# A Fully-Pipelined Single-Precision Floating Point Unit in the Synergistic Processor Element of a CELL Processor

Hwa-Joon Oh[1], Silvia M. Mueller[2], Christian Jacobi[2], Kevin D. Tran[1], Scott R. Cottier[1], Brad W. Michael[1], Hiroo Nishikawa[3], Yonetaro Totsuka[4], Tatsuya Namatame[5], Naoka Yano[5], Takashi Machida[5], Sang H. Dhong[1]

[1]IBM System and Technology Group, Austin, TX
[2]IBM Entwicklung GmbH, Boeblingen, Germany
[3]IBM Engineering and Technology Services, Yasu, Japan
[4]Sony Computer Entertainment of America, Austin, TX
[5]Toshiba America Electronic Components, Austin, TX

## Abstract

The floating point unit in the synergistic processor element of a CELL processor is a fully-pipelined 4-way SIMD unit designed to accelerate media and data streaming. It supports 32-bit single-precision floating point and 16-bit integer operands with two different latencies, optimizing the performance of critical single-precision multiply-add operations. It employs fine-grained clock gating for power saving. Architecture, logic, circuits and integration are co-designed to meet the performance, power, and area goals.

**(Keywords: VLSI, CMOS, SOI, Microprocessor, Static circuits, and Floating point unit)**

## Introduction

The Synergistic Processing Element (SPE) [1] of a CELL processor is the first implementation of a new processor architecture designed to optimize the multimedia applications, such as 3D graphics, media streaming, and signal processing [2]. Real time multimedia applications demand single precision performance significantly exceeding that of conventional processors. A single-precision floating-point unit (FPU) of the SPE is a 4-way single-instruction multiple-data (SIMD) design. Most instructions in the FPU process 128-bit operands, divided into four 32-bit words. Each of the 4 slices supports 32-bit single-precision and 16-bit integer multiply-add instructions and converts between floating-point and integer. The FPU has two different latencies, 6-cycle for single-precision and 7-cycle for integer and converts including result forwarding (Fig. 1).

## Challenges in 11FO4 design

The amount of logic per pipeline stage is 11 fanout-of-4 (FO4) inverter delays. Since result forwarding takes 6FO4, the delay budget including latches is 60FO4 for the whole FPU logic. A state-of-art FPU has a latency of around 100FO4 [3]. Saving 40FO4 and being power efficient for the high volume market required optimizations at all design levels: architecture, logic, circuits, layout, and placement.

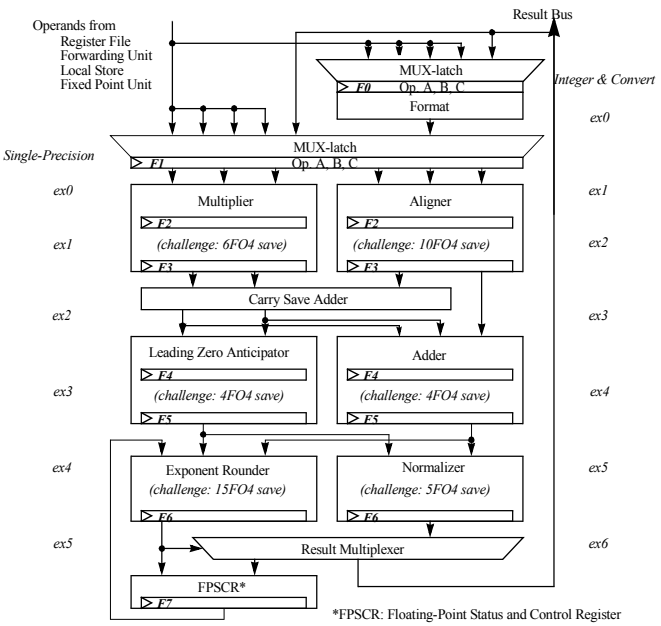


Fig. 1. Overview of the FPU: Design challengers are presented at each block to make a 6-cycle single-precision operation and 7-cycle integer operation with 11FO4 cycle time.

The FPU is optimized for single-precision multiply-add type instructions [4]. With truncation rounding and extended data set, denormal numbers are flushed to zero and NaNs are handled as normal numbers. An extra format stage is added to pre-process the integer and convert instructions while a late zero correction in aligner and multiplier allows single-precision operations to bypass the format stage, saving one cycle.

