CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Electric Drives and Traction

Name of the report

Technical report

Petr Zakopal

Prague 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

This is the introduction.

2 Calculating the division of fixed point numbers

Usually, when using numerical methods to solve the transcendetal equations, there is a need to calculate the division of two input numbers. Even for solving one set of two equations with Newton Raphson (NR) method, the calculation of reciprocal value of the Jacobian determinant is needed.

There are available some IP blocks, which are capable of calculating the division of two numbers, but the blocks are usually vendor specific intellectual property IP [1] or feature low performance [2].

The negative side of vendor specific IP is, that it is hard to use them with any other FPGA chip than the vendor specific. On the other hand the vendor specific IP is usually optimized to use the specific type of resources available at the vendor's chip whci resolves in better performance.

To preserve the compatibility of the design with multiple vendors, the custom solution for division design based on the very known Newton Rapshon (NR) algorithm was developed. [2]

2.1 Newton Rapshon algorithm for calculating the division

General Newton Raphson (NR) algorithm is a well known way how to solve equations the numerical way. It is the reason why it is utilized in many algorithms. However, the negative aspect of NR is that it's convergency strongly depends on initial values of unknown variables. When the initial variables are chosen poorly, the performed number of iterations before the convergency is reached can be high.

To reach the fastest convergency possible (determined in number of iterations) apart from the scaling the dominator into the interval [0.5,1] the initial value calculation formula should be utilized. [2] The initial value formula 2 - 1 is applied after the scaling of denominator is performed. The algorithm developed for the appropriate scaling is explained in the *Calculating number of bits to shift the denominator*.

$$x_0 = \frac{48}{17} - \frac{32}{17}D, \tag{2 - 1}$$

where the x_0 is the initial value for NR algorithm and D is the denominator value for calculating the expression N/D.

Because of the fixed point number format *Q32.15* is used, the fractional numbers in equation 2 - 1 are rounded to 2.8229 (32'b00000000000000010_110100101011000 in binary)
and 1.8819 (32'b00000000000000001_111000011100101 in binary) respectively.

After the initial value x_0 is calculated, the NR algorithm is performed. The idea for using NR algorithm to calculate the division of N/D is to trade the division for a multiplication, which can be synthetized in the FPGA fabric. For the NR algorithm the function which root is $1/D$ is crucial. There may be many functions, which root is the searched value $1/D$ but the most trivial is eq. 2 - 2.

$$\mathrm{F}(x) = \frac{1}{x} - D. \tag{2 - 2}$$

2.2 IP block design

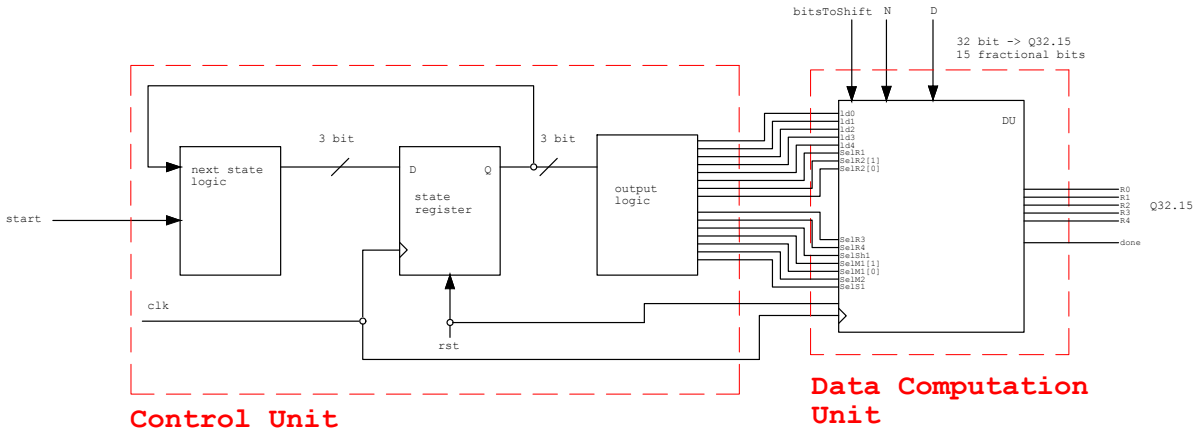

Figure 2 - 1 Top module design for the IP block design.

