Faithful Single-Precision Floating-Point Tangent for FPGAs

Martin Langhammer, Bogdan Pasca
ALTERA European Technology Center

What?

The tangent function

CORDIC implementations[7, 4]:

+ **iterative low-resource** for FPUs in embedded processors
- **unrolled stressful routing** due to multiple, deep arithmetic structures

Polynomial approximations (sin, cos + ÷ using inverse [6])

+ **map well** to DSPs and embedded memory blocks
- **wasteful** when implemented using **operator assembly** [2, 5]

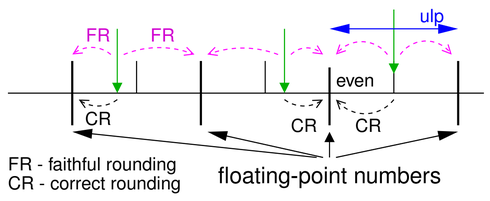
We implement the **floating-point tangent as a fused operator**.

Background

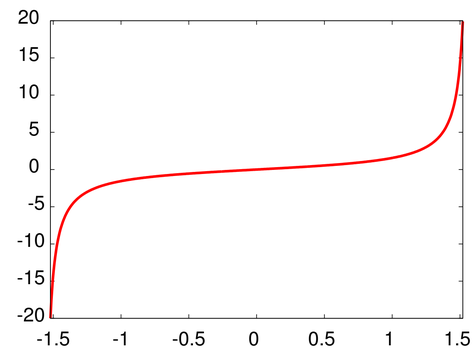
IEEE-754 floating-point value (sign, exponent, fraction):

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

w_E = 8 (exp. width), w_F = 32 (frac. width)- single precision



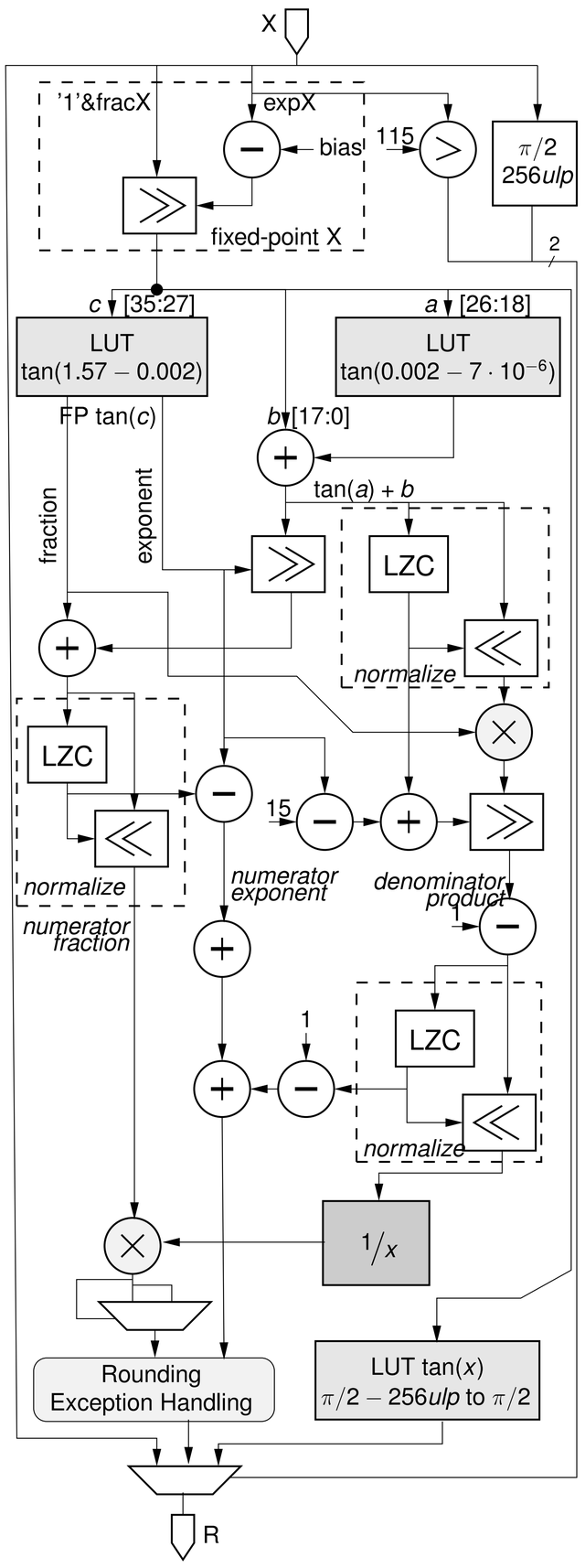
Here we target a **faithful floating-point tangent** function (1 ulp)



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

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

How?



