US011635957B2

(12) **United States Patent**
Harthcock

(10) **Patent No.: US 11,635,957 B2**
(45) **Date of Patent: Apr. 25, 2023**

(54) **HARDWARE-IMPLEMENTED UNIVERSAL FLOATING-POINT INSTRUCTION SET ARCHITECTURE FOR COMPUTING DIRECTLY WITH HUMAN-READABLE DECIMAL CHARACTER SEQUENCE FLOATING-POINT REPRESENTATION OPERANDS**

(71) Applicant: **Jerry D. Harthcock**, Boerne, TX (US)

(72) Inventor: **Jerry D. Harthcock**, Boerne, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **17/591,963**

(22) Filed: **Feb. 3, 2022**

(65) **Prior Publication Data**

US 2022/0156070 A1 May 19, 2022

Related U.S. Application Data

(62) Division of application No. 16/943,077, filed on Jul. 30, 2020, now Pat. No. 11,275,584.
(Continued)

(51) **Int. Cl.**
G06F 9/30 (2018.01)
G06F 7/483 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06F 9/30025*** (2013.01); ***G06F 5/00*** (2013.01); ***G06F 7/483*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC .. G06F 9/30025; G06F 7/483; G06F 9/30079; G06F 9/30101; G06F 9/3557;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

2007/0061387 A1* 3/2007 Carlough H03M 7/24 708/204
2007/0203965 A1* 8/2007 Reynolds G06F 40/111 708/200

(Continued)

OTHER PUBLICATIONS

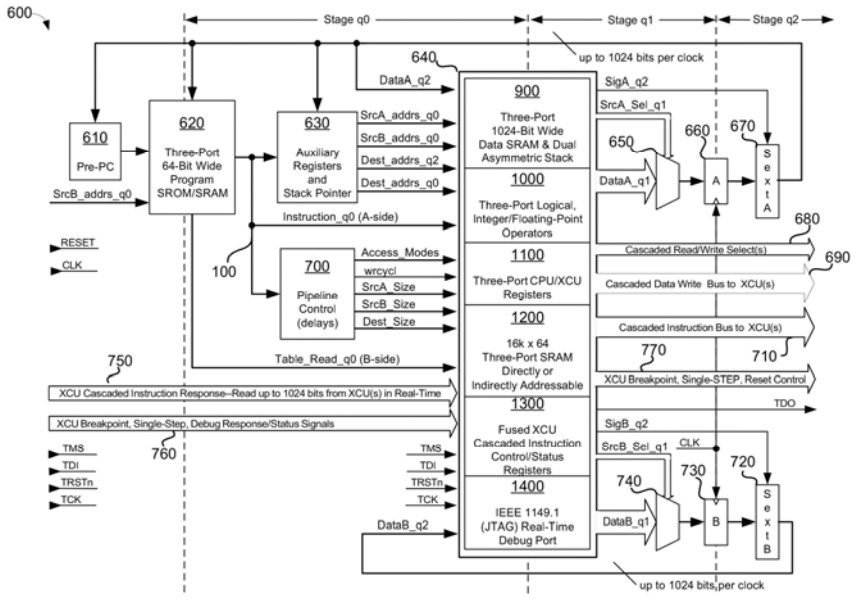
L. -K. W. et al., Hardware Designs for Decimal Floating-Point Addition and Related Operations, IEEE Transactions on Computers, vol. 58, No. 3, Mar. 2009, pp. 322-335 [online], [retrieved on Nov. 9, 2022]. Retrieved from the Internet <URL: https://ieeexplore.ieee.org/document/4599577> <doi: 10.1109/TC.2008.147>.*

Primary Examiner — Shawn Doman
(74) *Attorney, Agent, or Firm* — Steven W. Smith

(57) **ABSTRACT**

A universal floating-point Instruction Set Architecture (ISA) compute engine implemented entirely in hardware. The ISA compute engine computes directly with human-readable decimal character sequence floating-point representation operands without first having to explicitly perform a conversion-to-binary-format process in software. A fully pipelined convertToBinaryFromDecimalCharacter hardware operator logic circuit converts one or more human-readable decimal character sequence floating-point representations to IEEE 754-2008 binary floating-point representations every clock cycle. Following computations by at least one hardware floating-point operator, a convertToDecimalCharacterFromBinary hardware conversion circuit converts the result back to a human-readable decimal character sequence floating-point representation.

