Design of Quadruple Precision Multiplier Architectures with SIMD Single and Double Precision support

Manish Kumar Jaiswal[1,*], Hayden K.-H. So[2]

Abstract

This paper proposes architectures for dual-mode and tri-mode dynamically configurable multiplier for quadruple precision arithmetic. The proposed dual-mode QPdDP multiplier architectures can either compute on a pair of quadruple precision (QP) operands or provide SIMD support for two-parallel (dual) sets of double precision (DP) operands. The proposed tri-mode QPdDPqSP multiplier architectures are aimed to include the four-parallel (quad) single precision (SP) along with dual-DP and a QP operand processing. For the underlying largest sub-component, the mantissa multiplier, two methods are analyzed to design the dual-mode/tri-mode architectures. One is based on the Karatsuba method, and in another a dual-mode/tri-mode Radix-4 Modified Booth (MB) multiplier is proposed. The proposed dual-mode/tri-mode MB multiplier requires few extra 2:1 MUXs as an overhead compared to a simple MB multiplier. To support dual-mode/tri-mode functioning other important sub-components of the FP multiplication are also re-designed for multi-mode support. The proposed architectures are synthesized using UMC 90nm ASIC technology, and are compared against prior literature in terms of area, period, and a unified metric "$Area(Gate\ Count) \times Period(FO4) \times Latency \times Throughput(in\ cycles)$". The dual-mode/tri-mode FP architectures with MB mantissa multipliers shows better timings, however, those with Karatsuba mantissa multipliers acquires smaller area.

Keywords: Floating Point Arithmetic, Multiplier, SIMD, Modified Booth Multiplier, Karatsuba Multiplication, Multi-mode Arithmetic, ASIC

*Corresponding Author

Email addresses: manishkj@hku.hk, manishkj25@gmail.com (Manish Kumar Jaiswal), hso@eee.hku.hk (Hayden K.-H. So)

[1]Manish Kumar Jaiswal is with The University of Hong Kong, Hong Kong.

[2]Hayden K.-H. So is with The University of Hong Kong, Hong Kong.

1. Introduction

Many scientific and engineering applications are in demand of higher precision computation than the single precision (SP) and double precision (DP) computation [1, 2], which leads to the inclusion of quadruple precision (QP) arithmetic in the application's processing. Thus, the quadruple precision format becomes an integral part of the IEEE-754 floating point standard 2008 [3].

However, the hardware implementation of quadruple precision arithmetic requires a large amount of area. This can be compensated by the idea of including a dual (two-parallel) double precision support in it, albeit with little extra resources. Moreover, a possible inclusion of quad (four parallel) single precision support would further make it more promising. The contemporary computing systems (Cell-BE [4], ARM FPU [5], Nvidia GPU [6], etc) achieve high performance requirements by incorporating vector-arrays for these arithmetic units, which include separate vector-arrays for single precision (SP) processing and double precision (DP) processing units. These separate vector-arrays need a large silicon area. Also, these computing machines do not include any hardware support for quadruple precision arithmetic, and the software solutions for quadruple precision arithmetic are very slow.

In this view, this manuscript is exploring possible architectures for dual-mode QPdDP (quadruple precision with dual (two parallel) double precision) and tri-mode QPdDPqSP (quadruple precision with dual double precision and quad (four parallel) single precision) arithmetic. These have two prime targets, first to provide quadruple precision arithmetic support and second is aimed towards the inclusion of SIMD (single instruction multiple data) processing support for double precision and single precision arithmetic in it. These idea helps in saving a large silicon area compared to the individual units of a quadruple precision, and multiple units of double precision and single precision arithmetic units. Currently, this paper is focused on the architectures of QPdDP and QPdDPqSP multiplier arithmetic architecture.

Some recent literature have addressed on multi-mode architectures [7, 8, 9] for various arithmetic. However, literature contains a limited work on dual-mode QPdDP multiplier architectures. The multiplier architectures available in [10, 11, 12] are iterative in nature, while [13] has presented unfolded architectures. Few prior work also addressed fused multiply-add architectures [14, 15](unfolded architecture), [16] (iterative architecture) with multi-mode mantissa multiplier architectures similar to [10, 11]. Only available work on QPdDPqSP multiplier [17] also requires a huge amount of area with poor timings. All the prior arts

are using the rectangular/array multiplier technique for dual-mode/tri-mode mantissa multiplication which requires a large amount of area, and further they are without sub-normal and exceptional handing support. Very little literature have also focused on related adder arithmetic architectures [18, 19, 20] and divider architecture [21].

The proposed work is an extension of our prior work [22] which has presented a double precision FP multiplier architecture with dual single precision multiplication support. It had proposed a dual-mode double precision Modified Radix-4 Booth mantissa (54x54) multiplier around which the complete FP multiplier design has been built. However, the current manuscript proposes dual-mode QPdDP as well as tri-mode QPdDPqSP quadruple precision multiplier architectures. Moreover, the current manuscript incorporates two methods, Radix-4 Booth method and Karatsuba method, for building dual-mode and tri-mode quadruple precision mantissa multipliers, which provide an interesting area-speed trade-offs among them. All other essential sub-components of FP multiplier flow (dual/tri-mode: LOD, Dynamic Left/Right Shifter, Rounding, sub-normal and exceptional handling, etc) are also constructed for dual-mode QPdDP and tri-mode QPdDPqSP multiplier architectures.

Prior literature have explored extensively on the rectangular/array multiplier for multi-mode multiplier architectures. Thus, current exploration with Modified Booth and Karatsuba methodology for multi-mode multiplier architectures will provide a complimentary research on this subject, along with highlighting several design trade-offs among them. Modified-Booth multiplier is a traditional multiplication method for most of the implementation due to its smaller delay and, Karatsuba method, based on the divide-and-conquer paradigm, helps to reduce hardware cost in large size multiplier (for example QP mantissa size), however, with an extra delay cost [23].

The proposed QPdDP FP multiplier architectures explore the dual mode mantissa multiplication architecture by designing a dual mode mantissa multiplier (a 113x113 with dual 53x53 support) using Modified Radix-4 Booth multiplier (named as QPdDP_MB) and by using Karatsuba method [23, 24, 25] (which is named as QPdDP_KM). Similarly, tri-mode QPdDPqSP_MB and QPdDPqSP_KM multiplier architectures are also proposed and designed. Previously, the Karatsuba method is explored for single mode (SP, DP, QP) FP multiplier architectures [24, 25] and as dual-mode double precision architecture for divider and multiplier units [26, 27]. Here, it is here explored for the multi-mode functionality in QPdDP_KM and QPdDPqSP_KM multiplier architecture. The idea for multi-

mode multiplier architecture using Modified Booth technique, which can be utilized for QP multiplier architecture is lacking, and this motivates us to investigate in this direction to design QPdDP_MB and QPdDPqSP_MB architecture. The proposed architectures are also supported with subnormal & exceptional case handling and processing. They are synthesized using *UMC 90nm* standard cell based ASIC library, and compared with the prior arts.

The main contribution of this work can be summarized as follows:

- Proposed a dual mode QPdDP multiplier architecture, which supports QP with dual (two parallel) DP processing.
- Extended above architecture for a tri mode QPdDPqSP multiplier architecture, which further includes support for a four parallel SIMD SP processing.
- Two methods are explored for the architectures dual/tri mode mantissa multiplication, Karatsuba method and currently proposed dual/tri mode Radix-4 Modified Booth method. All other sub-components are also redesigned for dual/tri mode functioning.
- Proposed dual/tri mode multiplier architectures significantly out-performs the prior arts in terms of design metrics.

This manuscript organization proceeds as follows. Section 2 discusses the proposed dual mode QPdDP multiplier architecture and Section 3 describes the proposed tri-mode QPdDPqSP multiplier architecture. The detail implementation results and related comparisons with previous literature work are presented in the section 4. Finally, the manuscript is concluded in section 5.

2. Proposed Dual-Mode QPdDP FP Multiplier Architecture

The proposed multiplier architecture follows the state-of-the-art computational flow for single mode F.P. multiplication which supports the processing of subnormal and exceptional. To support the dual-mode processing, each individual stage of the flow are re-constructed with efficient resource sharing and tuned datapath to minimize the multiplexing circuitry. The architectures of proposed dual-mode multipliers are designed with four pipeline stages and is shown in Fig 1. Except the different dual-mode mantissa multiplier (KM or MB), all other components are same in the both dual-mode architectures (QPdDP_KM and QPdDP_MB).

The input operands for the QPdDP architecture either contains a set of QP operands or two sets of DP operands, as shown in Fig 2. Each component of the dual-mode QPdDP multiplier architecture is discussed below in details.

