Saturation Arithmetic

Ulrich Hoffmann <uho@xlerb.de>

EuroForth 2014 Palma de Mallorca

Overview

- What is saturation arithmetic?
- How to implement it in Forth?
- Demo
- Discussion

Problems with Circular Arithmetic

- Overflows and Underflows
  - undetected
  - detected and now what   (closed loop control)

```
16bit:   30000 30000 + .     → -5536
```

```
16bit:   -10000 30000 - .    → 25536
```

Saturation Arithmetic

- Idea:
    - Let there be a maximum/minumum values
        - if the calculation overflows use the max
        - if the calcualtion unterflows use the min

```
16bit:  30000 30000 +s .   →  32767
```

```
16bit:  -10000 30000 -s .   → -32768
```

Arithmetic properties

monotonicity

for all $x \in \mathbb{Z},\ a \in \mathbb{Z}, a \geq 0:$

$$x + a \geq x$$

$$x - a \leq x$$

- **Does not hold** for circular arithmetic
- **Holds** for saturation arithmetic $(\mathbb{A})$

Arithmetic properties

associativity

for all $a, b, c \in \mathbb{Z}:$

$$(a + b) + c = a + (b + c)$$

$$(a - b) + c = a - (b - c)$$

- **Holds** for circular arithmetic
- **Does not hold** for saturation arithmetic $(\mathbb{A})$

Strategies

- A priori
  - Detect over/underflow before calculating
  - return min/max if detected else calculate
- A posteriori
  - calculate
  - return min/max if calculation had over/underflow

Saturation Arithmetic for Forth

- A set of saturation operators

```
+s -s *s negate_s abs_s …
```

What about unsigned numbers?

What about unsigned numbers?

- Another set of unsigned saturating operators?

```
16bit:  30000 30000 +us u.   →  60000
```

```
16bit:  40000 40000 +us u.  → 65535
```

```
16bit:  10000 30000 -us u.       → 0
```

Too many operators!

- Just two new words:

sat (x -- x | max)   signed saturation

usat (x -- x | umax)   unsigned saturation

- Let `+  –  *` set (internally) enough information so that `sat` and `usat` can work.

```
16bit:   30000 30000 + sat .     →   32767
16bit:  -10000 30000 - sat .     →  -32768

16bit:   30000 30000 + usat u. →   60000
16bit:   40000 40000 + usat u. →   65535
16bit:   10000 30000 - usat u.  →        0
```

Has saturation happened?

- `usat` and `sat` set a flag **`usatq`** when saturation took place.
- Applications can check it to see if the results are exact.
- Applications must explicitly reset `usatq`.

Demo

Implementation

- 4e-Forth

```
;C +        n1/u1 n2/u2 -- n3/u3     add n1+n2
        HEADER  PLUS,1,'+',DOCODE
        ADD     @PSP+,TOS
        MOV     SR, &SRSAVE
        BIS     #1000h, &SRSAVE
        NEXT
```

- Implementation of – similar.

Implementation

- 4e-Forth

```
; SAT       x -- x
   HEADER  SAT,3,'SAT',DOCODE
         BIT     #100h,&SRSAVE     ; was overflow bit set?
         JZ      nosat
         BIT     #1h,&SRSAVE       ; check carry for over or underflow
         JZ      satovl
         MOV     #8000h, TOS
         jmp satsetq
satovl:  MOV     #7FFFh, TOS
satsetq: MOV     #-1, SATQ
nosat:   NEXT
```

- Implementation of `usat` similar.

Discussion

- Fewer error handling code as you can just continue to run.
- What to do with division by zero?
- Adding more tasks to + and – slows them down, even if you don't need saturation but
- Overall system-impact low
- As a kernel option or code generator configuration when saturation arithmetic is required

¿Questions?