20 Claims, 89 Drawing Sheets



Related U.S. Application Data

(60) Provisional application No. 62/886,570, filed on Aug. 14, 2019.

(51) **Int. Cl.**

G06F 12/06	(2006.01)
G06F 9/355	(2018.01)
G06F 9/54	(2006.01)
H03M 7/24	(2006.01)
G06F 5/00	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 9/30079*** (2013.01); ***G06F 9/30101*** (2013.01); ***G06F 9/3557*** (2013.01); ***G06F 9/544*** (2013.01); ***G06F 12/063*** (2013.01); ***H03M 7/24*** (2013.01)

(58) **Field of Classification Search**

CPC G06F 9/544; G06F 12/063; G06F 9/30; G06F 9/321; G06F 9/325; G06F 9/35; G06F 9/38; G06F 9/3824; G06F 2212/1048; G06F 12/0292; G06F 2212/1024; G06F 12/0284; H03M 7/24

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2007/0266073	A1*	11/2007	Cornea-Hasegan	G06F 7/4912 708/680
2009/0240753	A1*	9/2009	Carlough	G06F 7/483 708/442
2014/0046994	A1*	2/2014	Kamoshida	G06F 7/483 708/505
2014/0101215	A1*	4/2014	Ayoub	H03M 7/12 708/204
2015/0089205	A1*	3/2015	Carlough	G06F 9/3016 712/222
2017/0322768	A1*	11/2017	Brooks	G06F 5/00
2021/0034328	A1*	2/2021	Payer	G06F 7/49947

* cited by examiner

Since all the FMAs share the same size and signal inputs, the numbers making up 1024-bit operandA gob must be the same format and numbers making up 1024-bit operandB gob must be the same format, but gob A can be a different format than gob B.

Like the Universal FMA in **9800**, each Universal FMA in **9900** has a split 32-entry result buffer that is 37 bits wide (instead of 69 bits wide as shown in **9816**), which is wide enough to accommodate a binary32 format result plus five exception bits. Results are read out as one 512-bit gob or one 256-bit gob for each result buffer location. The wider gob is for situations where the exception bits for each FMA are read out as part of that FMA's results in their respective position in the gob, in which case 64-bit words are pulled out in one 512-bit gob. Many applications can do without the exception bits, in such cases results can be read out as quantity (8) binary32 format numbers in one 256-bit gob.

Once pushed into fat memory, results can be pulled out in properly aligned 1, 2, 4, 8, 16, 32, 64, or 128-byte accesses. For example, the results of a single 128-byte push into fat memory can be read out as a sequence of quantity (128) 1-byte pulls, (64) 2-byte pulls, (32) 4-byte pulls, and so on. However, care must be exercised to ensure that the original 128-byte push is on a even 128-byte address boundary. The results of a single 64-byte push into fat memory can be read out as a sequence of quantity (64) 1-byte pulls, (32) 2-byte pulls, (16) 4-byte pulls, and so on. Like the previous example, care must be exercised to ensure that the original 64-byte push is on an even 64-byte address boundary. Thus the same can be done for pushes of 32, 16, 8, etc. bytes in length.

Each of the eight FMAs in the Universal FMA operator **9900** can be used as sum-of-products operators, in that each has quantity 32 C-register/accumulators and operate the same was as those in the Universal FMA **9800**.

Finally, for intensive pure convertFromDecimalCharacter to binary conversion operations, the Universal FMA operator **9900** can be used to convert quantity (16) H=12 decimal character sequences every clock cycle.