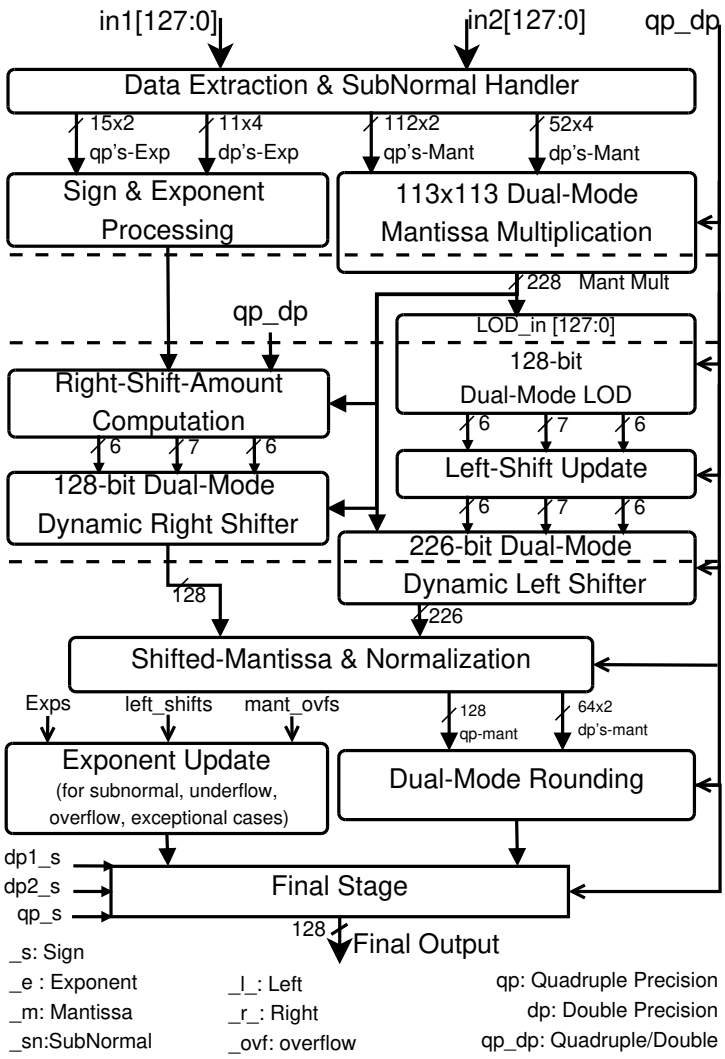

Figure 1: QPdDP F.P. Multiplier Architecture (Dotted lines refer to pipeline registers)

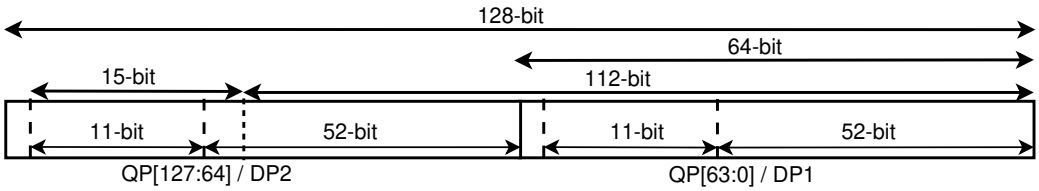

Figure 2: Input / Output Register Format.

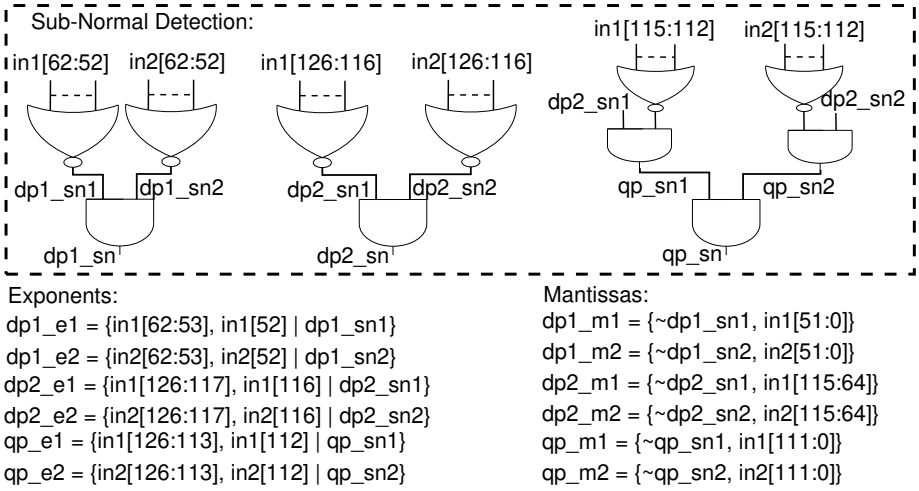

Figure 3: QPdDP Data Extraction and Subnormal Handler.

2.1. Data Extraction and Exceptional Check-up

This unit extract the signs (*_s*), exponents (*_e*) and mantissas (*_m*) for QP and both DPs from primary operands ($in1$, $in2$) based on the standard floating point format definition. Further, it undergoes for sub-normal and exceptional case (infinity, NaN, zero) checking, and accordingly update the mantissa and exponent (Fig 3). As appeared in Fig. 2 that the exponent portion of QP and second DP (DP-2) operands are overlapped as below:

$$\overbrace{\underbrace{xxxxxxxxxxx}_{DP-2\ Exponent}\ xxxx}^{QP\ Exponent}$$

Thus, the sub-normal and exceptional checks are shared among QP and second DP (DP-2) operands due to their shared bit position. Similarly, the resource sharing is implemented for other exceptional cases (infinity, NaN and Zero).

2.2. Dual-Mode Core Arithmetic: Sign, Exponent and Mantissa Processing

The sign computation (logical `xor` operation between operands sign-bit) and exponent processing ($e \leftarrow e1 + e2 - BIAS$) is trivial in nature. Here, the exponents sum computation are shared among QP and DP-2 operands to minimize the resource utilization.

The mantissa multiplier unit cost for most of the area, and here investigated under Karatsuba method and Modified Booth (MB) method. Prior literature on

dual-mode mantissa multiplication architectures are based on rectangular multiplier technique. In rectangular multiplication technique a large multiplier block is partitioned in to smaller multiplier blocks, which further combined using adders. Under this for a dual-mode operation, 2 block partitioning is required which proceeds as follows. Let A and B are 2N bits operands which is partitioned in half at N bits, then their product can be seen as

$$\begin{aligned} A \cdot B \quad & \rightarrow \quad (A_{2N} \rightarrow \{A_{H_N}, A_{L_N}\}, B_{2N} \rightarrow \{B_{H_N}, B_{L_N}\}) \\ & = \{A_{H_N} \cdot B_{H_N},\ A_{L_N} \cdot B_{L_N}\} + \{A_{H_N} \cdot B_{L_N} + A_{L_N} \cdot B_{H_N},\ N'b0\} \end{aligned} \tag{1}$$

Thus, it requires 4 NxN blocks for complete 2Nx2N multiplier. This method is used in prior literature in iterative manner using two 57x57 blocks to compute full 113x113 in two clock cycles and two parallel 53x53 in one clock cycle. Further, mostly tree/array multiplication method were used for 57x57 blocks. Dual-mode multiplier architecture based on this idea results in poor timing and throughput metric, along with poor $Area \times Period \times Latency \times Throughput(in\ cycles)$ metric.

To improve upon prior arts, we studied two more widely used strategies, Karatsuba Method and proposes a dual-mode Modified Booth Multiplier. Both are discussed below separately.

2.2.1. Dual-Mode Mantissa Multiplier using Karatsuba Method

Karatsuba methodology of multiplication reduces the number of multiplier blocks using partitioning method, as discussed in (1) with two partition. Using Karatsuba method with two partitioning, it requires only 3 blocks instead of 4 blocks. It is shown below in (2). However, it requires few extra adder/subtractor. Thus, it helps in reducing area compare to block rectangular method.

$$\begin{aligned} A \cdot B \quad & \rightarrow \quad (A_{2N} \rightarrow \{A_{H_N}, A_{L_N}\}, B_{2N} \rightarrow \{B_{H_N}, B_{L_N}\}) \\ & = \{\alpha, \beta\} + \{(A_{H_N} + A_{L_N}) \cdot (B_{H_N} + B_{L_N}) - \alpha - \beta, N'b0\} \\ & \text{where, } \alpha = A_{H_N} \cdot B_{H_N}, \quad \beta = A_{L_N} \cdot B_{L_N} \end{aligned} \tag{2}$$

In (2), it requires two NxN and one (N+1)*(N+1) multiplier block. For the requirement of implementing dual-mode 113x113 multiplier using Karatsuba method, it requires one 56x56, one 57x57 and one 58x58 multiplier blocks. These multiplier blocks of 56x56, 57x57 and 58x58 are designed using Radix-4 modified Booth multiplier, with Dadda reduction tree [28], followed by the Kogge-Stone [29] final adder. Kogge-Stone adder is a parallel prefix fast adder which is used as the final adder, however, other parallel prefix adders can also be incorporated based

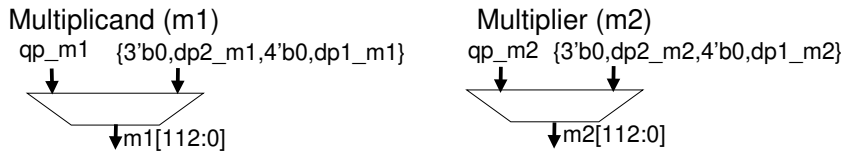

Figure 4: Dual-Mode Karatsuba Multiplier (KM) Operands Multiplexing

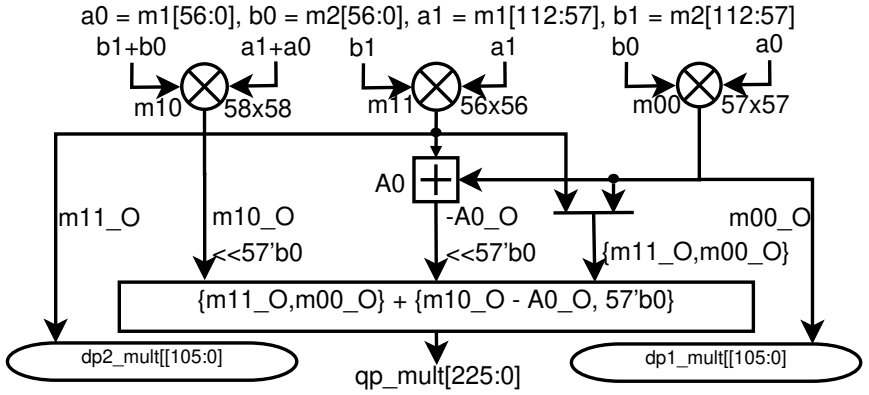

Figure 5: Dual-Mode Karatsuba Multiplier

on area/timing requirement. A pipeline register is placed between Dadda tree and Kogge-Stone adder to facilitate the pipelining of mantissa multiplier to achieve the critical path of the architecture.