Latch insertion delay of 2FO4 to 3FO4 occupies a significant portion of the 11FO4 cycle. In order to minimize this delay, NAND logic is merged with latch function. Also, there is a special type of latch which has multiplexers (MUX) at the input terminals [5]. Both types of latches are heavily used at the FPU to improve timing. CMOS static gates are used to implement most of the logic.

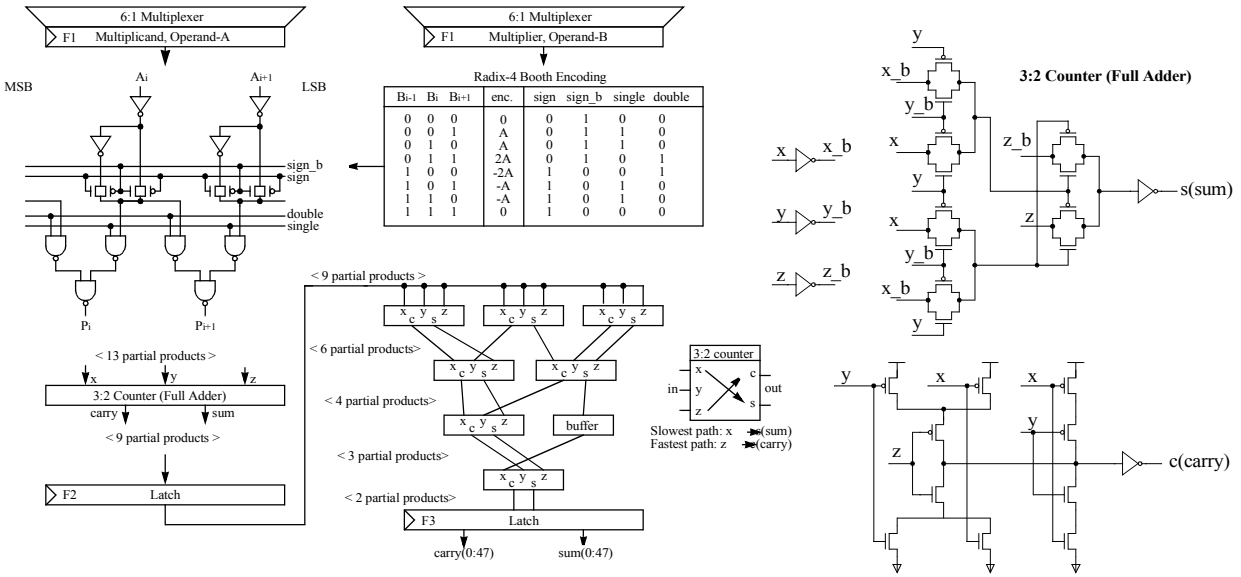


Fig. 2. Radix-4 Booth recording and partial product reduction trees at the multiplier. The 3:2 counter (Full Adder) circuits have static AOI and transmission-gate circuits.

### The multiplier and the aligner

The 2-cycle multiplier (Fig. 2) uses a radix-4 Booth encoding; it supports unsigned and 2's complement numbers. Partial product reduction tree uses static full adders (FA) consisting of transmission gate XOR and AOI. The static implementation has advantages over dynamic circuits of reduced wiring need and improved timing optimization. Dynamic adders need true and complement signals; static circuits cut the wire density in half. The timing is optimized exploiting the different path delays in a static FA [6]; in a dynamic FA every signal has the same block delay. Since the FA has the slowest path from x to sum and the fastest path from z to carry, the carry output of one stage is connected to the x input of the next stage. By doing so, we are able to put 4 FA reduction tree stages in 8.5FO4.

The aligner block shifts the addend based on the exponent difference (ED) in order to align it with the product. The shifter is broken into several 4:1 MUX stages. The exponent difference is usually generated by an adder then decoded to select signals for the MUX stages, such that both, adder and decoder, add critical path delay to the aligner (Fig. 3). Our 2-cycle aligner uses a special sum-addressed scheme; adding and decoding of the exponent difference are done in parallel. The last MUX stage is integrated into the special MUX latch, reducing the effective latch insertion delay.

Main 4:1 MUX stages at the aligner consist of 4-input transmission-gate multiplexers (TG-MUX) (Fig. 3). Its output inverter driver has better driving capability over the long horizontal wires than that of the AOI-type MUX. Its short cycle time (e.g., 11FO4) forces to insert non-scan staging latches in the middle of the select signal generations. To solve the reliability problem, special circuits have been implemented at local clock buffer; start latching function as soon as possible when the VDD voltage level becomes stable.

