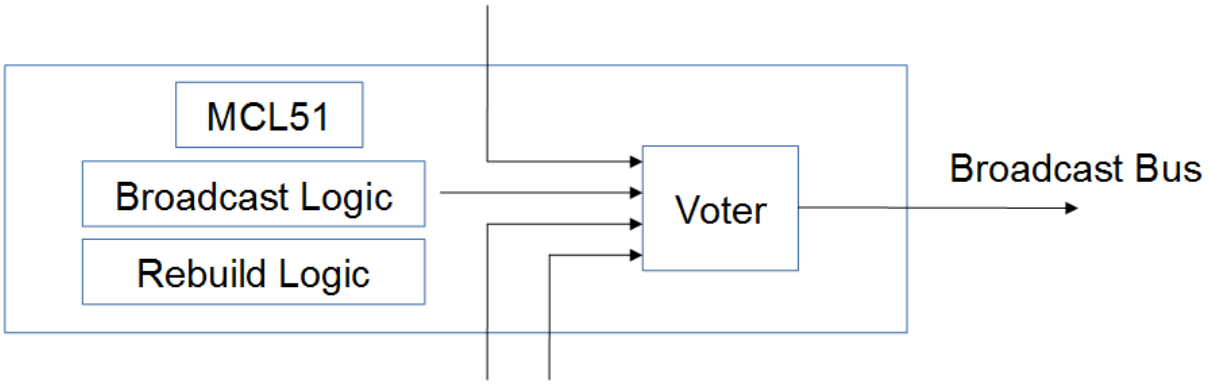
When this process is complete the module will wait for a Sync pulse from the Broadcast Bus to allow it to restart processing microcode at the same address as the healthy modules. At this point, the module can exit Rebuild Mode and rejoin the lockstep where it will resume broadcasting local information synchronously with the other modules.

The first diagram below illustrates the Broadcast Buses of the four modules which have independent connections to the other three modules in the system as well as to the peripherals which independently select which healthy module's information to use. The second diagram shows the components inside of a single module.

Lockstep Quad Modular Redundant System



Single Module



Broadcasting
All healthy modules continuously broadcast their memory and register contents as well as updates to these resources as they occur during the operation of the 8051 program. All microcode ROM, user data RAM, user program ROM and 8051 SFR registers are sent over the Broadcast Bus. Updates to the SFR registers and user data RAM will interleave with the normal broadcasting loop. When this occurs, the broadcast address is backed off a few locations to ensure that no addresses are skipped during the register/RAM broadcasts. All healthy modules synchronously generate identical data over their Broadcast Bus. When a module falls out of lockstep and enters Rebuild Mode it stops generating data on the Broadcast Bus.

Voting
Each module contains voting logic which is independent from the other modules. It compares the output of its Broadcast Bus with that of neighboring modules and decides if the local data is correct. If not, the voting logic will initiate Rebuild Mode which will disable the voting logic until the module rejoins the lockstep.

Rebuilding
If the voting logic detects a discrepancy between the module's data and neighboring modules, it will enter Rebuild Mode where it will write data received from the Broadcast Bus into its local registers, microcode ROM, user data RAM, and user program ROM. It also allows its local SFR registers and user RAM to be updated when interleaved into the broadcast stream. After the local registers and memories are refreshed, the module is ready to rejoin the lockstep. It waits to receive a Sync pulse from the healthy modules which indicates that they have just completed the execution of an 8051 instruction and are about to process the next one in the user program. At this point, the module can leave the rebuilding mode, restart broadcasting mode, and begin processing the next 8051 instruction at the same microcode address as the other modules. As the module rejoins the lockstep it will be completely synchronous with the other healthy modules at the microcode level. Modules that exhibit errors drop out of the lockstep silently and rebuild themselves without the intervention of any neighboring modules.

Error Handling

The Lockstep QMR System can detect and recover from Single Event Upsets (SEUs) and Single Event Transients (SETs) that occur in the microcode ROM, user program ROM, user data RAM, and registers. There are no state machines in the MCL51 processor core so SET and SEU errors that occur in these areas are not likely to have a permanent affect on the module. Hard errors will be detected and cause the module to silently drop out of lockstep without the ability for it to rebuild and rejoin.