The input operands generation and complete architecture of dual mode 113x113 Karatsuba mantissa multiplier is shown in Fig. 4 and Fig. 5, respectively. The multiplicand and multiplier operands are generated using QP and both DPs mantissa operands in such a way that it contains only QP operands in QP mode, and two sets of DP operands in dual-DP mode. In QP mode of operation, the complete unit produce the mantissa multiplication for QP, however, in dual-DP mode multipliers $m00$ and $m11$ (in Fig. 5) process mantissa multiplication for DP1 and DP2, respectively.

The above architecture is also studied with multi-level Karatsuba partitioning (ie. further partitioning of 56x56/57x57/58x58 multiplier blocks), however, it degrades the timing metric further.

2.2.2. Dual-Mode Radix-4 Modified Booth Mantissa Multiplier

This paper proposes an architecture for 113x113 Radix-4 Modified Booth multiplier which also supports dual (two-parallel) 53x53 multiplication. For this, as

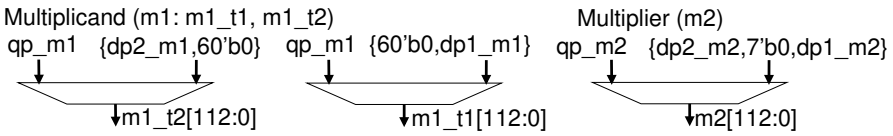

Figure 6: Dual-Mode Modified Booth (MB) Multiplier Operands Multiplexing

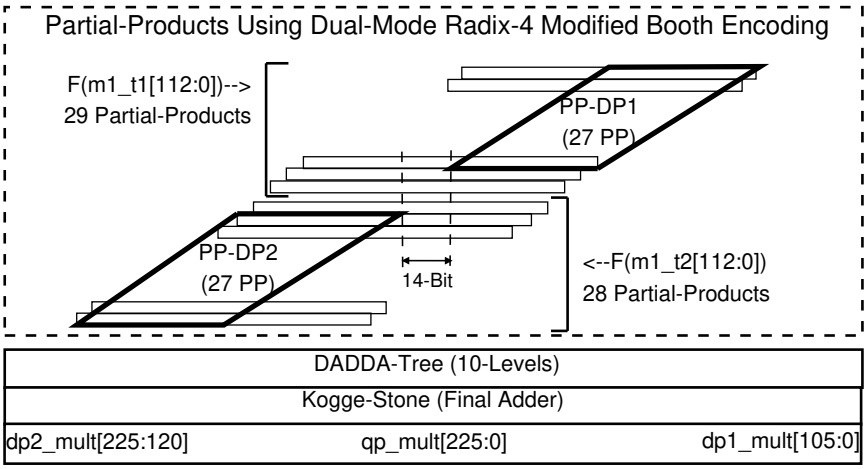

Figure 7: Dual-Mode Modified Booth (MB) Multiplier

shown in Fig. 6, two sets of multiplicand operands (m1_t1, and m1_t2), and a unified multiplier operand (m2) are generated. Based on these operands, as shown in the architecture Fig. 7, out of total 57 partial products in 113x113 MB multiplier, the first 29 partial products are generated using multiplicand m1_t1, which produces partial products either for QP or DP-1, and the remaining 28 partial products are generated using the multiplicand m1_t2, which produces it for QP or DP-2. Based on the structure of multiplier operand (m2) the MB partial product encoder either work for QP or dual DP.

In QP mode, the Booth encoding processing (for quadruple precision) is easy to visualize as both of m1_t1 and m1_t2 contain same operand, the quadruple multiplicand operand, and m2 contains the quadruple multiplier operand. During dual DP mode, Booth encoding of m2 can be seen from Fig. 8. The first 27 partial products which are encoded using DP-1 multiplier operand process the m1_t1 multiplicand, however, last 27 partial products are encoded using DP-2 multiplier operand which process the m1_t2 multiplicand. During DP mode, 60-bit MSB of first 27 partial products are zero (based on the format of m1_t1), while, the LSB 60-bit of last 27 partial products are zero (based on the format of m1_t2), however,

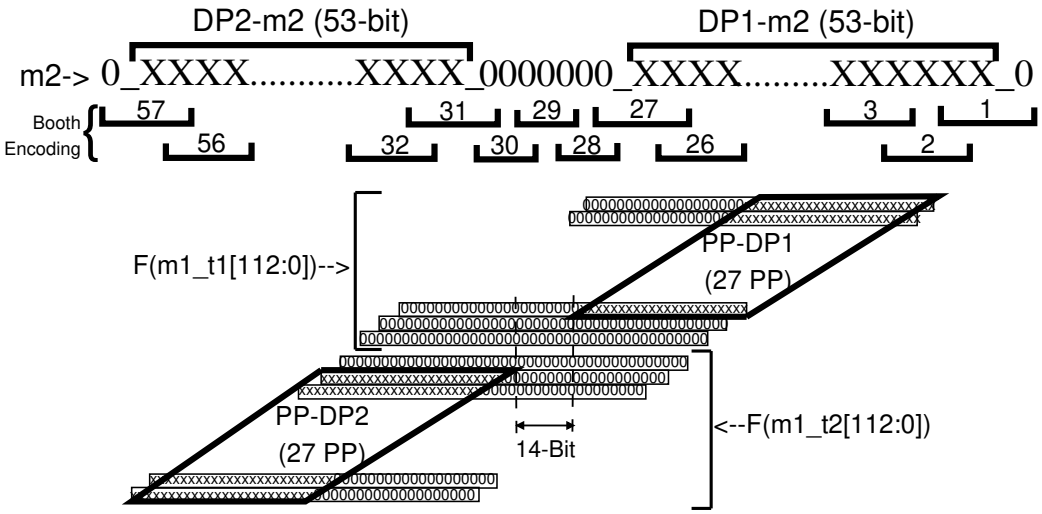

Figure 8: QPdDP Booth Encoding Scenario under Dual Double Precision Processing Mode

three partial products 28-30 are zero as their Booth encoding values are '000'. The last partial product of DP-1 would be separated by first partial product of DP-2 by 14-bit and thus avoid any carry forward during partial product reduction and final addition processing.

These partial product are further reduced by a 10 level Dadda tree, in which a register layer is inserted after level-7 for pipelining. The final addition is performed by Kogge-Stone adder.

The mantissa multiplication result contains either QP mantissa multiplication result or two DP mantissa multiplication results. Compared to a contemporary MB multiplier, the proposed dual-mode MB multiplier requires only 3, 113-bit 2:1 MUXs as a hardware overhead which are used for initial operands generation (m1_t1, m1_t2 and m2). Here, it is worth to mention that, since mantissa operands are always unsigned number (only positive operands) therefore the above architecture does not face the issue of sign extension (which is required for negative operands multiplication). In fact, this underlying observation only helped us to design the multi-mode Booth multiplier architecture (for QPdDP and QPdDPqSP) just by arranging its input operands combinations using few extra multiplexers while keeping rest of the architecture similar to contemporary Booth multiplier.

All the above processing, except a portion of mantissa multiplication (KM or MB) are part of `first pipeline stage` of the architecture. The `second stage` of the architecture includes part of mantissa multiplier (as discussed above for

both cases), and the generation of LOD_in, the input for Leading-One-Detector.

2.3. Sub-Normal Processing

The `third stage` of architecture mainly includes the components related to the sub-normal processing, as shown in Algorithm-1. It includes Leading-One-Detector (LOD), Left-Shift-update, part of Dynamic-Left-Shifter, Right-Shift-Amount computation, and Dynamic-Right-Shifter.