2.2.1 Data Path Unit

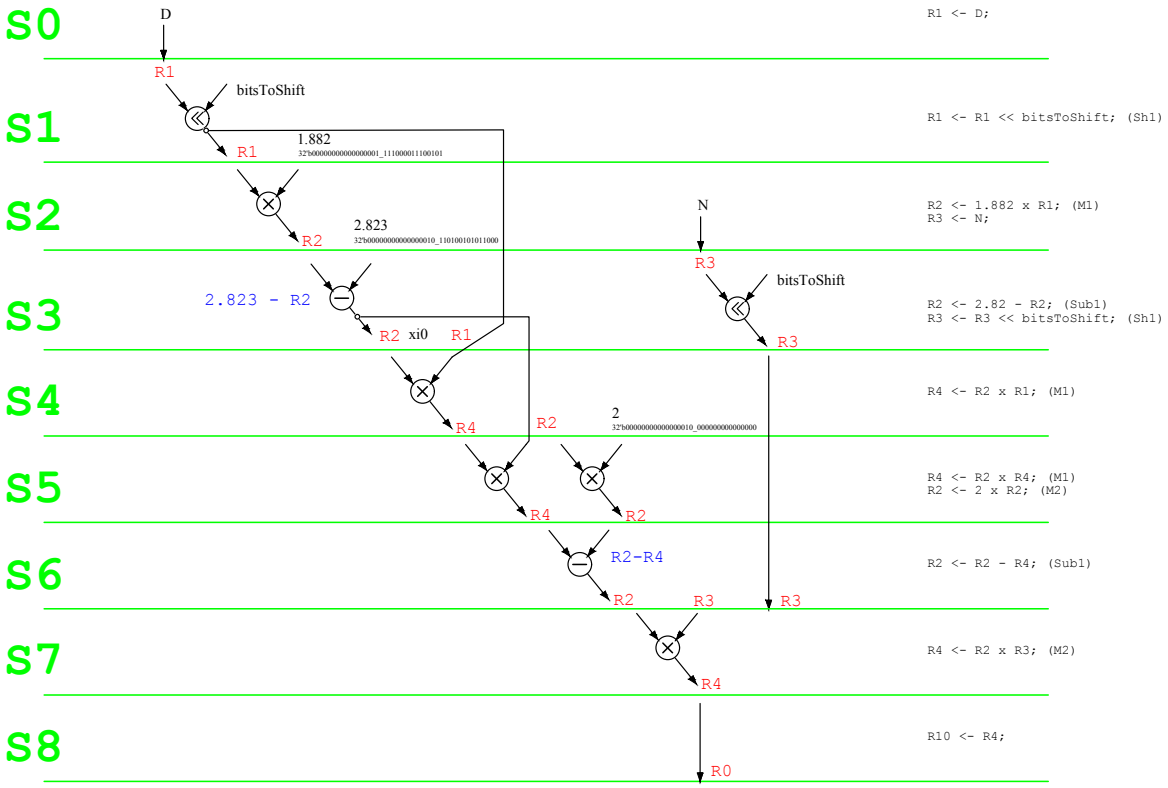

Figure 2 - 2 Alloccation and timing diagram for the Data Path Unit part of the IP.

