Physical Design Space Exploration

Ephrem Wu and Inkeun Cho

Xilinx, Inc.
2100 Logic Dr.
San Jose, CA 95124

ABSTRACT

A polynomial accelerator implemented with a custom high-dynamic-range number representation operates up to 534MHz in the slowest speed grade on a 28nm FPGA, a clock rate that a typical FPGA tool flow cannot achieve. This design tutorial shows how to achieve a physically scalable and high-speed numerical design by partitioning it into a cascade of identical stages, and balancing the LUT-to-DSP ratio within each stage to match the available resources on the FPGA.

Categories and Subject Descriptors

B.7.1 [Integrated Circuits]: Types and Design Styles – algorithms implemented in hardware, gate arrays, VLSI.

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

Field-programmable gate arrays; FPGAs; hardware accelerators; placement; polynomials; synthesis; floating point; digital signal processing

1. INTRODUCTION

A K^{th} -degree polynomial $h_K[n] = \sum_{k=1}^K a_k u[n]^{k-1}$ can be evaluated with Horner's method [1] as a cascade of $K-1$ multiply-add stages shown in Figure 1. The authors of [2] show that this recursion is bounded given bounded coefficients and $|u[n]| < 1$.

Figure 1 A Polynomial in $K-1$ Stages

For this design tutorial, each input sample $u[n]$ can change every clock cycle and is represented by an 18-bit fixed-point number in the range $[-(1 - 2^{-17}), 1 - 2^{-17}]$. The coefficients a_k 's are known ahead of time, have up to 17 significant bits, and are in the range $[-(1 - 2^{-17}) \times 2^7, (1 - 2^{-17}) \times 2^7]$. Mapping the Horner structure efficiently to Xilinx FPGA resources is the subject of this design tutorial.

Four design aspects are considered.

1. Even though the input can be represented as an 18-bit fixed-point quantity, the dynamic range of the coefficients a_k 's and the intermediate values r_k 's and s_k 's can exceed this range. A number representation in Section 2 facilitates efficient implementation of addition and multiplication of high-dynamic range numbers.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s).

FPGA '15, February 22–24, 2015, Monterey, CA, USA.

ACM 978-1-4503-3315-3/15/02.

<http://dx.doi.org/10.1145/2684746.2689080>

2. The Xilinx DSP48E1 slice [3] consists of a 25×18 two's complement multiplier followed by a 48-bit adder. The multiplier products (r_k 's in Figure 1) are not directly accessible for scaling. A method to pre-scale the coefficients to prevent overflow in the multiply-add unit is developed in Section 3.
3. To achieve maximum clock frequency and to ease scaling, each stage in Figure 1 is physically mapped to the FPGA to pitch-match the regular layout of the resources, a subject of Section 4.
4. Once one stage is mapped, multiple stages are cascaded to implement the structure in Figure 1. Section 5 presents the placement options beyond those provided by the standard tool flow.

2. NUMBER REPRESENTATION

2.1 Overview

Evaluating a polynomial requires a high dynamic range. A custom number representation called XDR is outlined here for efficient use of two's-complement hardware for high-dynamic-range operations. An $\text{XDR}(N, M)$ number is represented by the function $f_{N,M}(d, \varepsilon)$, such that $f_{N,M}(d, \varepsilon) = d \times 2^\varepsilon$, where d is an N -bit two's-complement significand excluding the most negative number -2^{N-1} , i.e. $|d| < 2^{N-1}$, and ε is an M -bit sign-magnitude integer exponent. Although normalization is optional in XDR and thus multiple representations of the same number are possible [4], the exponent is chosen so that the significand can be placed directly on the wires in two's complement arithmetic hardware. For instance, an $\text{XDR}(9, 7)$ number $f_{9,7}(68, -63)$ represents $68 \times 2^{-63} \approx 7.37 \times 10^{-18}$. The significand 68 is placed on some input port of at least 9 bits wide of a two's complement arithmetic circuit, and the 7-bit sign-magnitude exponent -63 is kept track of elsewhere. One can also write the same number as $f_{9,7}(136, -64)$, effectively normalizing the significand. Note that by excluding the most negative N -bit two's complement number, the zero significand is represented uniquely and there are as many negative significands as positive ones. This property results in the absolute value of any N -bit XDR significand still fitting within N bits, useful for normalization.

2.2 Specific to This Design Tutorial

Because the input sample $u[n]$ is an 18-bit fixed-point number and $|u[n]| < 1$, it can be written as $f_{18,5}(u[n] \times 2^{17}, -17)$. This value will be written as $f_{18,5}(x, \varepsilon_x)$ or sometimes just (x, ε_x) , where the significand $x = u[n] \times 2^{17}$ and the exponent is a constant $\varepsilon_x = -17$. The significand x is meant to be placed directly on the wires of an 18-bit port of a two's complement multiplier.

3. A REPEATING STRUCTURE

3.1 DSP48E1 as a Multiply-Add Unit

Horner's method evaluates a polynomial as a series of multiply-add operations. Each multiply-add unit maps to one Xilinx DSP48E1 slice (Figure 2) and forms the compute core of a

repeating structure. The multiply-add unit implements $P = A \times B + C$ as two's complement integers. Specifically, the internal product $Q = AB$ is added to C to produce P . The bit widths of A , B , C , Q , and P are $N_A = 25$, $N_B = 18$, $N_C = 48$, $N_Q = N_A + N_B = 43$, and $N_P = 48$ respectively.

