



Floating-Point Madness

May 2002

Presenter:

Justin Schauer



Floating-point Numbers

- Used extremely often
- A method of approximating any real number (whole or fraction) to a certain precision using a string of bits
- IEEE-754 is the standard for binary floating-point

Floating-point Representation



- Where the formula to calculate the real number represented by this floating point number is:

$$(-1)^s \times 1.f \times 2^{e-\text{bias}}$$

- The bias is determined by the number of exponent bits

IEEE-754 Standard

- Sets two standard precisions
 - Single: 8 exponent bits, 23 significand bits
 - Double: 11 exponent bit, 52 significand bits
- Four rounding modes
 - Round to nearest even, round to positive infinity, round to negative infinity, round to zero
- Dictates handling of special cases
 - Infinities, invalid numbers (NaNs), denormal numbers
- Four exceptions
 - Invalid operation, underflow, overflow, inexact result



Floating-Point Madness

- Library of floating-point hardware components
- Conform fully to IEEE-754 standard
- Written in Verilog hardware description language
- Verilog code free to anyone (open source)

Motivation

- Currently very few open source floating point component libraries
- Current libraries are very expensive
- Rise in FPGA use and decrease in cost
- Time-intensive process to design and test floating-point components
- Solution: fully developed, tested, open source components!

My Project

- Floating-point adder
- Fully IEEE-754 compliant
- Optimized for smallest possible area
 - Can more easily fit on FPGAs
- Fully modular design
 - Choice of exponent and significand bits
 - Rounding mode, special case, and exception handling can be removed

Modularization

- All data bus widths are set by constants
- Verilog is parsed by a CGI script using VERIMark parsing language
 - Developed by Mark Phair
 - Takes full, marked adder as input
 - Takes form information from website as input
 - Produces specific adder user desires

Demonstration

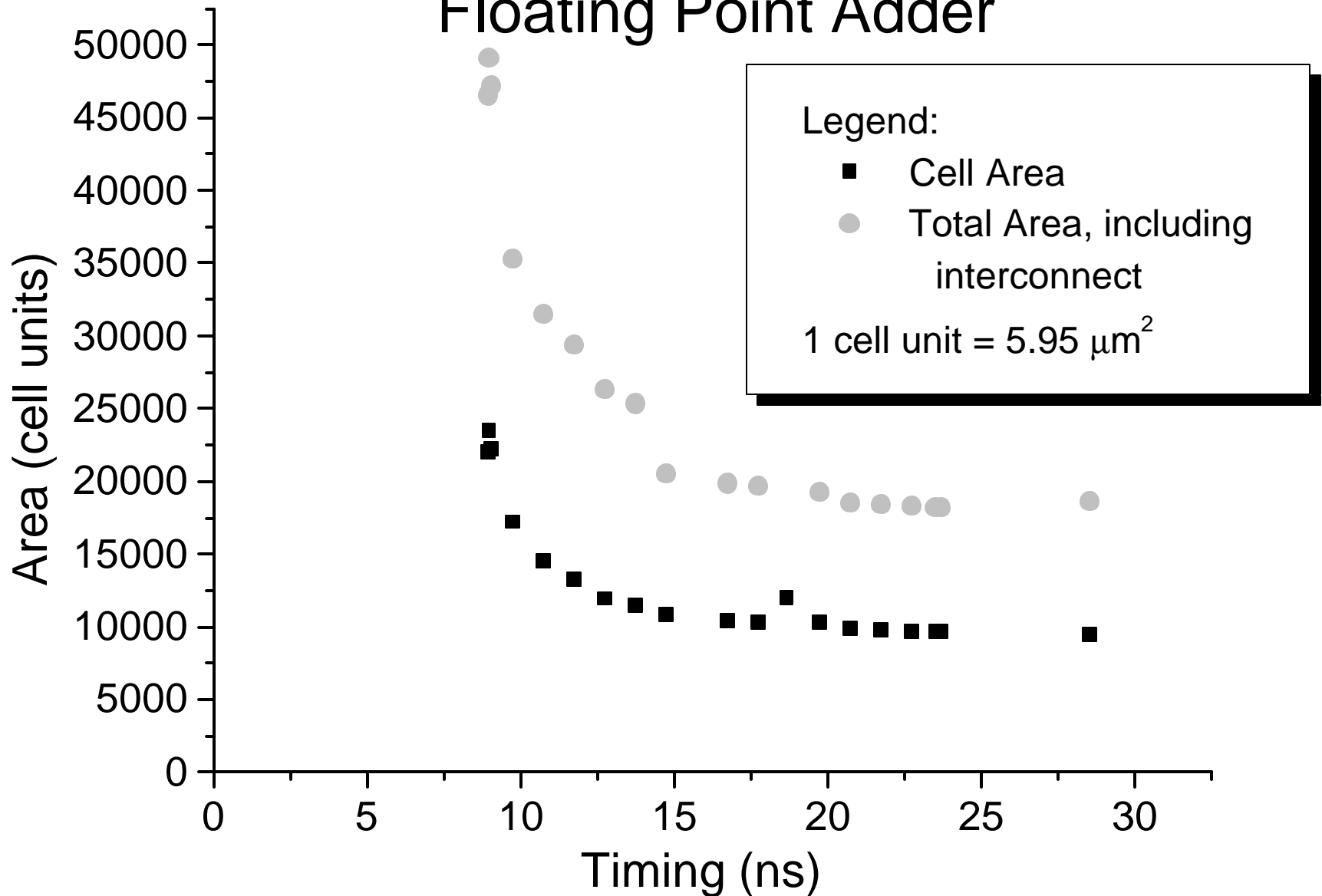
<http://www.hmc.edu/chips>



Last Summer's Work

- Extensive floating-point research
- Attended IEEE Computer Arithmetic Conference
- Coded adder
- Did preliminary verification vs. Intel FPU
- Compiled code to get estimated area
- Marked up adder with VERIMark

Area vs Timing Plot for Area Optimized Floating Point Adder



This Semester's Work

- Came up with test plan
- Performed extensive verification
 - Using a full software implementation of IEEE-754 standard
- Fixed bugs that arose from testing
 - Fundamental rounding problem
 - Magnitude subtraction of NaNs

Software Adder

- As versatile as hardware adder
- Produces test cases to verify hardware adder
- Not designed by me
- Originally Java design by Hang Tang
 - I am a terrible software programmer
- Re-designed in C by Prof. Harris
 - Checks results vs. Intel FPU when possible

Progress

- Checked adder with old testing program for limited cases
- Exhaustively verified at 1 sign, 3 exponent, 3 significand bits
 - 65536 test cases
- Performed all directed tests in single and double precision
 - 1871424 test cases each
- Performed 50 million random test cases in single and double precision

Future Work

- Exhaustive verification at 4-5 exponent bits and 4-6 significand bits
- Test and bug fix with features disabled
- Examine area costs of the features
- New floating-point components
 - Speed optimized adder
 - Other floating-point components

Solicitation

- Are you interested in designing floating-point components?
- Do you want to learn Verilog really well?
- Do you want access to the really neat VLSI lab?
- You can be part of a brand-new and exciting field!

Talk to Prof. Harris about researching with him and visit our website at: <http://www.hmc.edu/chips>

Acknowledgements

- Professor David Harris
 - Advisor
 - Wrote software adder
- Mark Phair
 - Created VERIMark
 - Helped with website
- Hang Tang
 - Wrote software adder



Questions?

