US011635956B2

(12) **United States Patent**
Harthcock

(10) **Patent No.: US 11,635,956 B2**
(45) **Date of Patent: Apr. 25, 2023**

(54) **FULLY PIPELINED HARDWARE OPERATOR LOGIC CIRCUIT FOR CONVERTING HUMAN-READABLE DECIMAL CHARACTER SEQUENCE FLOATING-POINT REPRESENTATIONS TO IEEE 754-2008 BINARY FLOATING-POINT FORMAT REPRESENTATIONS**

(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.: **17/555,408**

(22) Filed: **Dec. 18, 2021**

(65) **Prior Publication Data**

US 2022/0113968 A1 Apr. 14, 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.

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

(51) **Int. Cl.**

G06F 9/30	(2018.01)
G06F 5/00	(2006.01)
G06F 12/06	(2006.01)
G06F 9/355	(2018.01)
G06F 9/54	(2006.01)
H03M 7/24	(2006.01)
G06F 7/483	(2006.01)

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

CPC ***G06F 9/30025*** (2013.01); ***G06F 5/00*** (2013.01); ***G06F 7/483*** (2013.01); ***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 9/30025; G06F 9/30079; G06F 9/30101; G06F 9/3557; 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 12/0292; G06F 2212/1024; G06F 2212/1048; G06F 12/0284; H03M 7/24

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,185,723	B2 *	5/2012	Norin	G06F 7/72 712/221
2007/0018860	A1 *	1/2007	Moriya	H03M 7/24 341/51
2009/0037504	A1 *	2/2009	Hussain	G06F 7/556 708/277
2012/0259904	A1 *	10/2012	Bishop	H03M 7/02 708/503

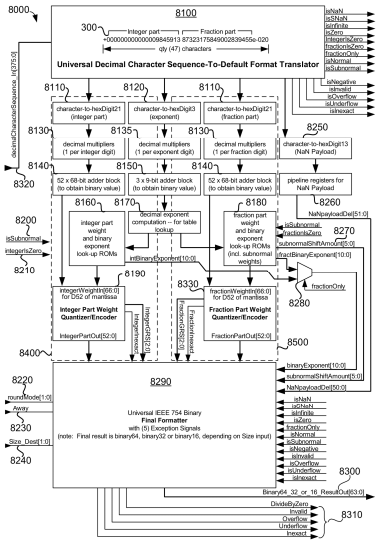
(Continued)

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

(57) **ABSTRACT**

A fully pipelined convertToBinaryFromDecimalCharacter hardware operator logic circuit configured to convert one or more human-readable decimal character sequence floating-point representations to IEEE 754-2008 binary floating-point representations every clock cycle. The circuit converts decimal character sequence floating-point representations up to 28 decimal digits in length to IEEE 754 binary64, binary32, or binary16 floating-point format representations.

8 Claims, 89 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

2014/0208068	A1*	7/2014	Wegener	H03M 7/3059 712/22
2015/0095387	A1*	4/2015	Lagneau	H03M 7/02 708/204

* cited by examiner

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.

FIGS. **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.

The invention claimed is:

1. A fully pipelined convertToBinaryFromDecimalCharacter hardware operator logic circuit configured to convert one or more human-readable decimal character sequence floating-point representations to IEEE 754-2008 binary floating-point representations every clock cycle, said hardware operator logic circuit comprising:

a hardware-implemented Decimal Character Sequence Format Translator logic front end that separates an integer part, a fraction part, and an exponent part of a character sequence and places the integer, fraction, and exponent parts into respective assigned character positions of a predetermined character sequence format and delivers an integer part character sequence to a hardware-implemented integer part mantissa logic, a fraction part character sequence to a hardware-implemented fraction part mantissa logic, and an exponent part character sequence to both a hardware-implemented integer part exponent conversion logic and a hardware-implemented fraction part exponent conversion logic;

wherein the integer part mantissa logic is configured to convert a delivered integer part character sequence into an equivalent decimal value representing an integer part mantissa and to deliver a converted decimal value representing the integer part mantissa to an integer part weight quantizer/encoder logic;

wherein the fraction part mantissa logic is configured to convert a delivered fraction part character sequence into an equivalent decimal value representing a fraction part mantissa and to deliver a converted decimal value representing the fraction part mantissa to a fraction part weight quantizer/encoder logic;

wherein the integer part exponent conversion logic is configured to convert the delivered exponent part character sequence into an equivalent decimal value representing an integer part exponent and to deliver a converted exponent decimal value representing an integer part exponent to a hardware-implemented integer part exponent look-up table/ROM and interpolation logic;

wherein the fraction part exponent conversion logic is configured to convert the delivered exponent part character sequence into an equivalent decimal value representing a fraction part exponent and to deliver a converted exponent decimal value representing a fraction

part exponent to a hardware-implemented fraction part exponent look-up table/ROM and interpolation logic;

wherein the integer part exponent look-up table/ROM and interpolation logic is configured to receive from the integer part exponent conversion logic, a decimal value representing the integer part exponent and to deliver an equivalent binary value representing the integer part exponent to a hardware-implemented selection logic;

wherein the fraction part exponent look-up table/ROM and interpolation logic is configured to receive from the fraction part exponent conversion logic, a decimal value representing the fraction part exponent and to deliver an equivalent binary value representing the fraction part exponent to the selection logic;