Synthesis results for Stratix-IV C2. MUL = 18-bit multipliers

Architecture	Lat @ Freq.	Resources
ours	30 @ 314MHz	18MUL, 8M9K, 1172LUT, 1078Reg
tan(πx) [1]	48 @ 360MHz	28MUL, 7M9K, 2633LUT, 4099Reg
sin cos(πx) [3]	85ns	10 MUL, 2*1365 LUTs
div [2]	16 @ 233MHz	1210LUT, 1308REG
div [6]	11 @ 400MHz	8MUL, 4M9K, 274LUT, 291Reg

Algorithm

1. *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).
2. *Use mathematical identities*

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

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

3. *Error analysis*
   - for faithful rounding $E_{total} < 1ulp$

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

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

$$p = n \times id$$

   - the approximation error (tildes are approx.):

$$E_{approx} = |(\tilde{p} - p)/p|$$

   - the computed product

$$\begin{aligned}\tilde{p} &= \tilde{n} \times \tilde{id} \\ &= n(1+\epsilon) \times id(1+\epsilon) \\ &= n \cdot id + 2 \cdot n \cdot id \cdot \epsilon + n \cdot id \cdot \epsilon^2.\end{aligned}$$

   - approximation error is:

$$\begin{aligned}E_{approx} &= |(p - \tilde{p})/p| \\ &= |(2 \cdot n \cdot id \cdot \epsilon + n \cdot id \cdot \epsilon^2)/(n \cdot id)| \\ &= |2\epsilon + \epsilon^2| \leq 1/2 \cdot 2^{-p}\end{aligned}$$

   - for single-precision $p = 24 \rightarrow \epsilon$ slightly smaller than 2^{-26} (error bound slightly better than $1/4ulp$ for numerator and inverse denominator)
4. *Precision-specific optimizations (single-precision)*
   - use the **fixed-point decomposition** of the input argument

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

→ tangent computed using:

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

   - **certify** approximations for **numerator**:
     - compute a bound on the error of this approximation:
       (a) tan(*c*) = 0 and tan(*a*) tan(*b*) maximal:

```
a =  .          111111111
b =  .                   1111111111111111000
```

       ∗ relative error is slightly less than 2^{-25}, and should be 2^{-26}.
       ∗ but denominator is 1 and carries no error → accuracy reached

       (b) 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 $256ulp$ before $\pi/2$
     - largest cancellation can now be produced by:

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

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

Implementation

1. **tabulate** tan(*a*) and tan(*c*):
   - tan(*c*) dynamic range is $2^{-8} - 2^{11}$.
     - **store in floating-point** format exponent 5 bits and fraction on $w_F + 5$ bits (explicit "1" stored)
     - total width = 34 bits (M9K has 36-bit, M20K 40-bit)
   - tan(*a*) dynamic range is just 9 positions
     - **store in fixed-point** on 9 + 23 + 5 = 36 + 1 bits
2. **computing the numerator**: tan(*c*) + tan(*a*) + *b*
   - tan(*a*) and *b* are in fixed-point format → added directly
   - tan(*a*) + *b* is aligned to the exponent of tan(*c*) (max 19-bit shift), beyond that return tan(*c*)
   - potentially normalize (1-bit)
3. **computing the denominator**: $1 - (\tan(a) + b)\tan(c)$
   - tan(*a*)+*b* normalized (in floating-point) is multiplied by tan(*c*)
   - denormalize the product to to fixed-point
   - perform a fixed-point subtraction
   - normalize (maximum cancellation is 3-bit)
4. **compute the denominator inverse** + (1-bit max normalize)
5. **perform final multiplication** + (1-bit max normalize)
6. **round** to nearest
7. **multiplex with other branches**:
   - if $e < -12$ return x
   - if $x > \pi/2 - 256ulp$ read output from table

Conclusion

- implement as a **fused operator**
- exploit FPGA flexibility: **exotic formats**, fixed-point and floating-point
- careful error analysis → **compute just right**

References

[1] DSP Builder Advanced Blockset. http://www.altera.com/technology/dsp/advanced-blockset/dsp-advanced-blockset.html.

[2] Florent de Dinechin and Bogdan Pasca. Designing custom arithmetic data paths with FloPoCo. *IEEE Design and Test*, 2011.

[3] Jérémie Detrey and Florent de Dinechin. Floating-point trigonometric functions for FPGAs. In *International Conference on Field Programmable Logic and Applications*, pages 29–34, Amsterdam, Netherlands, aug 2007. IEEE.

[4] E.I. Garcia, R. Cumplido, and M. Arias. Pipelined cordic design on fpga for a digital sine and cosine waves generator. In *Electrical and Electronics Engineering, 2006 3rd International Conference on*, pages 1 –4, sept. 2006.

[5] Martin Langhammer and Tom VanCourt. FPGA floating point datapath compiler. *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, 17:259–262, 2009.

[6] Bogdan Pasca. Correctly rounded floating-point division for DSP-enabled FPGAs. In *22th International Conference on Field Programmable Logic and Applications (FPL'12)*, Oslo, Norway, August 2012. IEEE.

[7] Yalei Shang. Implementation of ip core of fast sine and cosine operation through FPGA. *Energy Procedia*, 16, Part B(0):1253 – 1258, 2012. 2012 ICFEEM.