Target Design: ASIC and FPGA design to implement mathematical algorithms which should not use general-purpose IP cores such as CPU, DPC and GPU but design special-purpose hardware from scratch.

What to standardize: A new design language supporting both the description of mathematical algorithms, as well as Verilog, which is an IEC standard language. An example of such a language is FinSimMath, which is an extension of Verilog for mathematical descriptions.

Why now, what is the urgency: the power consumption of dedicated circuits is smaller than that of a processor-based system addressing the same problem by more than 100x. Also the speed of execution is much faster. Numerous projects for accelerating mathematical computations are being considered at this time. The pent up demand for ASICs in general is presented in www.fintronic.com/manuals/IEEE_HDLMath_Study_Group.pdf section 2. In section 5 of the same document it is pointed out that the EDA industry is concerned now with the issue of bringing together design and implementation which is best served by bringing in the same language both the mathematical and the Verilog levels of abstraction.

Merits: The new design language brings an increase in productivity compared to the current situation similar to the productivity increase brought by Verilog when it included for the first time RTL and Gate level description capabilities in the same language, allowing the design at RTL level and the implementation at the Gate level. The productivity increase is estimated to exceed 1000x based on the available precedent Please note that the only way to achieve such a productivity increase is to bring the two levels of abstraction (math and Verilog) in the same language. This is supported both by the precedent with Verilog mentioned above and by the Industry awareness expressed in forums such as the panel discussion mentioned in www.fintronic.com/manuals/IEEE_HDLMath_Study_Group.pdf section 5.

The new language brings an additional productivity increase due to special features other than having the two levels of abstraction in the same language. For example, the mathematical part must be included in a simulation environment which differentiates it from Mathematica which is the best mathematical environment but is not supported by a simulator. The m-language which is supported by the SimuLink simulator, lacks numerous features required by a language that supports the implementation of mathematical algorithms as ASICs and FPGAs. Some examples of such features lacking in m-language are: (1) the m-language does not support the gate-level description of circuits along with the mathematical description, making it impractical to use mixed-level assertions and to switch easily between levels of abstraction for modeling the various sub-modules for faster simulation, (2) the declaration of data structures is not part of the m-language making it more difficult to program operations of objects with different types and sizes, 3) formats (fixed point, floating point) and sizes of fields cannot be modified during the simulation for each data container, making is more difficult to explore the design space, 4) there is no support for exceptions such as overflow and underflow, 5) mixing numerical and symbolic computations in the same simulation; as a result the user must copy the results of symbolic computations and paste them in the numerical computation by hand.

A more detailed list of requirements for the new language is presented in www.fintronic.com/manuals/IEEE_HDLMath_Study_Group_Info.pdf and an example of using FinSimMath and several of its features is available in www.fintronic.com/Integrating_FinSimMath_in_an_Existing_Design_Flow.pdf. Other examples of using FinSimMath are available in links present in www.fintronic.com/finmath.html.