Faithful Single-Precision Floating-Point Tangent for FPGAs

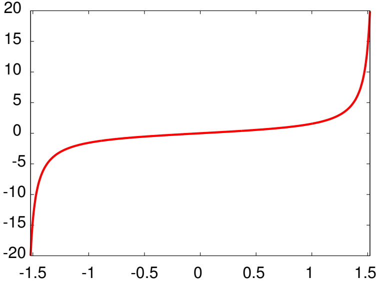
Martin Langhammer and Bogdan Pasca

ALTERA

Feb 12, 2013

What?

Tangent function:



- **periodic**, input range restricted to $(-\pi/2, +\pi/2)$
- **symmetrical** to the origin: $\tan(-x) = -\tan(x)$
- Taylor series:

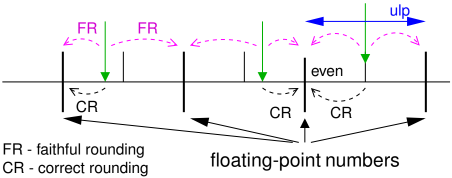
$$\tan(x) = x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \dots \; x \in \left(\frac{-\pi}{2}, \frac{\pi}{2}\right)$$

What?

- Compute in **floating-point** (IEEE-754)
- Triplet **(sign, exponent, fraction)** defines *x*:

$$x = (-1)^s 2^e 1.f$$

- Focus on **single-precision** $\mathrm{w_E} = 8$ (exp. width), $\mathrm{w_F} = 32$ (frac. width)
- Perform **faithful rounding**:



How?

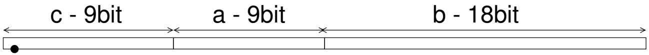
- **Restrict input to fixed-point**
  - $\tan(x) \approx x$ for $x < 2^{-w_F/2}$
  - dynamic input range: $[2^{-w_F/2}, +\pi/2]$
  - input in error-free fixed-point on $1 + w_F + \lceil w_F/2 \rceil$ bits (24+12=36 bits for single precision).
- **Use mathematical identities**:

$$\tan(a+b) = \frac{\tan(a)+\tan(b)}{1-\tan(a)\tan(b)},$$

$$\tan(a+b+c) = \frac{\dfrac{\tan(a)+\tan(b)}{1-\tan(a)\tan(b)} + \tan(c)}{1 - \dfrac{\tan(a)+\tan(b)}{1-\tan(a)\tan(b)}\tan(c)}$$

How? - Single-precision specific simplifications

- use the **fixed-point decomposition** of the input argument



- **simplify**:
  - $\tan(a)$ and $\tan(b)$ small $\rightarrow \tan(a)\tan(b)$ very small
  - $b < 2^{-17}$ safe to use $\tan(b) \approx b$

$\rightarrow$ tangent computed using:

$$\tan(x) = \frac{\tan(c) + \tan(a) + b}{1 - (\tan(a) + b)\tan(c)}$$

How? - Faithful precision requirement

$$E_{\text{total}} = E_{\text{approx}} + E_{\text{round}}$$

- E_{round} **pack result to floating-point** (nearest, 1/2*ulp*)
- E_{approx} **method errors + datapath trimmings**
- tangent implemnted as FP multiplication

$$p = n \times id$$

- **target: keep** $E_{\textbf{approx}} < 1/2ulp$

... some steps later:
$\rightarrow$ for single-precision $p = 24$ (error bound slightly better than $1/4ulp$ for numerator and inverse denominator)

How? - Datapath width

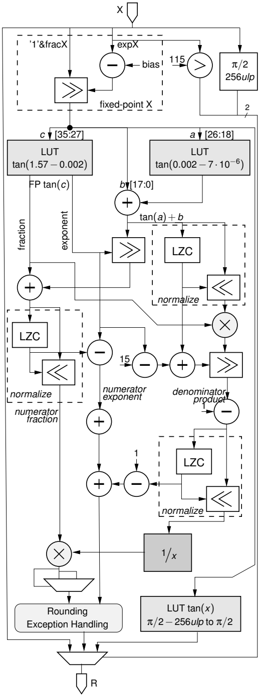
- **certify** approximations for the **numerator**:
  1. $\tan(c) = 0$ and $\tan(a)\tan(b)$ maximal:

```
a =  .        111111111
b =  .                      11111111111111000
```

     - relative error is slightly less than 2^{-25}, and should be 2^{-26}.
     - but denominator is 1 and carries no error $\rightarrow$ accuracy reached
  2. $\tan(c)$ minimal but > 0 and $\tan(a)\tan(b)$ maximal
     - $\tan(a) < \tan(c)$ relative error is 2^{-26} (tabulated precision for $\tan(c)$)
     - compute both $\tan(a)$ and $\tan(b)$ with $1 + w_F + 2$ bits of accuracy.
- **certify** approximations for **denominator**:
  - possible **cancellation amplifies** existing **errors**
  - **avoid large cancellation using additional table**
  - tabulate results for 256*ulp* before $\pi/2$
  - largest cancellation can now be produced by:

```
c = 1.10010010;
a =  .        000111001;
b =  .                 010000;
```

  - cancellation size is 3 bits $\rightarrow$ 3 additional bits for right term
  - compute $\tan(a)$ and $\tan(c)$ on $1 + w_F + 2 + 3$ bits with 0.5*ulp* of accuracy.



X
'1'&fracX
expX
bias
115
π/2
256ulp
fixed-point X
2
c [35:27]
a [26:18]
LUT
tan(1.57 − 0.002)
LUT
tan(0.002 − 7 · 10^−6)
FP tan(c)
b [17:0]
tan(a) + b
fraction
exponent
LZC
normalize
LZC
normalize
15
numerator exponent
denominator product
1
numerator fraction
1
LZC
normalize
1/x
Rounding
Exception Handling
LUT tan(x)
π/2 − 256ulp to π/2
R

Results

Architecture	Lat @ Freq.	Resources
ours	30 @ 314MHz	18MUL, 8M9K, 1172LUT, 1078Reg
$\tan(\pi x)$ [1]	48 @ 360MHz	28MUL, 7M9K, 2633LUT, 4099Reg
$\sin\cos(\pi x)$ [2]	85ns	10 MUL, 2*1365 LUTs
div [3]	16 @ 233MHz	1210LUT, 1308REG
div [4]	11 @ 400MHz	8MUL, 4M9K, 274LUT, 291Reg

- shorter latency
- fewer resources

[1] Altera DSP Builder Advanced Blockset.
`http://www.altera.com/technology/dsp/advanced-blockset/dsp-advanced-blockset.html`
[2] Jérémie Detrey and Florent de Dinechin. *Floating-point trigonometric functions for FPGAs.* FPL'07
[3] Florent de Dinechin and Bogdan Pasca. *Designing custom arithmetic data paths with FloPoCo.*. IEEE DT 2011
[4] Bogdan Pasca. *Correctly rounded floating-point division for DSP-enabled FPGAs.* FPL'12

Conclusion

- we implement the tangent function as a **fused operator**
- exploit FPGA flexibility: **exotic formats**, fixed-point and floating-point
- careful error analysis $\rightarrow$ **compute just right**
- make **efficient use of** existing **FPGA resources** (memories and multipliers)

Thank you and see you at the poster!