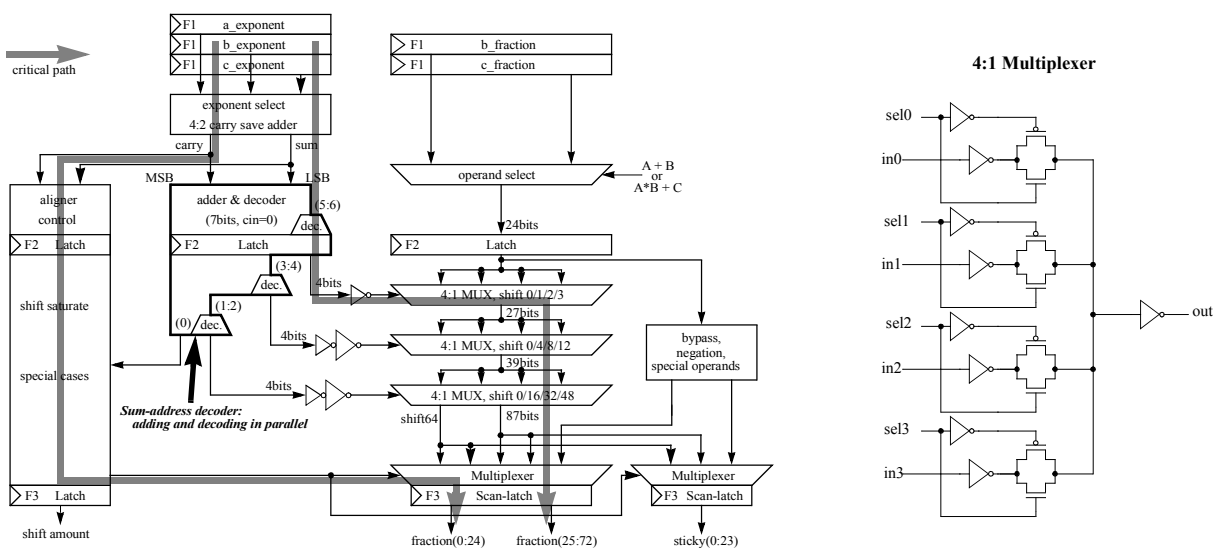


Fig. 3. Sum-address decoder speeds up the shifting stages at the aligner. Main shifting stage consists of 4-input TG-MUXes.

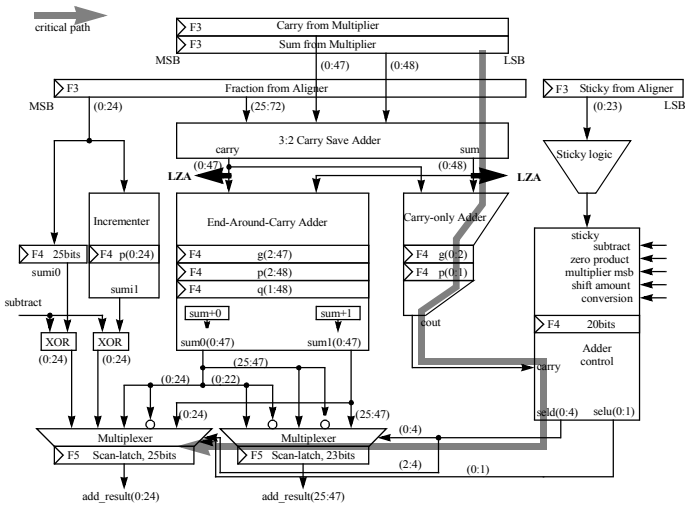


Fig. 4. Critical path of the adder: generating select signals of the multiplexer.

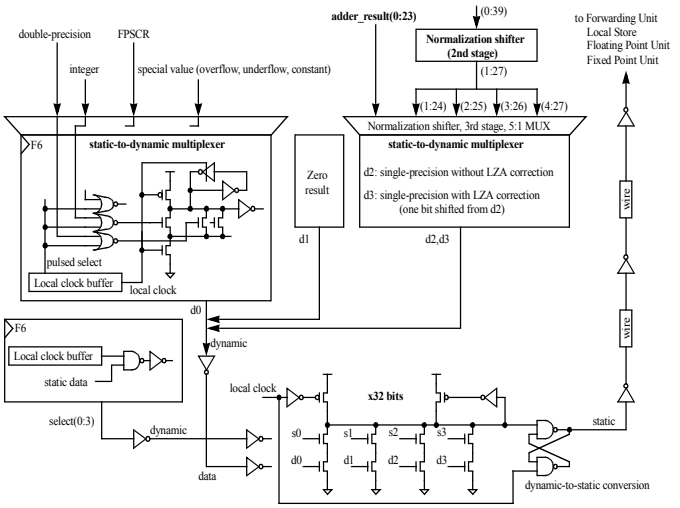


Fig. 5. Static-to-dynamic multiplexer generates the dynamic signals, and then the dynamic result MUX converts the signals to static forwarding signals.

