A fault tolerant for processor

The mips – fault tolerant is mips 32 bits processor with error detection (Fault Tolerant). The processor implementation was designed by Lazaridis Dimitris.

Main aspects

The core is in 5 stages:

- Instruction extraction

- Instruction decoding

- Execution

- Memory access

- Update registers

It supports almost all instructions of mips technology, R type, I type, Branch, Jump and multiply packet instructions.

The multiply result is stored until is needed regardless if others instructions follows.

There is an error detection circuits for fault tolerant. It is implementing in hardware 100% which provides error detection at reset start-up.

There is a separate memory for instructions and another for data read – write which can be changed.

At each stage one clock cycle is used. Both memories function in descending pulse and the remaining pulse is used for developing the necessary functions (e.g. pipeline), which makes the core faster and more flexible.

All I types instructions are part decoded in first stage and all R types also part decoded in Alu control reducing the complexity in main Control unit (FSM).

All instructions are tested for correct execution. A test benchs from separate circuit implementation is also included (to verify the program which exists in Instruction memory).

The mips – fault tolerant was integrated in an FPGA from Xilinx version 13.1 in Spartan 3 xc3s400-5tq144 target device but can be fit in another similar target device.

The processor is implemented all in VHDL.

Error detection

With continuous scaling in CMOS technology the number of transistors grows more and more in a single chip. Chip multiprocessors (CMPs) are an efficient way for using this very large number of transistors integrated in a chip. Several researches show that high density integration makes modern processors prone to the risk of transient or permanent fault. However, the increase of temperature and decrease of the voltage in the chip lead to a higher susceptibility to faults. As the feature size shrinks the probability of a single transistor to become faulty, it increases due to the low threshold voltages.

It is projected that the rate at which the transient errors occur will grow exponentially and will soon represent one of the most significant issues in the design of future generation high-performance microprocessors.

This work proposes a fault tolerant architecture that tolerates the high fault rates that are expected in future technologies. It is test the multiply block counting a 0f as an input data and compare the result with a signature. It has a high fault coverage and fast execution due to hardware implementation, which will be more popular method for errors detection in future for the time saving (there is not time penalty), reliability, low cost and high presentence to fault coverage, low power consumption.

Further research

Most error detect methods for fault tolerance check the mips or a circuit at start up or at once or periodically to find any errors for fault coverage, but what if an error occurs during the tests? A fault data will process as correct. To work around with this, a non stop searching method test the mips continuously, it can be implement and find any error as it appears at the beginning and further more if the fpga has enough room to relocate the damaged place in another undamaged.

To implement this error detect method, we can inject in fsm and detect the errors for fault tolerance. Knowing the next stage (instruction) through fsm, and decide to test the “multiply” block circuits, could test the multiply circuits until the next instruction it is not concern this circuits, if a multiply instruction is coming up we can stop the process and continue when it is free again, thus we can find if an error occurs in this part of cpu and cover the fault tolerance. The same process it is possible to test all critical parts of mips or central unit and find if an error exits. The advantage in this method is that the error detects circuit works continuously. This method does not require double cores, but only some additional parts (low cost) and which can work in conjunction with fsm without consume the microprocessor’s working time but it works simultaneously.