Algorithm 1 Sub-Normal Processing

SUB-NORMAL Processing
  Right-Shift Amount:
    Check if E is negative? (For Right Shifting)
    Right_Shift ← BIAS - (E1 + E2) + $\overline{(\text{Mult\_M[MSB]}\&\overline{|\text{E}})}$
  Left-Shift Amount:
    Left_Shift ← Leading-One-Detection (LOD) of M
    Left_Shift (Update) ← Check if less than or equal to E?
  Dynamic Right Shifting and Dynamic Left Shifting:
    Mult_M ← Mult_M >> Right_Shift
    Mult_M ← Mult_M << Left_Shift

Leading-One-Detector is required to compute the amount of left-shift for mantissa multiplication result in case the input operand is sub-normal, and the result can be normalized. A dual-mode 128-bit LOD with hierarchical modeling is shown in Fig 9. It also shown the generation of an unified input for LOD, LOD_in, which contains 128-bit mantissa multiplication results either for QP or for both DPs. LOD_2:1 serves as a basic building block for LOD design, and combination of its two units forms a LOD_4:2. Likewise, two LOD_4:2 forms a LOD_8:3, and similarly up to LOD_64:6 and then LOD_128:7 is built. The dual-mode LOD can process either a 128-bit input or two 64-bit inputs and produce corresponding left-shift amounts respectively. Compared to a single mode LOD, it requires a MUX for generating LOD_in as an hardware overhead. The Right-Shift-Amount processing and Left-Shift update processing are accomplished using trivial methods as shown in Algorithm-1.

Dynamic (left/right) shifter is designed using two parallel set of barrel shifter. The architecture for Dual-Mode Dynamic Left Shifter is shown in Fig 10, which can process either a 226-bit input or two parallel 113-bit input operands. It contains 7 level of shifting units, the very first is a single mode shifter (shift by 64-bit) used only for QP mode, and next 6 stages are dual-mode shifter which works either for shifting a 226-bit data or for shifting two-parallel 113-bit data. In the dual-mode stage architecture, two top MUXs are used to process two parallel 113-bit data for dynamic left shifting, which further can combined with the help

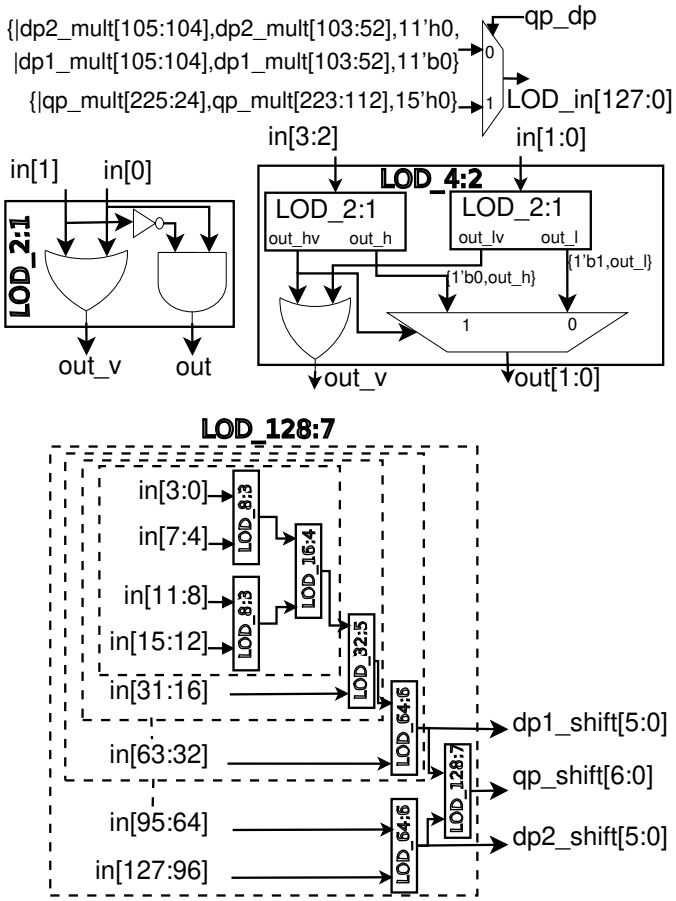

Figure 9: QPdDP Dual Mode Leading-One-Detector.

of bottom MUX to produce the dynamic left shifting for a 226-bit data (as a combined two 113-bit inputs). A pipeline register is inserted after 3 shifting stage in left shifter to meet the timings. A unified 226-bit input for dynamic left shifter is generated by combining the mantissa multiplication result of QP and both DP's as follows:

$$\begin{aligned} Left\_Shift\_in[225:0] &= qp\_dp \ ? \ qp\_mult[225:0] \\ &: \{dp1\_mult[105:0], 7'b0, dp1\_mult[105:0], 7'b0\} \end{aligned} \tag{3}$$

Dynamic right shifting of mantissa multiplication result is required when input exponents sum is less-than-equal-to BIAS, and it produces a sub-normal result. Similar to dual-mode Dynamic Left Shifter, a 128-bit dual-mode Dynamic

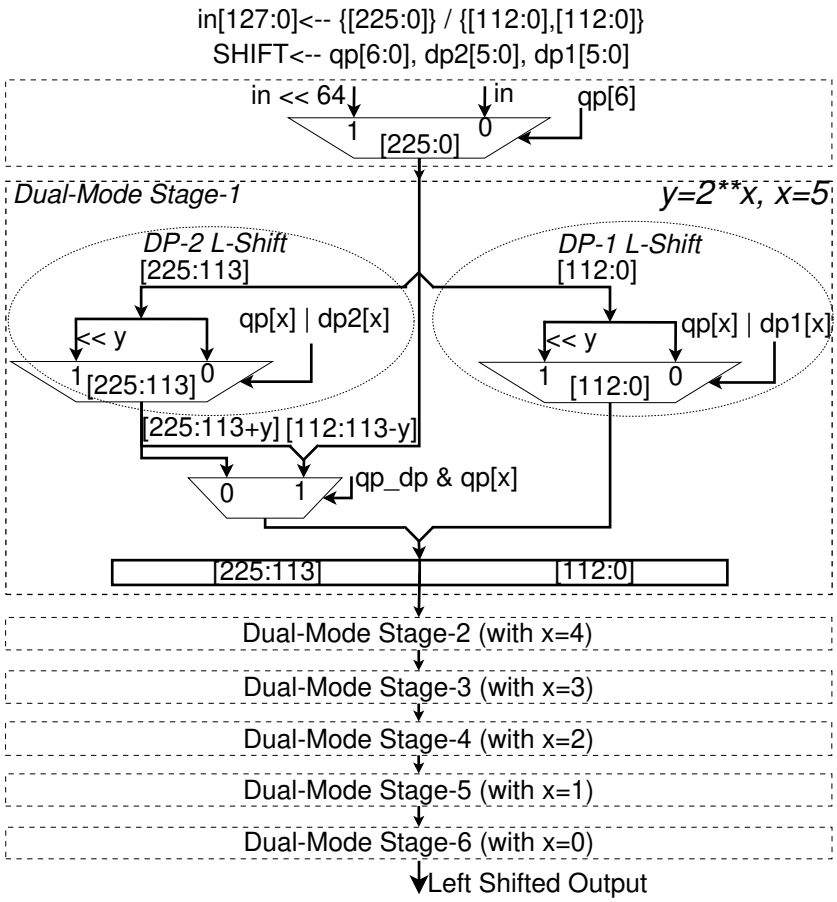

Figure 10: QPdDP Dual Mode Dynamic Left Shifter.

Right Shifter is also designed (its close details can be visualized from portion of tri-mode dynamic right shifter architecture discussed in QPdDPqSP architecture section). A unified 128-bit input for dual-mode dynamic right shifter is generated as follows:

$$\begin{aligned} &Right\_Shift\_in[127:0] = qp\_dp \ ? \ qp\_mult[225:98] \\ &: \{dp1\_mult[105:42], dp1\_mult[105:42]\} \end{aligned} \tag{4}$$

The `fourth stage` of architecture includes the 4 shifting stages of Dual-mode Dynamic-Left-Shifter unit, rounding, normalization, exponent updates and final processing. The *shifted-mantissa and normalization* unit takes the left shifted data

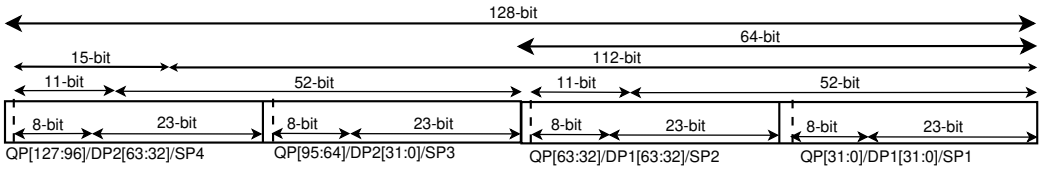

Figure 11: QPdDPqSP: Input / Output Register Format.

and right shifted data, and decides which one to forward ahead with normalization (1-bit right shifting in case of mantissa overflow). It also generates the unified shifted and normalized mantissa which consists either of QP data or both DPs data (with required shifting and normalization).

2.4. Rounding, Normalization, and Finalizing Output

Rounding includes ULP (unit at last place) generation and its addition to mantissa result. Round-to-nearest components are implemented separately, for each QP and DPs, for ULP computations, however, and its addition with mantissa is shared by using two blocks of adders with controlled carry forward. The ULP generation is based on the LSB precision bit, Guard bit, Round bit and Sticky bit.