Errors that occur in the FPGA's configuration logic are not recoverable with this application; however they will cause the module to silently drop out of the lockstep. To address this, the Xilinx Soft Error Mitigation (SEM) Core can be employed to detect and correct errors that occur in the FPGA's configuration memory.

The Lockstep QMR System can detect and attempt to rebuild two modules simultaneously. Each one can passively rebuild themselves using information received from the Broadcast Bus. No intervention is required from the healthy modules.

The Lockstep QMR System has a very small footprint which presents a proportionally small target for failures due to charged particles. Adding parity, scrubbing, and error correcting logic would reduce the likelihood of module failure, however it would also greatly increase the footprint and the susceptibility to soft errors. In this application, we made the decision to simply add more modules rather than increasing each one’s footprint with this additional logic. We contend that it is a better use of logic to have a greater number of modules that can passively rebuild themselves than to spend it implementing a smaller number of potentially more robust modules.

In theory, errors that occur in the FPGA due to logic timing errors as a result of an from improperly constrained design will also be masked by the Lockstep QMR System. Errors that occur at temperature and voltage corners because of inadequate constraint margins may cause modules to drop out of lockstep; however if the error is recoverable it will be able to rebuild itself and rejoin the lockstep. Clearly, a design should not be approached with this deliberately relied upon, however it is something to consider. This QMR application may be sufficiently small that it could be casually used as an embedded processor since it can recover from errors due to charged particles and potentially by those caused by internal FPGA timing errors.

Quad Module Redundancy (QMR) versus Triple Modular Redundancy (TMR)

In a Triple Modular Redundant (TMR) system the failure of one module immediately puts the system in a fragile condition. If a subsequent error occured in one of the two remaining modules there would be no way to decide which module's output is correct, which leaves the TMR system as essentially inoperable.

A Quad Modular Redundant (QMR) system's additional module greatly increases the robustness of the system by not being placed into a fragile condition upon a single module failure. Even if a second module were to fail there would still be two healthy modules providing broadcast information to rebuild the failed modules.

MCL51 Microsequencer-Based 8051 Processor Description

The MCL51 is an 8051-compatible embedded processor core that utilizes a high performance, seven instruction, 32-bit microsequencer with an ultra-small footprint of approximately 300 Xilinx Artix-7 LUTs. The core is divided into the Execution Unit (EU) and the Bus Interface Unit (BIU). The EU contains the microsequencer-based 8051 ALU and the BIU contains the 8051 registers, peripherals, and bus interface. All 8051 instructions are executed in microcode and there are no state machines in the processor core.

Lockstep QMR Resource Utilization

The total space for the Lockstep QMR application consisting of four MCL51 cores, voting logic, and peripherals is roughly 2,500 Xilinx Artix XC7A35 LUTs, which is approximately 12% of the device.

Implementation Considerations

This Lockstep QMR application provides a robust solution capable of recovering from SEU and SET errors that occur in any of the registers or memories in the system. It can not recover from hard or permanent errors or from those that occur in the FPGA's configuration logic. The Xilinx Soft Error Mitigation (SEM) Core can be instantiated into the design which can detect and correct errors in the FPGA's configuration memories.

Each module requires a clock that is synchronous across all modules. It would be preferred if each clock entered the FPGA via separate pins and used independent clock networks. Each of the four modules can also be partitioned into separate physical areas on the FPGA so a cluster of errors will be less likely to affect multiple modules.

Music Demonstration Description

The “Flight of the Bumblebee” demonstration uses an interrupt generated by a dual timer. The music data is held in the program ROM which contains frequency and duration information for each note. The main loop of the program is simply a loop of NOPs. A musical note frequency is programmed into timer0 and the duration programmed into timer1. When timer1 expires, an interrupt is generated which runs code to reload the timer with the next note's frequency and duration.

This demonstration allows the user to inject errors of various types into each module. The error types are selected by the slider switches and the error is injected when the user presses the push-buttons. There is one push-button for each module.

The demonstrations injects errors into the 8051 SFR registers, microsequencer address, module broadcast data, microcode data output, program RAM & ROM data output, and rebuilding address. One test even allows the user to completely zero out the MCL51 microcode ROM. Errors injected into these areas force the module to silently drop out of the lockstep and trigger them to rebuild themselves which they cannot do while the user is pressing the push-button. When the user releases the push-button, the module is then able to successfully rebuild itself and rejoin the lockstep.