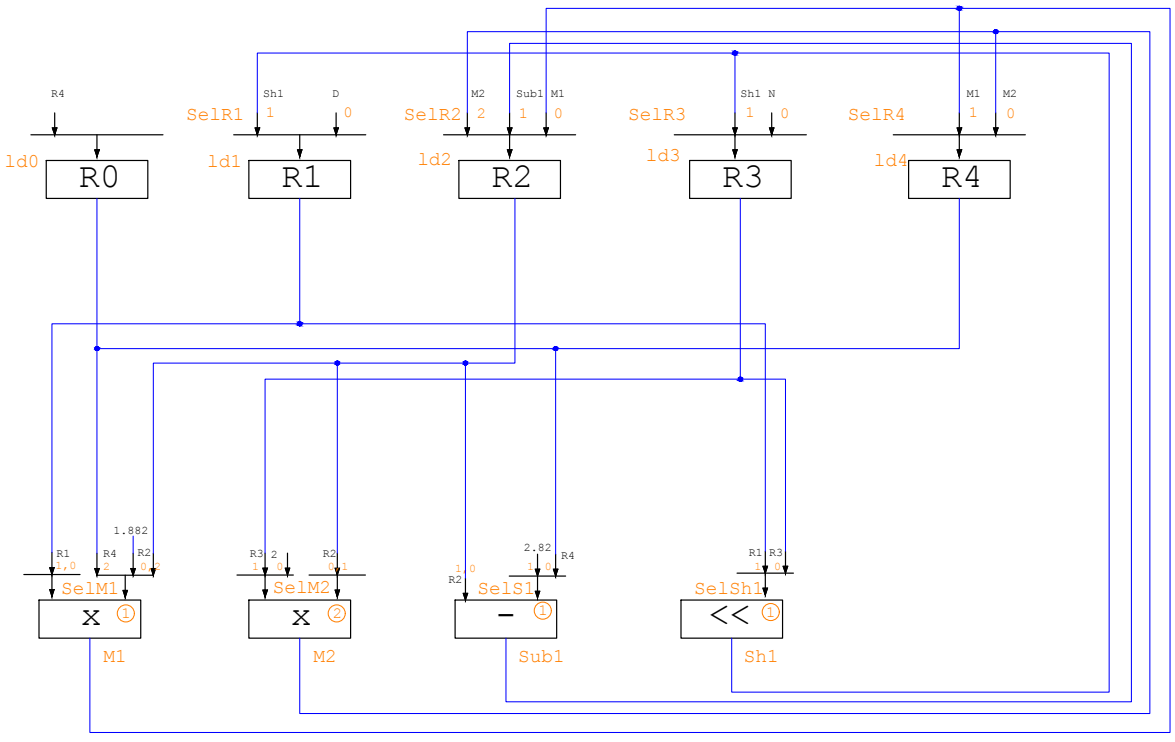

Figure 2 - 3 Register transfer level RTL scheme of the IP Data Path Unit part of the IP.

2.2.2 Control Unit

The encoded operations in CV signal bits are depicted in the table 2 - 1.

Table 2 - 1 Control signal encoding table for instructions to be processed by the Division Module.

State	RTL Code	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CV
		ld0	ld1	ld2	ld3	ld4	SelR1	SelR2[1]	SelR2[0]	SelR3	SelR4	SelSh1	SelM1[1]	SelM1[0]	SelM2	SelS1	
S0	R1 ← D;	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000h
S1	R1 ← R1 « 32; (Sh1)	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	2210h
S2	R2 ← 1.882 x R1; (M1) R3 ← N;	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1804h
S3	R2 ← 2.82 - R2; (Sub1) R3 ← R3 « 32; (Sh1)	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	18C0h
S4	R4 ← R2 x R1; (M1)	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	420h
S5	R4 ← R2 x R4; (M1) R2 ← 2 x R2; (M2)	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	1528h
S6	R2 ← R2 - R4; (S1)	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1081h
S7	R4 ← R2 x R3; (M2)	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	402h
S8	R0 ← R4;	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000h

2.3 Calculating number of bits to shift the denominator

As presented in the section *Newton Rapshon algorithm for calculating the division* the denominator must be appropriately scaled for the division algorithm to work. This section presents algorithm for scaling the denominator specified in the fixed point number format *Q32.15*. After the scaling value is successfully determined, the numerator is scaled accordingly.

The presented algorithm shifts the value of denominator at every positive edge of the clock signal and saves the shifted value in the `compare` register. Then the combinational circuit is utilized to compare the shifted value in `compare` register with the number `1` specified in *Q32.15* format. If the compared value is the same or lower than `1` the shifting algorithm is done and the value `scaleToShift` is successfully found. If not, the inner value of shifting bits is incremented and the algorithm proceeds to the next iteration.