wherein the selection logic is configured to select the delivered equivalent binary value representing the fraction part exponent when the original human-readable decimal character sequence is fraction-only, or to select the delivered equivalent binary value representing the integer part when the original human-readable decimal character sequence is not fraction-only, wherein the selection logic then delivers a selected equivalent binary value exponent to a hardware-implemented final IEEE 754 formatter logic;

a hardware-implemented integer part greatest weight look-up table/ROM and interpolation logic configured to receive from the integer part exponent conversion logic, a decimal value representing the integer part exponent and to deliver to an integer part quantizer logic, a binary value representing an integer part greatest weight corresponding to a decimal value representing the integer part exponent;

a hardware-implemented fraction part greatest weight look-up table/ROM and interpolation logic configured to receive from the fraction part exponent conversion logic, a decimal value representing the fraction part exponent and to deliver to a fraction part quantizer logic, a binary value representing a fraction part greatest weight corresponding to a decimal value representing the fraction part exponent;

wherein the integer part quantizer logic is configured to receive from the integer part greatest weight look-up table/ROM and interpolation logic, a binary value representing the integer part greatest weight and to receive from the integer part mantissa logic, a delivered binary value representing an equivalent binary value representing the integer part mantissa and to deliver a quantized/encoded integer part value to the hardware-implemented final IEEE 754 formatter logic;

wherein the fraction part quantizer logic is configured to receive from the fraction part greatest weight look-up table/ROM and interpolation logic, a binary value representing the fraction part greatest weight and to receive from the fraction part mantissa logic, a delivered binary value representing an equivalent binary value representing the fraction part mantissa and to deliver a quantized/encoded fraction part value to the hardware-implemented final IEEE 754 formatter logic; and

wherein the hardware-implemented final IEEE 754 formatter logic is configured to accept the selected equivalent binary value exponent from the selection logic, the quantized/encoded integer part value from the integer part quantizer logic, and the quantized/encoded fraction part value from the fraction part quantizer logic and to output an IEEE 754 binary floating-point format final result.

2. The fully pipelined convertToBinaryFromDecimalCharacter hardware operator as recited in claim **1**, further comprising hardware logic enabling the hardware operator to convert human-readable decimal character sequence floating-point representations that also include a token exponent.

3. A fully pipelined convertToBinaryFromDecimalCharacter hardware operator logic circuit configured to convert a human-readable decimal character sequence floating-point representation having an integer part and a fraction part to an IEEE 754-2008 binary floating-point format representation every clock cycle, said hardware operator logic circuit comprising:

a hardware decimal character sequence format translator configured to:

receive the human-readable decimal character sequence floating-point representation up to IEEE 754-2008 "H=20" in length,

separate, in character format, the integer part and the fraction part of the human-readable decimal character sequence floating-point representation,

place the integer part in a first assigned character position of a predetermined character sequence format, and

place the fraction part in a second assigned character position of the predetermined character sequence format;

computational hardware logic connected in a fully pipelined configuration to the hardware decimal character sequence format translator, said computational hardware logic configured to simultaneously receive the integer part and the fraction part from the hardware decimal character sequence format translator and to convert the integer part and the fraction part in parallel to weighted binary values for the integer part and the fraction part; and

a hardware final formatter connected in a fully pipelined configuration to the computational hardware logic, said hardware final formatter configured to receive the weighted binary values for the integer part and the fraction part and to format the weighted binary values for a significand part of the IEEE 754-2008 binary floating-point format representation every clock cycle.

4. The fully pipelined convertToBinaryFromDecimalCharacter hardware operator logic circuit as recited in claim **3**, wherein the hardware final formatter is configured to selectively output a binary16, binary32, or binary64 IEEE 754-2008 binary floating-point format representation depending on a Size input.

5. The fully pipelined convertToBinaryFromDecimalCharacter hardware operator logic circuit as recited in claim **3**, wherein the human-readable decimal character sequence floating-point representation also includes an exponent part, and the hardware decimal character sequence format translator is further configured to separate the exponent part and place the exponent part in a third assigned character position of the predetermined character sequence format.

6. The fully pipelined convertToBinaryFromDecimalCharacter hardware operator logic circuit as recited in claim **5**, wherein the computational hardware logic is further configured to:

convert the exponent part to an exponent part binary value in parallel with the integer part and the fraction part;

adjust the exponent part binary value for use as an index; and

utilize the index to look up weights for the integer part and the fraction part.

7. The fully pipelined convertToBinaryFromDecimal-Character hardware operator logic circuit as recited in claim 3, wherein the human-readable decimal character sequence floating-point representation has no explicit exponent part, and the hardware decimal character sequence format translator is further configured to create a character sequence exponent for the human-readable decimal character sequence floating-point representation and place the character sequence exponent in a third assigned character position of the predetermined character sequence format.

8. The fully pipelined convertToBinaryFromDecimal-Character hardware operator logic circuit as recited in claim 3, wherein the human-readable decimal character sequence floating-point representation also includes a token exponent part, and the hardware universal decimal character sequence format translator is further configured to create a character sequence exponent for the human-readable decimal character sequence floating-point representation and place the character sequence exponent in a third assigned character position of the predetermined character sequence format.

* * * * *