The Exponent Update unit update exponents for mantissa underflow or overflow, which is achieved by decrementing exponents by left-shift-amount or incrementing it by one, respectively. This processing is shared among QP and DP-2 as their input exponents shares same space, however, done separately for DP-1.

The mantissas and exponents are finally updated for underflow, overflow, subnormal and exceptional cases to produce the final output. These are done separately for QP and both DPs. For overflow, the exponent is set to infinity and mantissa is set to zero. In the underflow case exponent is set to zero and mantissa takes its related computed value. The computed signs, exponents and mantissas of QP and both DPs are finally multiplexed with a 128-bit 2:1 MUX to produce the final output, which either contains a QP output or two DPs outputs.

3. Proposed Tri-Mode QPdDPqSP FP Multiplier Architecture

This section will discuss the proposed tri-mode QPdDPqSP (quadruple precision with dual double precision and quad single precision support) FP multiplier architecture. The flow of QPdDPqSP architecture is similar to that of QPdDP (in Fig. 1, which is also designed with four pipeline stages at similar computational stages. More details on the individual component design of this architecture are discussed below.

The unified QPdDPqSP input/output register format is shown in Fig. 11, which either contains a QP operand or two parallel DP operands or four parallel SP operands.

The very first component of the architecture, QPdDPqSP Data Extraction and Subnormal Handler, is shown in Fig. 12. It extracts the sign, exponent and mantissa portion for each format of floating point operands, ie. one pair of QP operands, two pairs of DP operands and four pairs of SP operands. As appears from Fig. 11, the exponents positions of QP, DP-2 and SP-4, as well as that of DP-1 and SP-2 are shared (similar to QPdDP, where QP and DP-2 exponents are sharing positions). Thus, as shown in Fig. 12, it helps in sharing the hardware resources required for subnormal and exceptional checks for these operands sets, whereas for others (SP-1 and SP-3) it is done separately.

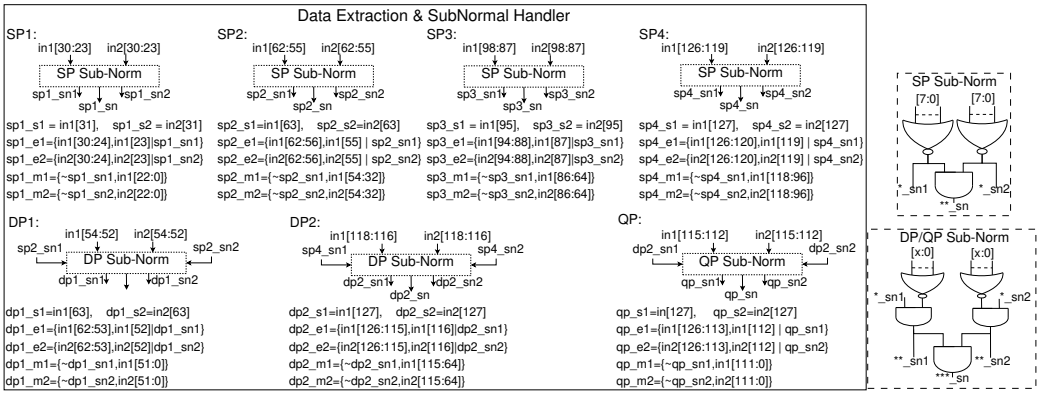

Figure 12: QPdDPqSP Data Extraction and Subnormal Handler.

The signs and exponents computations are simple and implemented as contemporary method while sharing resources for QP,DP-2 & DP-4, and DP-1 & SP-2 exponents processing.

The mantissa multiplier for QPdDPqSP is also designed using both, the Karatsuba method and newly designed tri-mode Radix-4 Modified Booth multiplier architecture.

3.0.1. Tri-Mode Karatsuba Mantissa Multiplier

The architecture of tri-mode 113x113 mantissa multiplier using Karatsuba method is shown in Fig. 13. This architecture is based on the recursive use of two-partition Karatsuba method. First, the unified mantissa multiplicand and multiplier operands are generated which either consists of QP or dual DP or quad SP

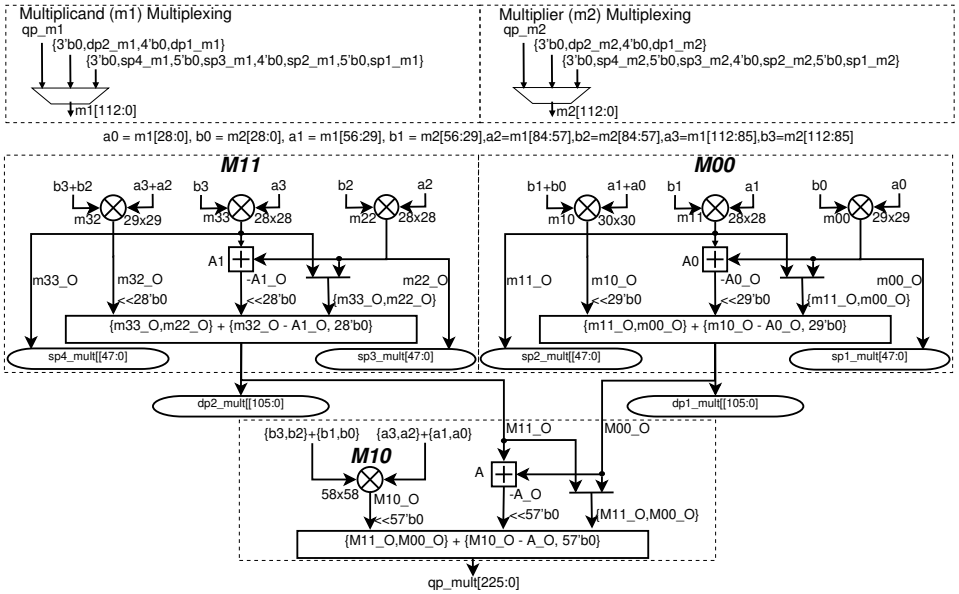

Figure 13: Tri-Mode QPdDPqSP Karatsuba Mantissa Multiplier.

operands, as shown in Fig. 13. This unification is segmented at 29th, 57th, 96th bit position for SP's operands while at 57th bit position for DP's operands.

Their implementation follows by partitioning them at 57th bit position as follows:

$$\begin{aligned} m1[112:0] &\leftarrow \{m1[112:57], m1[56:0]\} \leftarrow \{X_1, X_0\} \\ m2[112:0] &\leftarrow \{m2[112:57], m2[56:0]\} \leftarrow \{Y_1, Y_0\} \end{aligned} \tag{5}$$

Thus, similar to eq(2), their multiplication using two partition Karatsuba method would be,

$$m1 \cdot m2 \leftarrow \{Y_1 \cdot X_1, Y_0 \cdot X_0\} + \{(Y_1 + Y_0) \cdot (X_1 + X_0) - Y_1 \cdot X_1 - Y_0 \cdot X_0, 57'b0\} \tag{6}$$

Thus, it requires two 57x57 (for $Y_1 \cdot X_1$ and $Y_0 \cdot X_0$) and one 58x58 (for $(Y_1 + Y_0) \cdot (X_1 + X_0)$) multipliers. These multipliers are further partitioned at 29th bit position and implemented using two partitioning Karatsuba method.

In Fig. 13, the multiplier block $M00$ implements the 57x57 $Y_0 \cdot X_0$ (similar to Fig. [5] for QPdDP multiplier) and produces SP-1 and SP-2 mantissa multiplication results along with (by further combining them) producing DP-1 mantissa multiplication result. Similarly, the multiplier block $M11$ produces results for SP-3, SP-4 and DP-2 mantissa multiplication. The multiplier block $M10$ implements

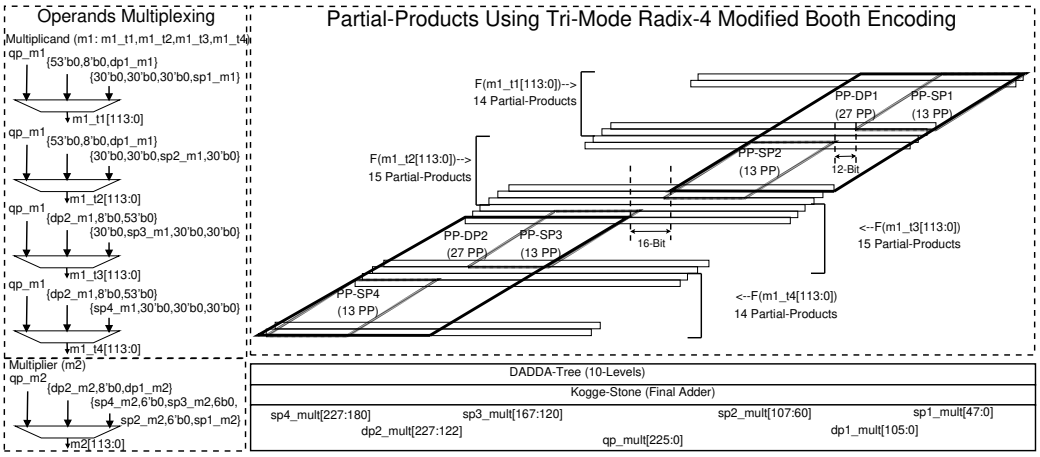

Figure 14: Tri-Mode QPdDPqSP Radix-4 Modified Booth Mantissa Multiplier.