### The fraction adder and the leading zero anticipator

The fraction adder (Fig. 4) computes the sum or absolute difference of its inputs using the end-around-carry concept. It is partitioned as incrementer, adder and sticky part. The main adder generates sum (sum0) and sum+1 (sum1). Instruction type, shift amount in the aligner, the sticky bit, and the carry-out (cout) of the adder determine whether *sum0* or *sum1* is chosen, and whether complement of the result is needed. An incrementer generates *sumi1* incremented by one from the 25-bit MSB of addend (sumi0). The subtraction-type instructions re-complement both *sumi0* and *sumi1*. The selection of *sum0*, *sum1*, *complement of sum0*, *sumi0*, and *sumi1* is implemented with the special 6-port MUX latch. The path delay of this multiplexing is hidden by the latch insertion delay. The computation of the select signals based on the *cout* is the critical path of the adder. An extra carry-only adder is used to speed up signal *cout*.

The leading zero anticipator (LZA) estimates the number of leading zeros in the adder result. LZA correction logic checks whether the estimate is accurate or too large by one bit.

### The normalizer and the result multiplexer

The normalizer block normalizes the fraction, shifting out the leading zeros. Moving one shift stage to the adder and integrating the last shift stage in a special MUX latch allows for a 1-cycle normalizer.

The FPU only supports round toward zero and non-trap exception conditions. That simplifies the rounding step considerably; the fraction is truncated and no exponent wrapping is needed. The fraction rounder is implemented as a result-MUX. Exponent rounder (ERND) adjusts the exponent, detects exceptions and controls the result-MUX. To match the fast rounder, the ERND delay was improved by 15FO4 using highly optimized checkers and dynamic circuits. ERND uses multiple pre-computed exponent copies (e, e+1, e-1), and a special MUX latch selects the appropriate copy of the ERND results.

The result-MUX requires four select signals: a) results from integer or double precision, b) a zero result, c) single-precision results which need LZA correction, and d) single-precision results which do not need LZA correction. Since the last cycle has only 5FO4 budget due to the result forwarding, a dynamic 4:1 MUX (Fig. 5) was designed. The fraction data and the select signals from ERND use a local clock signal (50% duty-cycle). The output of the result-MUX is converted to static signals through cross-coupled NAND2 circuits.

### Physical designs

Physical design considerations such as wiring, pin locations, and floor-plan are carefully considered (Fig. 7). The physical width of each FPU slice is 46 bits and aligned to the operand sources such as the register file. This is very narrow for a FPU since intermediate results require 11-bit exponent and 96-bit fraction. The data flow is split into an exponent and a fraction part. Most fraction macros are folded and their layouts use two different bit images, 16-track/bit and 12-track/bit. 16-track/bit regions assign 9 bits for exponent and 37 bits for fraction, whereas 12-track/bit regions assign 12 and 49 bits, respectively.

From early in the project, wire and re-powering delays were limited to 30% of a cycle. Since wires determine the size of the unit and have timing impacts, strict wiring assignments are applied. Macro level wiring was restricted to Metal-3 and below. Some macros have mandatory empty wiring tracks at Metal-2 for global wiring.

### Clock gating to save power consumption

Clock-gating is extensively used to reduce the active power consumption. Only pipeline stages with valid instructions are activated by controlling the local clock buffer. Also,

circuit blocks are clocked down based on instruction type and on operand values, e.g., the multiplier is bypassed for add-type instructions, and the aligner is idle for multiply-type instructions.

## Conclusions

The design has roughly 450K transistors in 1.3mm2, fabricated in 90nm SOI technology with 8 levels of copper interconnects. Correct operation has been observed up to 5.6GHz, 1.4V of supply voltage, and 56°C, delivering a peak performance of 44.8GFlops. An interpolated hardware power of 1.4W at 4GHz (e.g., 32GFlops), gives 43.75mW/GFlop.

The VDD voltage vs. operating frequency schmoo plot is shown in Fig. 6., when a SPE is running single-precision floating point computations for lightning and transform workloads.