The presented algorithm is packed in the *denominatorSizeScaleUnit* module and it's pseudocode is depicted in the code 2 - 1.

```
at every negative edge of clock or positive edge of reset
if(rst)
  scaleToShift = 0;
  scaleToShiftInternal = 1;
  started = 0;
end if
else if (start)
started = 1;
end else if

at every positive edge of clock
if (compare <= 32'b00000000000000001_000000000000000)
done = 1;
started = 0;
scaleToShift = scaleToShiftInternal;
end if
else
done = 0;
scaleToShiftInternal = scaleToShiftInternal + 1;
```

```
20  end else
```

Code 2 - 1 Pseudocode for the denominatorSizeScaleUnit module algorithm.

Conclusion

And this is the conclusion of my report.

References

[1] ADVANCED MICRO DEVICES, Inc. Divider Generator LogiCORE™ IP. In: *Intellectual Property* [online]. [B.r.] [visited on 2023-10-01]. Available from: https://www.xilinx.com/products/intellectual-property/divider.html.

[2] BURKE, Tom. Verilog Fixed point math library. In: *GitHub* [online]. [B.r.] [visited on 2023-10-01]. Available from: https://github.com/freecores/verilog_fixed_point_math_library.

[3] BURENEVA, Olga I.; KAIDANOVICH, Olga U. FPGA-based Hardware Implementation of Fixed-point Division using Newton-Raphson Method. In: *2023 IV International Conference on Neural Networks and Neurotechnologies (NeuroNT)*. 2023, pp. 45–47. Available from DOI: 10.1109/NeuroNT58640.2023.10175844.

[4] DIGILENT, Inc. Zybo. In: *Digilent Documentation* [online]. [B.r.] [visited on 2022-11-11]. Available from: https://digilent.com/reference/programmable-logic/zybo/start.

[5] DIGILENT, Inc. Zybo Z7 Migration Guide. In: *Digilent Documentation* [online]. [B.r.] [visited on 2022-11-11]. Available from: https://digilent.com/reference/programmable-logic/zybo-z7/migration-guide.

[6] XILINX, Inc. SoCs with Hardware and Software Programmability. In: *Xilinx Website* [online]. [B.r.] [visited on 2022-11-11]. Available from: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

[7] XILINX, Inc. Zynq-7000 SoC Technical Reference Manual. In: *Xilinx Documentation* [online]. 02. 04. 2021 [visited on 2022-11-11]. Available from: https://docs.xilinx.com/v/u/en-US/ug585-Zynq-7000-TRM.

[8] DIGILENT, Inc. Zybo Reference Manual. In: *Digilent Documentation* [online]. [B.r.] [visited on 2022-11-11]. Available from: https://digilent.com/reference/programmable-logic/zybo/reference-manual.

[9] XILINX, Inc. Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973). In: *AMD Xilinx Documentation Portal* [online]. [B.r.] [visited on 2022-11-18]. Available from: https://docs.xilinx.com/r/en-US/ug973-vivado-release-notes-install-license/.

[10] XILINX, Inc. PetaLinux Tools Documentation: Reference Guide (UG1144). In: *AMD Xilinx Documentation Portal* [online]. [B.r.] [visited on 2022-11-18]. Available from: https://docs.xilinx.com/r/en-US/ug1144-petalinux-tools-reference-guide.

[11] XILINX, Inc. Downloads. In: *AMD Xilinx Downloads* [online]. [B.r.] [visited on 2022-11-19]. Available from: https://www.xilinx.com/support/download.html.

[12] XILINX, Inc. Downloads. In: *AMD Xilinx PetaLinux Tools* [online]. [B.r.] [visited on 2022-11-19]. Available from: https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html.

[13] INC., Xilinx. Zynq-7000 SoC Technical Reference Manual (UG585). In: *Xilinx Documentation Portal* [online]. 02. 04. 2021 [visited on 2023-02-28]. Available from: https://docs.xilinx.com/v/u/en-US/ug585-Zynq-7000-TRM.