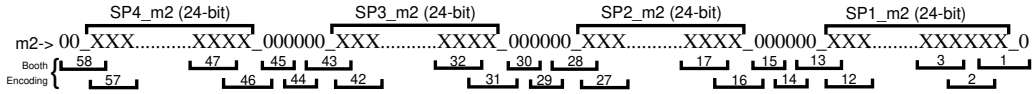

Figure 15: QPdDPqSP Booth Encoding Scenario under Quad Single Precision Processing Mode.

58x58 $(Y_1 + Y_0) \cdot (X_1 + X_0)$ multiplier using two partition method, and by combining DP-1 and DP-2 results produces QP mantissa multiplication result. In this architecture, the basic multiplier units of $M00$, $M11$ and $M10$, ie. 28x28, 29x29 and 30x30 multipliers are designed using Radix-4 MB partial product generation with 7 Dadda layers reduction tree and a final Kogge-Stone adder. A pipeline register is placed before the Kogge-Stone adder as the first pipelining stage of the QPdDPqSP architecture.

3.0.2. Tri-Mode Radix-4 Modified Booth Mantissa Multiplier

The proposed architecture of tri-mode Radix-4 MB multiplier is shown in Fig. 14. Here, a four sets of unified multiplicand (m1_t1, m1_t2, m1_t3 and m1_t4) and a unified multiplier (m2) operands are initially generated. The m1_t1 and m1_t2 are similar to m1_t1 of QPdDP MB architecture which contains either QP or DP-1 multiplicand operands, however, here, they individually also contains SP-1 and SP-2 multiplicand operands, respectively. Similarly, m1_t3 and m1_t4 contains SP-3 and SP-4 multiplicand operand, respectively, or contains QP or DP-2 multiplicand operands. The unified multiplier operand m2 either contains QP multiplier operand, or both DP multiplier operands separated by 8-bit zeros, or it

contains four SP multiplier operands each separated by 6-bit zeros.

The processing under QP and dual-DP mode is similar to the QPdDP architecture. Further, each SP contains 13 partial products, which are vertically separated by 2 partial products (both of them are zeros) and horizontally separated by 12-bit, which enables smooth processing of partial product reduction and final addition, without corrupting any neighborhood data. The Radix-4 Booth encoding scenario of multiplier operand during quad SP mode is shown in Fig. 15. A 10 layer Dadda tree is used for the reduction of partial products which finally added by a Kogge-Stone adder. A register level is inserted after 7th layer of Dadda tree for the pipelining purpose. This proposed tri-mode Radix-4 Modified Booth architecture is able to perform dynamically on either of a QP operand pair or a dual DP operands pairs or a quad SP operands pair. Compare to a traditional MB multiplier, extra MUXs used for the generation of multiplier and multiplicand operands sets act as a hardware overhead in proposed tri-mode MB architecture.

All the processing till the pipelining registers in tri-mode KM/MB mantissa multiplier are part of first stage of QPdDPqSP architecture. Remaining portion of mantissa multiplier architecture are part of second stage along with the generation of LOD_in, the input for Leading-One-Detector (Fig. 16. The 128-bit LOD_in contains MSB mantissa multiplication results either of QP or dual DP or quad SP, in chunks of 32-bit.

The third stage of architecture deals with the post subnormal processing based on the Algorithm-1, and contains Tri-mode Leading-One-Detector (LOD), tri-mode dynamic right shifter, right shift amount computations and part (4 stages) of dynamic left shifter.

The architecture of QPdDPqSP LOD is shown in Fig. 16. It is designed using hierarchical modeling, similar to that of QPdDP. Here, data partitioning is done at 32-bit level, to take out SP's left-shifting amount along with further taking out DP's and QP left-shifting amounts.

The architecture of 128-bit tri-mode dynamic right shifter is shown in Fig. 17. It contains 7 stages, first one is a single mode (only for QP) stage, the second one is a dual-mode stage (similar to a dual mode stage of QPdDP), and remaining five are stages with tri-mode functionality. The functioning of dual mode is similar to QPdDP. The tri-mode stages are designed by further partitioning the QPdDP dual-mode stage mid-way. Thus, in tri-mode stage architecture, the top-level MUXs are functioning on 32-bit segments for each SP, which further (conditionally) combined to produce shifting for both DP's and finally by (conditionally) combining them QP shifting result can be obtained. The conditional combining are dependent on the mode of processing.

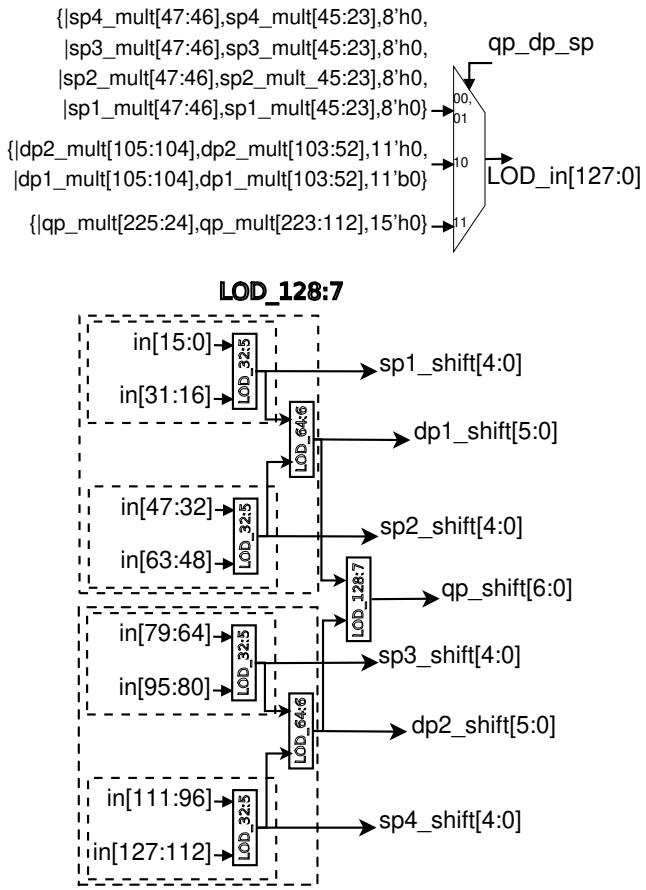

Figure 16: QPdDPqSP Tri-Mode Leading-One-Detector

Similarly, a 228-bit tri-mode dynamic left shifter is designed with 7 stages, in which first four stages are part of third pipeline stage of the architecture, and remaining 3 are part of fourth pipeline stage. It can either process a 228-bit input for QP, or two 114-bit input each for both DPs or four 57-bit inputs for each SPs. Instead of 226-bit, it is taken as 228-bit to partition it well by 4 units to handle SP's accommodation. The architectural combination of QPdDP dual-mode left shifter and QPdDPqSP tri-mode right shifter can easily complement each other to design the QPdDP dual-mode right shifter and QPdDPqSP tri-mode left shifter architectures.

