www.bezem.de

Processor Self-Tests

Testing Correct Operation of a Processor for Safety Implementations

Johan Bezem

Target Audience

www.bezem.de

- Embedded Developers with:
  - low-level ambitions
  - a basic understanding of assembly programming
  - a basic understanding of logic design

Program

- Introduction
- ISO 26262 and ASIL
- Why do we need a processor self-test?
- Limitations checking hardware by software
- Checking the registers
- A special register: Processor flags
- Checking instructions
- Example: Half-adder

Introduction

www.bezem.de

- Johan Bezem, freelance developer (1990-)
- C programming experience since 1984
- Relevant efforts:
  - Two Basic-compilers for ZX Spectrum, written in Z80-assembler, 1983-1984
  - A pre-emptive real-time multitasking kernel, written in C/assembler, 1991-1994
- Assembler experience since 1978
- Written PSTs for 3 different processors

ISO 26262 and ASIL

www.bezem.de

- The ISO standard defines safety levels
- Basic processor checks are done in the factory
- During life-time, a single-processor implementation has to check itself to detect deteriorations in operation
- This shall be done before anything safety-related is executed

Limitations

www.bezem.de

- Using software to check the hardware is like Munchhausen

- Software is far from perfect to test its own processor, but it’s the only method we have at our disposal
- We shall limit untested dependencies!

Limitations (2)

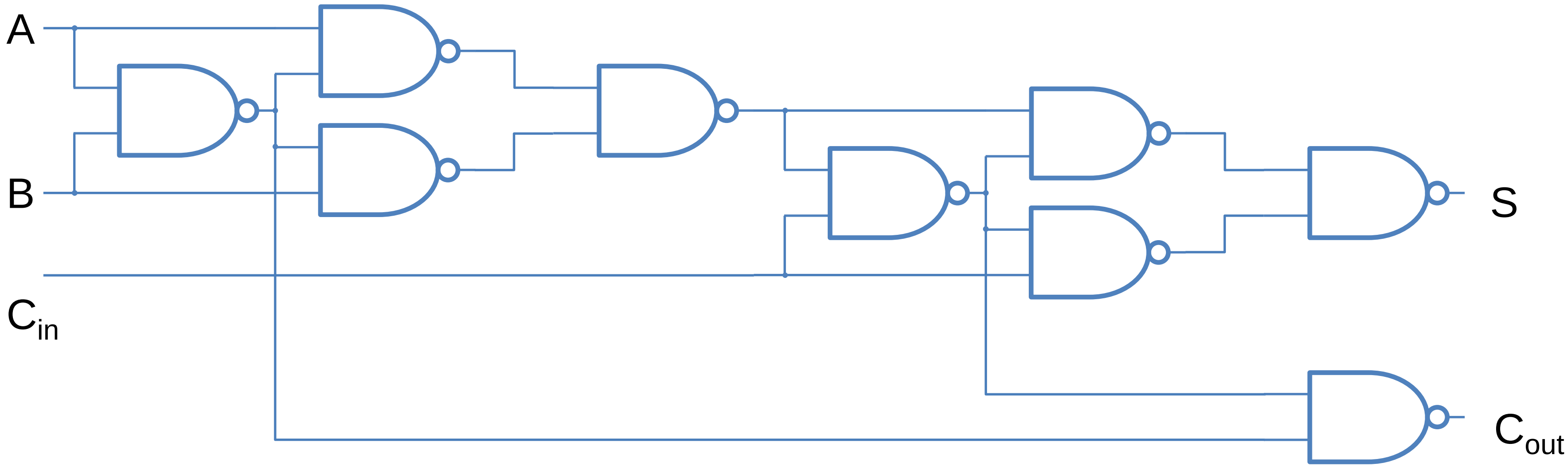
- We shall not use RAM – at all
  Not even the stack!
- We shall start at the very beginning of operations, immediately after each RESET
- The principle shall be “Test before use”
- Where that is not possible, limit its use to the absolute minimum

Faults

Possible hardware faults:

- Shortcut to GND
- Shortcut to V_{cc}
- Open connection
- Shortcut to neighboring signal line

Full-Adder



$$S = (A \veebar B) \veebar C_{in}$$

$$C_{out} = A \wedge B \vee C_{in} \wedge (A \veebar B)$$

What type of ALU?

- Look at execution times for 8/16/32/64-bit addition/subtraction to deduce the ALU width
- Look at flags like AC (auxiliary carry), N (negative result) to guess at the possible ALU operations
- Look at execution times for multiplication and division to see if it's part of a dedicated circuit, or micro-coded

Testing registers

- Test with 0x5555 and 0xAAAA
- Use different opcode families for load and test, if possible
- Covers all four faults

Testing flags

www.bezem.de

- Some flags change upon normal operations
  Examples: ADD, SUBC, MULU
- Others are static;
  Examples: bank selection, interrupt enable
- Very processor dependent

Flags - examples

www.bezem.de

Setting Z, C and AC to zero:

```
MOV     A, #0x01
CMP0    A
```

Setting Z, C and AC to one:

```
MOV     A, #0x01
ADD     A, #0xFF
```

[Make sure to check the processor manuals]

Testing instructions

www.bezem.de

- We do not know how the instructions are implemented in silicon
- We have some hints:
  - The flags modified by the instruction
  - The number of clock cycles needed per execution – and fixed or variable
  - The (deduced) width of the ALU
- And, more generally, the hardware development manual

Testing instructions (2)

- Increment is the same as adding #0x01
- ADD is the same as ADC with C = 0
- If we have a NEG instruction (creating a 2s-complement), we can assume that SUB is using the same logic gates as ADD
- CMP is the same as SUB, without storing the result

Testing instructions (3)

www.bezem.de

- Test multiplication with primes – avoid silicon optimizers
- Test division with the same primes, but add a constant to avoid a zero remainder
- Ignore composite instructions

Instruction examples

www.bezem.de


```
MOVW  AX, #0x8013      ; -32749
MOVW  BC, #0x8031      ; -32719
MULH                   ;   = +1 071 514 531
CMPW  AX, #0x03A3      ;   = 0x3FDE_03A3
SKZ
BR    L_ERROR
MOVW  AX, BC
CMPW  AX, #0x3FDE
SKZ
BR    L_ERROR
```

Testing addressing

- We need one reliable (=tested!) location in memory available for our tests
- We use one addressing mode only to test the location, similar to testing a register
- Then all possible addressing modes are used to check the results as well

Addressing example


```
; Set the initial value
MOVW        S:L_LOC, #0XAAAA
…
; Check
MOVW        HL, #LWRD(L_LOC)
MOVW        AX, [HL]
CMPW        AX, #0XAAAA
SKZ
BR          L_ERROR
```

If we have time left

www.bezem.de

- Using sentinel codes to register results
- Using hamming distance 75%
- Using 0xF5 for TRUE, 0x9A for FALSE
- Where to store the result
- What to do in case of an error

Thank You!

Author:

Johan Bezem

t: +49 172 5463210

m: administration@bezem.de