UNIVERSITA' degli STUDI di ROMA
TOR VERGATA

Dependability of Digital Systems

Error detecting using arithmetic codes inside the ZERORISCY core ALU's

Marco Spaziani Brunella

0255562

marco.spaziani.brunella@uniroma2.it

1 Introduction

The aim of this work is to implement aritmetic code to protect the adder of the Arithmetic and Logic Unit of the Pulpino's ZERORISCY core. In particular, a mod-3 arithmetic code has been implemented and the resource occupancy has been compared to a design in which te ALU has a fully-replicated adder.
The mod-3 arithmetic code implementation is depicted in **Fig. 1**.

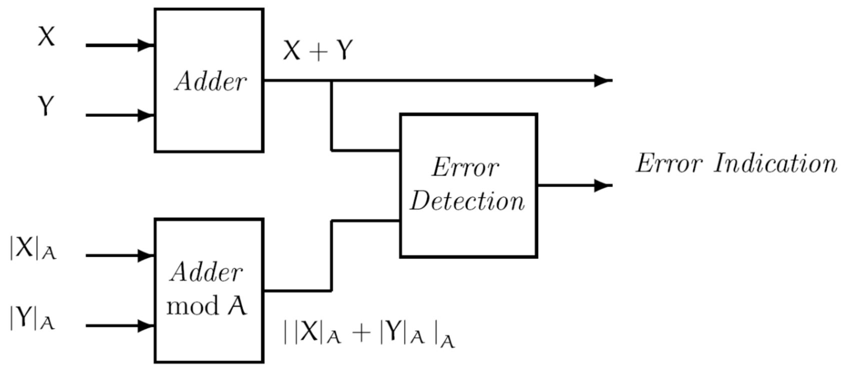


Figure 1: mod-3 adder scheme

2 Implementation

The implemented design is depicted in **Fig. 2**. The last 3 bits of the operands of the adder are taken. Then we perform the mod-3 operation and add them together. The result of this operation is compared to the lowest 3 bits of the full-adder result, after taking the mod-3 of the result. If the two are equal, the *arithmetic_check_ok* signal is raised and the result of the full-adder is actually passed to the output port, otherwise the output port is brought to all-zeros.

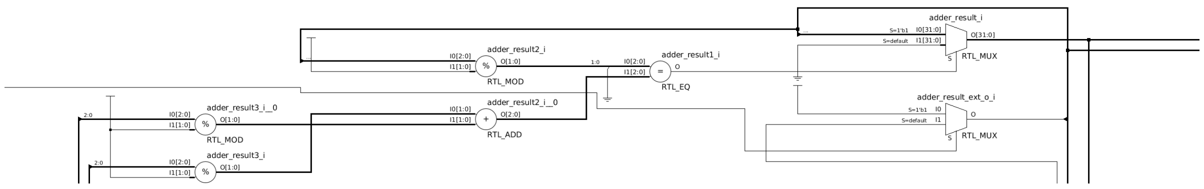


Figure 2: Implemented design under the spot

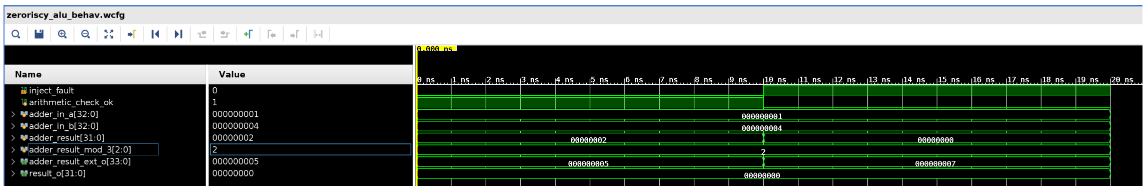
A **fault injection** logic has been implemented: when the *inject_fault* signal is raised, it flips the second bit on the full-adder, thus making the arithmetic check fail.

The modified code is available below:

```
  always_comb  begin

  adder_result_ext_o = $unsigned(adder_in_a) + $unsigned(adder_in_b);
  adder_result_mod_3 = ($unsigned(adder_in_a[2:0]) % 3) + ($unsigned(adder_in_b[2:0]) % 3);


    if (inject_fault == 1) adder_result_ext_o[1] = 1;

    if (((adder_result_ext_o[2:0]) % 3) == adder_result_mod_3) arithmetic_check_ok = 1;
    else arithmetic_check_ok = 0;

    mod_3_error = ~(arithmetic_check_ok);

    if (arithmetic_check_ok == 1) adder_result       = adder_result_ext_o[32:1];

    else adder_result = 31'b0;

␣␣␣␣adder_result_o␣␣␣␣␣=␣adder_result;

␣␣end
\label{listing}
```

3 Simulation

The simulated design is depicted in **Fig. 3**. In the first half of the simulation, the *inject_fault* signal is low, thus indicating normal operation of the adder. In fact, the *arithmetic_check_ok* signal is high, indicating no detected errors on the sum. When the *inject_fault* signal is raised, the second bit on the result of the main adder gets flipped. At this point, the *arithmetic_check_ok* signal falls since the mod-3 adder result does not match the sum of the mod-3 operands.



```
zeroriscy_alu_behav.wcfg
Name                      Value
inject_fault              0
arithmetic_check_ok       1
adder_in_a[32:0]          000000001
adder_in_b[32:0]          000000004
adder_result[31:0]        00000002
adder_result_mod_3[2:0]   2
adder_result_ext_o[33:0]  000000005
result_o[31:0]            00000000
```

Figure 3: Simulation of the mod-3 arithmetic code

4 Synthesis results

The resource utilization on a ZedBoard for the unmodified ALU is depicted in table 1, while the resource occupation for the ALU modified with the mod-3 arithmetic code is depicted in 2. For comparison, a fully-replicated adder synthesis has been carried on 3. In the modified ALUs (mod-3 and fully-replicated), the synthesis takes into account also the **fault injection** logic and the **verification** logic.

Resource Type	**Used**	**Available**	**% Used**
Slice LUTs	259	53200	0.49

Table 1: Resource utilization for the unmodified ZERORISCY ALU

Resource Type	**Used**	**Available**	**% Used**
Slice LUTs	284	53200	0.53

Table 2: Resource utilization for the mod 3 ZERORISCY ALU

Resource Type	**Used**	**Available**	**% Used**
Slice LUTs	320	53200	0.60

Table 3: Resource utilization for the fully-replicated adder ZERORISCY ALU

Using a mod-3 arithmetic code on the adder, including the fault injection logic and the verification logic, adds an overhead of 9.65%, while fully-replicating the adder unit will add an overhead of 23.55%.

5 Conclusions

In this work I've implemented a mod-3 arithmetic code to detect errors on the adder of the ALU of the Pulpino's ZERORISCY core. Synthesis results have been provided and comparison between vanilla ALU and fully-replicated adder ALU have been discussed. The results confirm what depicted by theory: a full replication of the adder, while giving full error coverage, introduces a huge overhead on the occupied area.
The mod-3 modified core has been later flashed in the ZedBoard, mapping the *inject_fault* signal to one of the available toggle switch and the *arithmetic_check_ok* signal to one of the LED.