FIG. **50**A thru FIG. **50**L comprise a example assembled source listing of a actual working 3D transform program **9950** written in the present disclosure's ISA assembly language that employs up to qty. (16) child XCUs to perform a 3D transform (scale, rotate, translate) on all three axes of a 3D object in .STL file format and write the resulting transformed 3D object back out to external memory when the transformation is complete. This program was simulated using the XILINX® Vivado® development environment (Vivado HLx Edition, v2018.2 (64-bit)) targeted to its Kintex® UltraScale and UltraScale+ brand FPGAs.

Referring to 3D transform program **9950**, the parent CPU first determines how many child XCUs are available to perform the transformation by first counting the active DONE, one corresponding to each XCU, and then divides the number of triangles that make up the 3D object as evenly as possible. The parent then performs a "poke-all" monitor instruction to push the thread code simultaneously into all XCUs. It sets a breakpoint simultaneously on all XCUs at the same location in their program memory and then simultaneously asserts then releases a reset so that all XCUs begin to execute at the same location until they all reach their breakpoint.

The parent CPU then pushes the number of triangles each XCU is to transform, the location of where the triangles are located in XCU memory space, and then the triangles themselves. The parent then releases the breakpoint and single-steps all child XCUs out of the break state simultaneously, at which time the XCUs begin processing the triangles. During this time, each XCU brings its DONE bit inactive low, to signal it is busy. At this time the parent is monitoring its XCU status register for the DONE bits to be brought back to active high, indicating XCU processing is complete. When complete, the parent XCU then pulls the transformed triangles from each XCU result memory and pushes them back out to external memory.

If, at the initial stages described above, the parent CPU determines there are no XCUs available to perform the transform, the parent CPU performs the entire 3D transform solo (without the use of any XCU) and pushes the transformed triangles back out to external memory when complete. It should be understood that the parent CPU and the child XCUs (if any) execute the same instruction set. In this implementation, the 3D transform routine physically resides in the parent CPU program memory space. Thus, in this instance, when the parent performs a "push-all" of the required routine and parameters, it pulls the routine code out of its program memory space and pushes it into all XCUs program memories simultaneously. If the required program is not resident in the CPU's program space, it could alternatively pull it in from any child XCU program memory space or external memory space.

FIG. **51** is an actual wire-frame "Before" and "After" rendering **9955** of a simple "olive" 3D model in .STL file format performed by from 1 to 16 child XCUs or solo parent CPU using the scale, rotate, and translate parameters shown for each axis.

In the drawings and specification, there have been disclosed typical preferred embodiments of the disclosure and, although specific terms are employed, they are used in a generic and descriptive sense only and not for purposes of limitation, the scope of the invention being set forth in the following claims.

What is claimed is:

1. A universal floating-point Instruction Set Architecture (ISA) compute engine implemented entirely in hardware, said ISA compute engine comprising:

means for converting human-readable decimal character sequence floating-point representation operands to binary format without explicitly performing a conversion-to-binary-format process in software; and

means for computing directly with the converted human-readable decimal character sequence floating-point representation operands.

2. The universal floating-point ISA compute engine as recited in claim **1**, wherein the means for converting includes means for accepting the human-readable decimal character sequence floating-point representation operands up to IEEE 754-2008 H=20 in length.

3. A universal floating-point Instruction Set Architecture (ISA) compute engine implemented entirely in hardware, said ISA compute engine comprising:

a program memory; and

hardware circuitry connected to the program memory, said hardware circuitry configured to compute directly with human-readable decimal character sequence floating-point representation operands using a single instruction per floating-point operation, the hardware circuitry comprising:

a data memory having a write port and first and second read ports, said data memory configured to write data into the write port while simultaneously reading binary format operands or human-readable decimal character sequence floating-point representation operands from the first and second read ports;

at least one fully pipelined convertToBinaryFromDecimalCharacter hardware conversion circuit connected to the data memory and having one or more inputs and an output, the convertToBinaryFromDecimalCharacter hardware conversion circuit being configured to:

receive the binary format operands or human-readable decimal character sequence floating-point representation operands from the data memory,

convert the received human-readable decimal character sequence floating-point representation operands, if any, to IEEE 754-2008 binary floating-point representations, and

preserve the received binary format operands, if any, as unconverted binary format operands;

a multistage instruction pipeline logic circuit configured to fetch, decode, and execute instructions from the program memory, wherein, a single instruction that specifies one or more read addresses in the data memory causes one or more operands to be set in-flight within the convertToBinaryFromDecimalCharacter hardware conversion circuit, wherein the instruction pipeline logic is configured to:

simultaneously read from the first and/or second read ports of the data memory, at least one IEEE 754-2008 binary floating-point representation operand and/or at least one human-readable decimal character sequence floating-point representation operand residing at the one or more read addresses specified by the single instruction, and

simultaneously write the one or more operands to the one or more inputs of the at least one fully pipelined convertToBinaryFromDecimalCharacter hardware conversion circuit; and

at least one memory-mapped, fully pipelined, hardware floating-point operator connected in series with the output of the at least one convertToBinaryFromDecimalCharacter hardware conversion circuit, the at least one hardware floating-point operator automatically receiving the at least one IEEE 754-2008 binary floating-point representations and utilizing the received at least one IEEE 754-2008 binary floating-point representations as operands for use in the at least one hardware floating-point operator's computation when a destination address in the single instruction corresponds to the memory-map address of the at least one hardware floating-point operator, wherein results computed by the at least one hardware floating-point operator automatically spill into the data memory write port at the destination address specified by the single instruction.

4. The universal floating-point ISA compute engine as recited in claim **3**, wherein the multi-stage instruction pipeline logic circuit is configured to fetch, decode, and execute instructions that do not include an opcode field.

5. The universal floating-point ISA compute engine as recited in claim **3**, wherein the data memory includes hardware logic configured to enable results that automatically spill into it from the at least one hardware floating-point operator to be read out and immediately restored without having to re-compute the results.

6. The universal floating-point ISA compute engine as recited in claim **3**, wherein the hardware circuitry is configured to implement an instruction set configured to specify with the single instruction:

a destination address for spilling results from the at least one hardware floating-point operator into the data memory write port; and

at least one operand source address for reading from the data memory first and/or second read ports, operands used in the floating-point operation.

7. The universal floating-point ISA compute engine as recited in claim **3**, wherein the hardware circuitry is configured to implement an instruction set configured to specify with the single instruction, a rounding mode to be carried out as a step in the at least one hardware floating-point operator, wherein the at least one hardware floating-point operator includes hardware circuitry configured to perform the rounding step according to the rounding mode specified in the single instruction.

8. The universal floating-point ISA compute engine as recited in claim **3**, wherein when the single instruction is preceded by a REPEAT instruction, hardware logic within the multistage instruction pipeline is configured to cause the single instruction to be repeated an additional number of times specified by a REPEAT value in the REPEAT instruction.

9. The universal floating-point ISA compute engine as recited in claim **8**, wherein:

the at least one convertToBinaryFromDecimalCharacter hardware conversion circuit and the at least one hardware floating-point operator each include at least one stage, and each repetition of the single instruction causes a next stage to be filled with a result from an immediately preceding stage each clock cycle; and

when the REPEAT value is greater than or equal to a total combined number of stages in the at least one convertToBinaryFromDecimalCharacter hardware conversion circuit and the at least one hardware floating-point operator, and when the final repeated single instruction sets the last one or more operands in-flight into the at least one convertToBinaryFromDecimalCharacter hardware conversion circuit, the results from the first repeated single instruction have already spilled into the data memory write port and are available for reading, thereby effectuating an apparent single-clock latency to perform both the convertToBinaryFromDecimalCharacter conversion and the floating-point operator computation;

wherein the operand first and second read addresses and the result write address for the data memory are automatically added to by respective amounts specified by the single instruction each time the single instruction is repeated.