The unified inputs for tri-mode left shifter and tri-mode right shifter are gen-

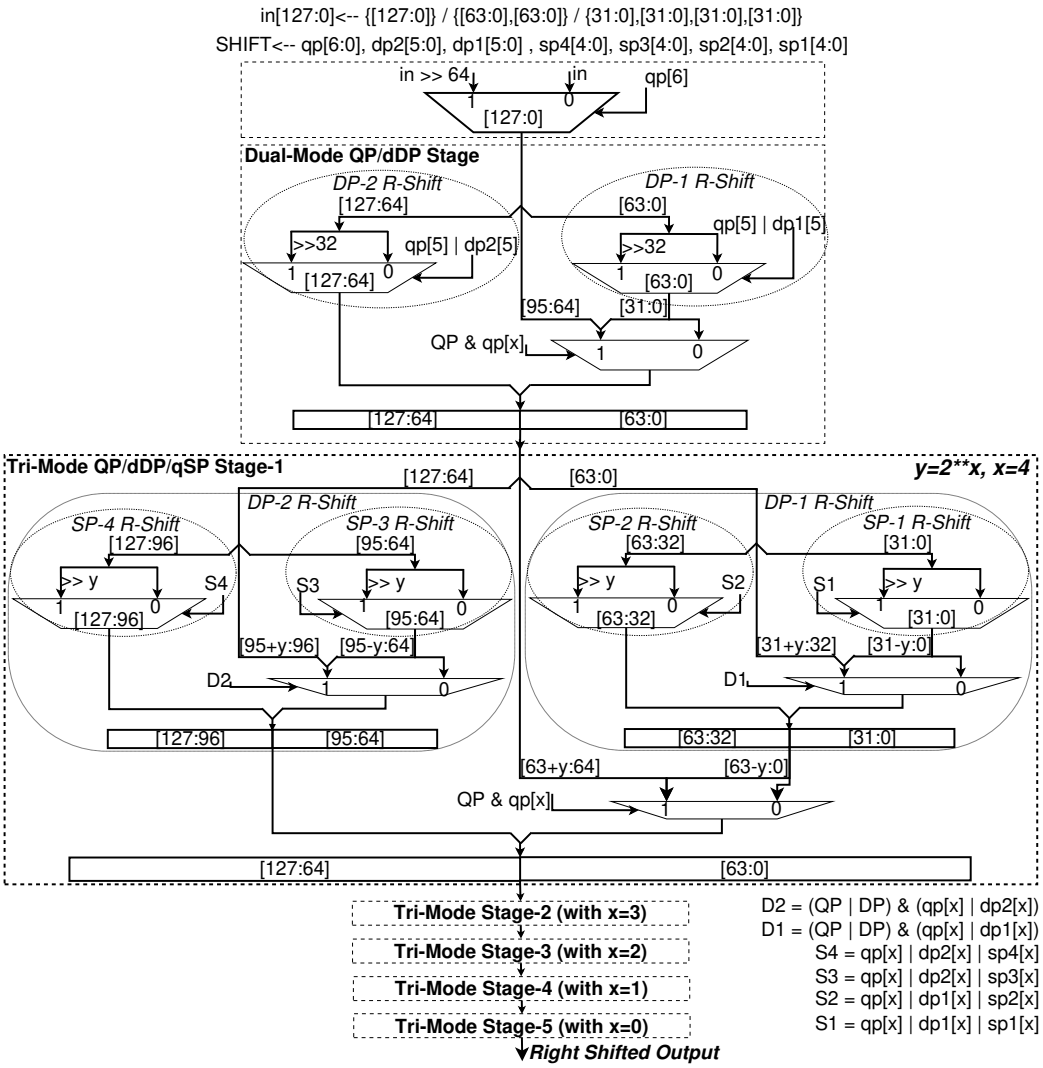

Figure 17: QPdDPqSP Tri-Mode Dynamic Right Shifter

erated as follows:

$$
\begin{aligned}
Left\_Shift\_in = QP\ ?\ \{qp\_mult, 2'b0\} \\
:\ (DP\ ?\ \{dp2\_mult, 8'b0, dp1\_mult, 8'b0\} \\
:\ \{sp4\_mult, 9'b0, sp3\_mult, 9'b0, \\
sp2\_mult, 9'b0, sp1\_mult, 9'b0\})
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
Right\_Shift\_in = QP\ ?\ qp\_mult[225:98] \\
: (DP?\ \{dp2\_mult[105:42], dp1\_mult[105:42]\} \\
: \ \{sp4\_mult[47:16], sp3\_mult[47:16], \\
sp2\_mult[47:16], sp1\_mult[47:16]\})
\end{aligned} \tag{8}
$$

After subnormal processing, the rounding ULP is generated separately for each SP's, DP's and QP, which are added to the mantissa using four blocks of adders with controlled carry forward, in order to share the resources among QP, two DPs' and four SPs' processing. Later, the exponents are updated for possible mantissa underflow or overflow, and related processing is shared among QP/DPs/SPs similar to the exponent computation in first stage. All the exponents and mantissas are further updated individually for overflow, underflow and exceptional cases and produces final result using a 128-bit 3:1 MUX, based on the processing mode.

4. Implementation Results and Comparisons

The proposed QPdDP and QPdDPqSP FP multiplier architectures are synthesized using UMC 90nm standard-cell ASIC library and implementation details are presented in Table 1. For the area and timing overhead measurement of proposed dual-mode architecture compared to a single-mode QP FP multiplier, a single-mode QP FP multiplier is also designed and synthesized using similar computation flow with Radix-4 MB method used for mantissa multiplication. The architecture with Karatsuba method is named as QPdDP_KM / QPdDPqSP_KM and that with MB method is named as QPdDP_MB / QPdDPqSP_MB. These architectures are synthesized for the best possible achievable timing constraint. These architectures are designed with latency of 4 clock cycles and have throughput of 1 clock cycles.

All the above multiplier architectures are functionally verified against Synopsys Design Compiler floating point multiplier IP (intellectual property) using extensive random test cases for each mode, with all possible combinations of input operands, like normal-normal, normal-subnormal, subnormal-normal, and subnormal-subnormal, with exceptional cases.

Compared to the single mode QP multiplier, the proposed QPdDP_MB multiplier needs 9.58% more area and 4.54% more period, with minor power overhead. Whereas, as expected, QPdDP_KM multiplier requires 1.2% less area and 40% more period than single mode QP multiplier. The very nature of Karatsuba method is to reduce area by reducing the required number of multiplier blocks, however,

Table 1: Implementation Details for Proposed Architecture

	QP (MB Method)	QPdDP_MB (Dual MB Method)	QPdDP_KM (Karatsuba Method)	QPdDPqSP_MB (Dual MB Method)	QPdDPqSP_KM (Karatsuba Method)
Latency	4	4	4	4	4
Area(μm^2)	428554	469641	423397	508857	424609
Area(gates)	142851	156547	141132	169619	141536
Period(ns)	1.10	1.15	1.54	1.20	2.10
Period(FO4)	24.44	25.55	34.22	26.67	46.67
Power(mw)	193.05	201.47	161.01	197.5	112.44

Table 2: Comparison of Multiplier Architectures

	QPdDP Multiplier				QPdDPqSP Multiplier		
	[10] (Normal)	[11] (Normal)	Proposed (With Sub-Normal)		[17] (Normal)	Proposed (With Sub-Normal)	
			QPdDP_MB	QPdDP_KM		QPdDPqSP_MB	QPdDPqSP_KM
Latency (QP/dDP/qSP)	3/2	3/2	4/4	4/4	3/3/2	4/4/4	4/4/4
Throughput[1] (QP/dDP/qSP)	2/1	2/1	1/1	1/1	2/1/1	1/1/1	1/1/1
Gate Count[2]	146566	144178	156547	141132	832133*	169619	141536
Area @ 90nm					2496400**	508857	424609
Period (FO4)[3]	48.9	49.12	25.5	34.22	88	26.67	46.67
Area × Period × Latency × Throughput (10^6) #	43.0	42.49	15.97	19.32		18.09	26.42

[1] in clock-cycle, [2]Based on minimum size inverter, [3]1 FO4 (ns) $\approx (\frac{Tech.\ in\ \mu m}{2})$

* Based on computed scaled area @ 90nm, ** Scaled area in μm^2 @ $90nm$ = (Area @ $45nm$) * $(90/45)^2$

#Gate Count × Period (FO4) × Latency × Throughput (in clock-cycle)

at cost of some extra delay due to extra-layers of adder/subtractor. Similarly, the proposed tri-mode QPdDPqSP_MB multiplier requires 18.7% more area and 9% more period than a single mode QP multiplier, while the QPdDPqSP_KM requires smaller area and more period than QP only multiplier. Thus, QPdDP_MB / QPdDPqSP_MB architectures are better for timing metric, whereas, QPdDP_KM / QPdDPqSP_KM are doing better in area saving. The power overhead for multi-mode architectures are obviously due to the large area and switching activities involved in them. The power of tri-mode architectures are litter lower than their dual-mode counterparts due to their high time-periods (low frequency).

Furthermore, both QPdDP multipliers also provide computational support for SIMD dual DP multiplication, and thus saves on 2 DP FP multiplier units. Likewise, QPdDPqSP multipliers further include SIMD support for four SP multiplier computations, and thus saves on 2 DPs' and 4 SPs' multiplier units.

A comparison with prior work on QPdDP multiplier available in literature is presented in Table 2. All the previous work on QPdDP multiplier [10, 11] consist of only an iterative mantissa multiplier and rounding circuitry. They have poor throughput, and are without sub-normal and exceptional handling support. Comparison is presented in terms of technological independent parameters like, area in

terms of Gate-Counts and period in terms of FO4 (Fan-Out-of-4) delay number. A unified metric "Area X Period X Latency X Throughput (in clock cycles)" is also used for comparison, which should be smaller for a better architecture.

The architecture presented in [10, 11] are almost similar, and have used two 57x57 integer array multipliers in iterative manner to accomplish one 114x114 (for QP mantissa) multiplication in 2 cycles or two 57x57 (for two DPs mantissa) multiplications in one cycle. Thus, their throughput is 2 cycles for QP multiplier and 1 cycles for DPs multiplication. Despite the proposed work implements a complete multiplier for better throughput, it requires only a little more area for QPdDP_MB than prior arts, however, QPdDP_KM's area metric is even better than earlier work. This is the benefit of using Modified Booth method over array/tree multiplication method, and it gives even better when used with Karatsuba methodology. Moreover, both proposed architectures have better throughput and period along with the support of sub-normal computations and exceptional case handling. Also, the fully unfolded version of [10, 11] architectures would require almost double of currently reported area. Thus, the unified metric of "Area X Period X Latency X Throughput(cycle)" of proposed QPdDP multipliers shows a significant improvement (more than 3 times) over [10, 11] architectures.

Table 2 also includes the comparison of QPdDPqSP with only available literature on it [17]. [17] has presented an architecture for QPdDPqSP FP multiplier which consists of an iterative mantissa multiplier with rounding circuitry. It has also used rectangular multiplier concept for mantissa multiplier, and has incorporated four 58x29 multipliers which can perform QP mantissa multiplication in 2 clock cycles, can process two DPs or four SPs in one clock cycle. The latency is 3/2/2 clock cycles for QP/dDP/qSP with throughput of 2/1/1 cycle for QP/dDP/qSP multiplications. With an implementation on 45nm technology, it requires a huge amount of area 624100 μm^2 with 505 MHz frequency (1.98ns or 88 FO4 period delay). The area requirement and timing metric of [17] lags far behind the proposed tri-mode QPdDPqSP FP multiplier architecture.