[14] XILINX, Inc. System-on-Modules (SOMs): How and Why to Use Them. In: *Xilinx Website* [online]. [B.r.] [visited on 2023-03-10]. Available from: https://www.xilinx.com/products/som/what-is-a-som.html.

[15] XILINX, Inc. Kria KR260 Robotics Starter Kit. In: *Xilinx Website* [online]. [B.r.] [visited on 2023-03-10]. Available from: https://www.xilinx.com/products/som/kria/kr260-robotics-starter-kit.html.

[16] XILINX, Inc. Kria SOM Carrier Card Design Guide (UG1091). In: *AMD Xilinx Documentation Portal* [online]. 27. 07. 2022 [visited on 2023-03-18]. Available from: https://docs.xilinx.com/r/en-US/ug1091-carrier-card-design/Introduction.

[17] XILINX, Inc. Kria K26 SOM Data Sheet (DS987). In: *AMD Xilinx Documentation Portal* [online]. 26. 07. 2022 [visited on 2023-03-18]. Available from: https://docs.xilinx.com/r/en-US/ds987-k26-som.

[18] XILINX, Inc. Kria KR260 Robotics Starter Kit User Guide (UG1092). In: *AMD Xilinx Documentation Portal* [online]. 17. 05. 2022 [visited on 2023-04-05]. Available from: https://docs.xilinx.com/r/en-US/ug1092-kr260-starter-kit/Interfaces.

[19] XILINX, Inc. Kria K26 System-on-Module. In: *AMD Xilinx Product Brief* [online]. [B.r.] [visited on 2023-04-05]. Available from: https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/xilinx-k26-product-brief.pdf.

[20] XILINX, Inc. XTP743 - Kria KR260 Starter Kit Carrier Card Schematics (v1.0). In: *AMD Xilinx Board Files* [online]. 09. 06. 2022 [visited on 2023-04-06]. Available from: https://www.xilinx.com/member/forms/download/design-license.html?cid=bad0ada6-9a32-427e-a793-c68fed567427&filename=xtp743-kr260-schematic.zip.

[21] XILINX, Inc. XTP685 - Kria K26 SOM XDC File (v1.0). In: *AMD Xilinx Board Files* [online]. 14. 05. 2021 [visited on 2023-04-06]. Available from: https://www.xilinx.com/member/forms/download/design-license.html?cid=29e0261a-9532-4a47-bb06-38c83bbbb8c0&filename=xtp685-kria-k26-som-xdc.zip.

[22] ADMIN, Confluence Wiki; ROY, Debraj; DYLAN. Embedded SW Support. In: *Xilinx Wiki* [online]. 28. 02. 2023 [visited on 2023-04-06]. Available from: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841631/Embedded+SW+Support.

[23] FOUNDATION, Linux. Real-Time Linux. In: *Linux Foundation DokuWiki* [online]. [B.r.] [visited on 2023-04-06]. Available from: https://wiki.linuxfoundation.org/realtime/start.

[24] XILINX, Inc. Zynq UltraScale+ MPSoC Processing System Product Guide (PG201). In: *Xilinx Documentation* [online]. 11. 05. 2021 [visited on 2023-04-13]. Available from: https://docs.xilinx.com/r/en-US/pg201-zynq-ultrascale-plus-processing-system/Fabric-Reset-Enable.

[25] ZAKOPAL, Petr et al. [Kria SOM KR260 Starter Kit] Schematic (pdf) vs constrains (xdc) pin confusion. Possible explanation on fan pinout. In: *Xilinx Support Community Forum*. 18. 03. 2023. Available also from: https://support.xilinx.com/s/question/0D54U00006alUwcSAE/kria-som-kr260-starter-kit-schematic-pdf-vs-constrains-xdc-pin-confusion-possible-explanation-on-fan-pinout?language=en_US.

Appendix A: List of symbols and abbreviations

A.1 List of abbreviations

FPGA	Field Programmable Gate Array
IP	Intellectual property
NR	Newton Raphson
RTL	Register Transfer Level