10. The universal floating-point ISA compute engine as recited in claim **9**, wherein the human-readable decimal character sequence floating-point representations are IEEE 754-2008 H=20 in length, and there are two fully pipelined convertToBinaryFromDecimalCharacter hardware conversion circuits operating in parallel with each convertToBinaryFromDecimalCharacter hardware conversion circuit having up to 64 stages, said hardware circuitry of the ISA compute engine being thereby configured to convert the H=20 human-readable decimal character sequence floating-point representations to IEEE 754-2008 binary floating-point representations in 64 or fewer clock cycles.

11. The universal floating-point ISA compute engine as recited in claim **10**, wherein the at least one hardware floating-point operator is configured to perform a binary64 MUL operation in four or fewer clock cycles or a binary64 ADD operation in eight or fewer clock cycles and to

automatically spill the result into the data memory write port at the destination address specified by the single instruction.

12. The universal floating-point ISA compute engine as recited in claim **11**, wherein the hardware circuitry of the ISA compute engine further comprises a convertToDecimalCharacterFromBinary hardware conversion circuit configured to convert the result from the at least one hardware floating-point operator back to an H=20 human-readable decimal character sequence floating-point representation in an additional 64 or fewer clock cycles.

13. The universal floating-point ISA compute engine as recited in claim **3**, wherein the at least one memory-mapped, fully pipelined, hardware floating-point operator includes a fully pipelined, floating-point Fused-Multiply-Add (FMA) operator configured to accept on its inputs, a new set of human-readable decimal character sequence floating-point representation or IEEE 754 binary floating-point format representation operands every clock cycle.

14. The universal floating-point ISA compute engine as recited in claim **13**, further comprising a hardware logic automatic conversion circuit situated in series with the FMA pipeline output, wherein the hardware logic automatic conversion circuit is configured to automatically convert the FMA operator output to human-readable decimal character sequence floating-point representations before automatically spilling into the data memory write port at the destination address specified by the single instruction.

15. The universal floating-point ISA compute engine as recited in claim **3**, wherein the universal ISA compute engine is configured as a parent universal ISA compute engine connected to at least one child universal ISA compute engine via fused instruction registers, and is further configured to:

fetch a real-time monitoring and data exchange instruction; and

execute the real-time monitoring and data exchange instruction simultaneously with the at least one child universal ISA compute engine connected to the parent universal ISA compute engine.

16. The universal floating-point ISA compute engine as recited in claim **3**, further comprising an IEEE 1149.1-1990 (JTAG) interface configured to issue opcodeless real-time monitoring and data exchange instructions to the universal ISA compute engine and to retrieve results once executed, without the use of hardware breakpoints, interrupts, or Direct Memory Access (DMA) cycles.

17. The universal floating-point ISA compute engine as recited in claim **3**, wherein logic within the compute engine is configured to implement in hardware, all computational floating-point operations mandated by the IEEE 754-2008 specification using a single instruction per operation such that a computation can be performed without having to explicitly convert the operands used in the computation from decimal character string floating-point representations beforehand.

18. The universal floating-point ISA compute engine as recited in claim **3**, wherein an exponent of the human-readable decimal character sequence floating-point representation includes a token exponent.

19. The universal floating-point ISA compute engine as recited in claim **2**, wherein the means for accepting further comprises means for accepting, in combination, a mix of the human-readable decimal character sequence floating-point representation operands and IEEE 754-2008 standard binary arithmetic format floating-point representation operands.

20. The universal floating-point ISA compute engine as recited in claim **19**, wherein the means for converting includes means for delivering to an input of a specified computing means, either an accepted unconverted binary arithmetic format floating-point representation operand or a binary format floating-point representation operand that was converted from an accepted human-readable decimal character sequence floating-point representation operand, wherein delivering is based on a state of a format Select bit in a received instruction.

* * * * *