Among others, the architecture in [12] has used a 76x27 bit rectangular multiplier in iterative manner to process either of a DP/DEP, or two SP multiplication (however, 2 SP multiplication is carried out with a common multiplier operand). Further, [13] has also used only array/tree rectangular multiplier and rounding circuit (similar to [10, 11]) for a double precision implementation. Few other literature [14, 16, 15] have also targeted for multi-mode FMA architectures with multi-mode mantissa multiplier architecture similar to [10, 11], and thus they also face similar issue of large area and delay. Also, all the previous works do not include support for multi-mode sub-normal and exceptional handling, which

requires additional area and delay cost.

The contemporary multiplier architectures prefer to incorporate Modified Booth based multiplier due to its better area and timing metrics, and thus, is a part of present proposals in dual-mode/tri-mode fashion, instead of array/tree multiplier which needs large area with poor timings, as can be seen from above discussion. Further, by using Karatsuba method some more area reduction can be achieved, albeit, with some sacrifice on timings.

5. Conclusions

Two configurable architectures QPdDP and QPdDPqSP for quadruple precision with SIMD support for lower precision (DP/SP) computation are presented in this paper. QPdDP supports QP with dual (two-parallel SIMD) double precision, and QPdDPqSP supports QP with SIMD dual DP and SIMD quad SP floating point multiplications. Both are further explored with two kinds of mantissa multipliers technique, Karatsuba Method (KM) and currently proposed dual/tri mode Radix-4 Modified Booth (MB) multiplier. All the other key components of processing flow are designed for the efficient dual/tri mode processing. Architectures with dual/tri mode MB mantissa multipliers are better in terms of timing metric, whereas, that with KM method outperforms in terms of area metric. The proposed dual-mode/tri-mode quadruple precision multiplier architectures improve significantly on the area, timing and throughput compared to previous literature.

6. Acknowledgments

This work is partly supported by the Research Grants Council of Hong Kong (Project GRF 17245716), and the Croucher Foundation (Croucher Innovation Award 2013).

References

[1] D. H. Bailey, R. Barrio, J. M. Borwein, High-precision computation: Mathematical physics and dynamics, Applied Mathematics and Computation 218 (20) (2012) 10106–10121. doi:10.1016/j.amc.2012.03.087.

[2] F. de Dinechin, G. Villard, High precision numerical accuracy in physics research, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 559 (1) (2006) 207–210. doi:10.1016/j.nima.2005.11.140.

[3] IEEE standard for floating-point arithmetic, IEEE Std 754-2008 (2008) 1–70doi:10.1109/IEEESTD.2008.4610935.

[4] H.-J. Oh, S. Mueller, C. Jacobi, K. Tran, S. Cottier, B. Michael, H. Nishikawa, Y. Totsuka, T. Namatame, N. Yano, T. Machida, S. H.Dhong, A fully pipelined single-precision floating-point unit in the synergistic processor element of a cell processor, Solid-State Circuits, IEEE Journal of 41 (4) (2006) 759–771. doi:10.1109/JSSC.2006.870924.

[5] NXP Semiconductors, AN10902 : Using the LPC32xx VFP, in: Application note, 2010.
URL www.nxp.com/documents/application_note/AN10902.pdf

[6] Nvidia, NVIDIA's Next Generation CUDA^{TM} Compute Architecture: Kepler^{TM} GK110, in: White Paper, 2014.
URL www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[7] M. Jaiswal, R. Cheung, M. Balakrishnan, K. Paul, Unified architecture for double/two-parallel single precision floating point adder, Circuits and Systems II: Express Briefs, IEEE Transactions on 61 (7) (2014) 521–525. doi:10.1109/TCSII.2014.2327314.

[8] M. Jaiswal, B. Varma, H.-H. So, M. Balakrishnan, K. Paul, R. Cheung, Configurable architectures for multi-mode floating point adders, Circuits and Systems I: Regular Papers, IEEE Transactions on 62 (8) (2015) 2079–2090. doi:10.1109/TCSI.2015.2452351.

[9] M. K. Jaiswal, H. K. H. So, Area-efficient architecture for dual-mode double precision floating point division, IEEE Transactions on Circuits and Systems I: Regular Papers 64 (2) (2017) 386–398. doi:10.1109/TCSI.2016.2607227.

[10] A. Akkaş, M. J. Schulte, A quadruple precision and dual double precision floating-point multiplier, in: Proceedings of the Euromicro Symposium on Digital Systems Design, DSD '03, 2003, pp. 76–81.

[11] A. Akkaş, M. J. Schulte, Dual-mode floating-point multiplier architectures with parallel operations, Journal of Systems Architecture 52 (10) (2006) 549 – 562. doi:http://dx.doi.org/10.1016/j.sysarc.2006.03.002.

[12] D. Tan, C. E. Lemonds, M. J. Schulte, Low-power multiple-precision iterative floating-point multiplier with SIMD support, IEEE Trans. Comput. 58 (2) (2009) 175–187.

[13] A. Baluni, F. Merchant, S. K. Nandy, S. Balakrishnan, A fully pipelined modular multiple precision floating point multiplier with vector support, in: Electronic System Design (ISED), 2011 International Symposium on, 2011, pp. 45–50. doi:10.1109/ISED.2011.14.

[14] M. Gök, M. M. Özbilen, Multi-functional floating-point maf designs with dot product support, Microelectronics Journal 39 (1) (2008) 30 – 43. doi:https://doi.org/10.1016/j.mejo.2007.11.001.
URL http://www.sciencedirect.com/science/article/pii/S0026269207003497

[15] K. Manolopoulos, D. Reisis, V. Chouliaras, An efficient multiple precision floating-point multiply-add fused unit, Microelectronics Journal 49 (2016) 10 – 18. doi:https://doi.org/10.1016/j.mejo.2015.10.012.
URL http://www.sciencedirect.com/science/article/pii/S0026269215002591

[16] L. Huang, S. Ma, L. Shen, Z. Wang, N. Xiao, Low-cost binary128 floating-point fma unit design with simd support, IEEE Transactions on Computers 61 (5) (2012) 745–751. doi:10.1109/TC.2011.77.

[17] K. Manolopoulos, D. Reisis, V. Chouliaras, An efficient multiple precision floating-point multiplier, in: 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2011, pp. 153 –156. doi:10.1109/ICECS.2011.6122237.

[18] A. Akkaş, Dual-Mode Quadruple Precision Floating-Point Adder, Digital Systems Design, Euromicro Symposium on 0 (2006) 211–220. doi:http://doi.ieeecomputersociety.org/10.1109/DSD.2006.47.

[19] A. Akkaş, Dual-mode floating-point adder architectures, Journal of Systems Architecture 54 (12) (2008) 1129–1142. doi:10.1016/j.sysarc.2008.05.004.

[20] M. Ozbilen, M. Gok, A multi-precision floating-point adder, in: Research in Microelectronics and Electronics, 2008. PRIME 2008. Ph.D., 2008, pp. 117–120. doi:10.1109/RME.2008.4595739.

[21] A. Isseven, A. Akkaş, A dual-mode quadruple precision floating-point divider, in: Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on, 2006, pp. 1697–1701. doi:10.1109/ACSSC.2006.355050.

[22] M. K. Jaiswal, H. K. H. So, Dual-mode double precision / two-parallel single precision floating point multiplier architecture, in: 2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), 2015, pp. 213–218. doi:10.1109/VLSI-SoC.2015.7314418.

[23] A. Karatsuba, Y. Ofman, Multiplication of Many-Digital Numbers by Automatic Computers, in: Proceedings of the USSR Academy of Sciences, Vol. 145, 1962, pp. 293–294.

[24] F. de Dinechin, Large multipliers with fewer DSP blocks, in: International Conference on Field Programmable Logic and Applications, 2009, pp. 250–255. doi:10.1109/FPL.2009.5272296.

[25] M. K. Jaiswal, H. K. H. So, DSP48E efficient floating point multiplier architectures on fpga, in: 2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID), 2017, pp. 1–6. doi:10.1109/ICVD.2017.7913322.

[26] M. Jaiswal, R. Cheung, M. Balakrishnan, K. Paul, Configurable architecture for double/two-parallel single precision floating point division, in: VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on, 2014, pp. 332–337. doi:10.1109/ISVLSI.2014.45.

[27] M. K. Jaiswal, R. C. C. Cheung, Area-efficient architectures for double precision multiplier on FPGA, with run-time-reconfigurable dual single precision support, Microelectronics Journal 44 (5) (2013) 421–430. doi:10.1016/j.mejo.2013.02.021.
URL http://www.sciencedirect.com/science/article/pii/S0026269213000591

[28] L. Dadda, Some schemes for parallel multipliers, Alta Frequenza 34 (1965) 349–356.

[29] P. M. Kogge, H. S. Stone, A parallel algorithm for the efficient solution of a general class of recurrence equations, Computers, IEEE Transactions on C-22 (8) (1973) 786–793. doi:10.1109/TC.1973.5009159.