There are 3 key enablers for this low latency, power efficient, and high frequency FPU: architecture and implementation are optimized for target applications, trading uncritical function for overall performance; logic, circuits and integration are co-designed; the pipeline stages are carefully balanced, achieving the maximum path delay difference of 3% between stages.

## Acknowledgments

We thank Osamu Takahashi, Gordon C. Fossum and Barry L. Minor for design efforts, Gilles Gervais and SPU verification team for supports, and Linda V. Grinsven and Robert Putney for the management team.

## References

[1] B. Flachs et. al., "The microarchitecture of the streaming processor for a CELL processor," accepted for 2005 IEEE International Solid-State Circuits Conference (ISSCC-2005).

[2] O. Takahashi et. al., "The circuits and physical design of the synergistic processor element of a CELL processor", submitted to 2005 VLSI Circuits Symposium (VLSI-2005).

[3] N. Rohrer et. al., "PowerPC 970 in 130nm and 90nm technologies," ISSCC Dig. Tech. Papers, pp.68-69, Feb. 2004.

[4] S. M. Mueller et. al., "The vector floating-point unit in a synergistic processor element of a CELL processor," submitted to 2005 IEEE 17th Symposium on Computer Arithmetic (ARITH-17).

[5] D. Pham et. al., "The design and implementation of a first-generation CELL processor," accepted for 2005 IEEE International Solid-State Circuits Conference (ISSCC-2005).

[6] V. G. Oklobdzija et. al., "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Transactions on Computers, Vol. 45, No.3, pp. 294-305, March 1996

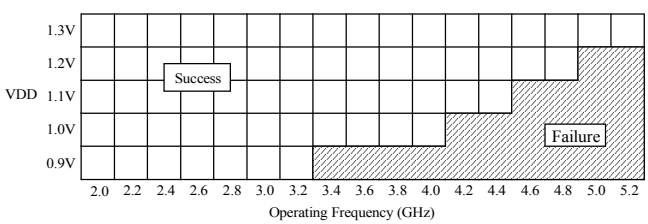


Fig. 6. Voltage vs. Frequency Schmoo: A Synergistic Processing Element is running single-precision floating point computations for lighting and transform workloads.

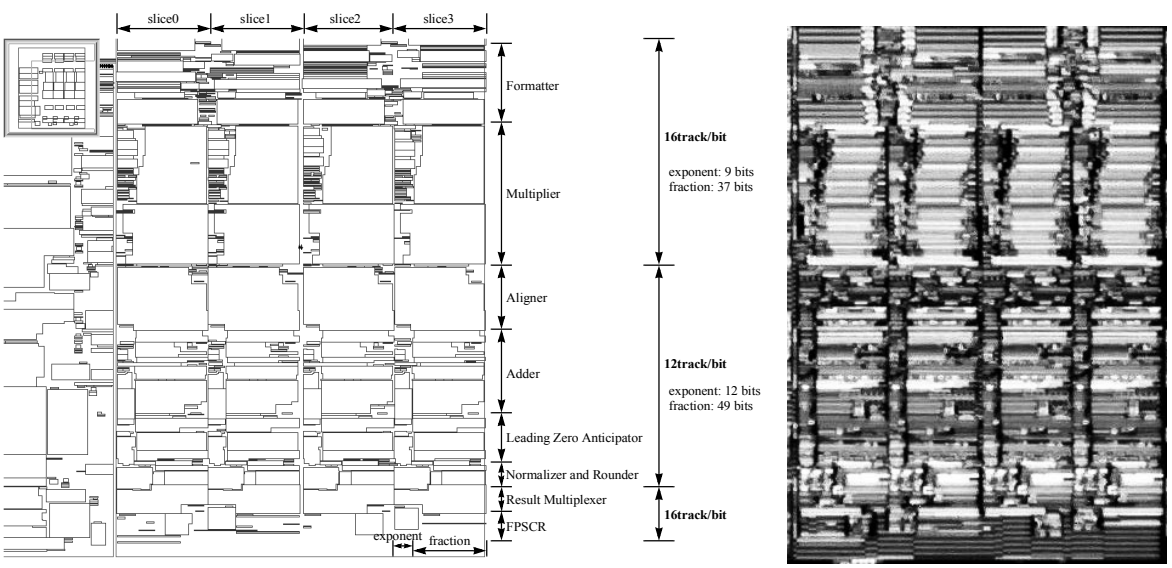


Fig. 7. Macro placement and Die photo: Macro placements are shown at left figure. Two different bit images are used, 16track/bit and 12track/bit. Die photo of FPU in the Synergistic Processing Element are shown at right photo.