Note that the XDR(18,5) significand of the input sample $u[n]$ is placed on the wires of the B port of the DSP48E1 slice.

Figure 2 Multiply-Add Unit in One Stage of the Horner Chain

3.2 Input Pre-processing

The DSP48E1 slice is an integer multiply-add unit with neither its own leading-one detector nor arithmetic shifters. To minimize the area of the polynomial accelerator, pre-scaling inputs is necessary to fuse XDR multiplication and addition within one DSP48E1. Figure 3 shows a repeating structure that pre-processes the inputs to the multiply-add unit to avoid overflow while retaining as many significant digits as possible. A cascade of $K - 1$ stages of this structure implements a K^{th} -degree polynomial. The stages are labeled according to the indices of the polynomial coefficients attached to the adders. As a result, the leftmost stage in Figure 1 is stage $K - 1$, the stage immediately to its right is stage $K - 2$, and so on until the rightmost stage, which is stage 1.

Figure 3 Repeating Structure for Place-and-Route

Stage $K - i$ performs the XDR recurrence multiply-add operation $f_{N_P,M}(w_i, \rho_i) = f_{N_P,M}(w_{i-1}, \rho_{i-1}) \times f_{N_B,M}(x, \epsilon_x) + f_{N_B,M}(b_{K-i}, \epsilon_{K-i})$, where $M = 7$ is the common exponent width in this design. A common exponent ρ_i is needed for the addition. Let $f_{N_P,M}(w_i, \rho_i) = f_{N_Q,M}(q_i, \rho_i) + f_{N_C,M}(c_{K-i}, \rho_i)$ as shown at the post adder in the DSP48E1 slice in Figure 3.

1. The first addend $f_{N_Q,M}(q_i, \rho_i)$ is the product of the input $f_{18,5}(x, \epsilon_x)$ and some version of the previous-stage input $f_{N_P,M}(w_{i-1}, \rho_{i-1})$. This version of the previous-stage input

must have the exponent $\rho_i - \epsilon_x$ or else the product with $f_{18,5}(x, \epsilon_x)$ would not result in the exponent ρ_i . This version of the previous-stage input turns out to be $f_{N_A,M}((-1)^{\text{sgn}(w_{i-1})} t_i, \rho_i - \epsilon_x)$, where the unsigned significand t_i is a shifted version of $|w_{i-1}|$, and fits into the A port of the DSP48E1 multiplier¹. The shift amount to go from $|w_{i-1}|$ to t_i is chosen to minimize loss of significant digits, and is coordinated with the shift amount for the coefficient. The bit-loss minimization algorithm that solves a recurrence formula across all stages is beyond the scope of this paper. It suffices to say that the shift amount generator E_{K-i} in Figure 3 instructs the significand of the previous-stage input to shift by the appropriate amount.

2. The second addend is some version of the coefficient, and again is shifted by the same shift amount generator that shifts the previous-stage input.

Now the DSP48E1 slice performs $w_i = (-1)^{\text{sgn}(w_{i-1})} t_i x + c_{K-i}$ with two's complement operands, blissfully unaware of all the exponent bookkeeping happening around it.

Unfortunately, the exponent bookkeeping uses many LUTs and creates a high LUT-to-DSP utilization ratio. Having too many LUTs per DSP slows down the maximum clock frequency and blocks many DSPs from being used (Figure 4). A ratio of about 300 LUTs to one DSP48E2 per stage achieves a good balance (Figure 5).

Figure 4 High LUT-to-DSP Ratio Per Stage

Figure 5 Balanced LUT-to-DSP Ratio

¹ An unsigned significand is used purely for implementation reasons and not for mathematical reasons.

4. REDUCING THE LUT COUNT

Converting some LUTs to DSPs results in a lower LUT-to-DSP ratio and is beneficial for pitch-matching each stage in the Horner form to rows of DSP48E1 slices. The absolute value block (abs_{K-i}) is a good candidate to map to a DSP48E1 slice. Taking the absolute value of an XDR significand is a bit more work than taking that of a sign-magnitude value, and even that of a two's complement value. This is because the most negative two's complement value is not allowed; its absolute value is saturated to the most positive value. For instance, a four-bit significand ranges from -7 to 7 but -8 is possible to be placed on the four wires by two's complement arithmetic units. The absolute value of -8 in this case is saturated to 7.

Let d be an N -bit XDR significand. The absolute value of d is

$$\text{abs}(d) = \begin{cases} d, & d \geq 0 \\ -d, & d = -2^{N-1} \\ -d + 1, & -2^{N-1} < d < 0. \end{cases}$$

The DSP48E1 slice can be configured to implement the XDR absolute function by setting CARRYIN to be $\text{sgn}(d) \& (-2^{N-1} < d < 0)$ and ALUMODE[0] to be $\text{sgn}(d)$.

5. MULTI-STAGE PLACEMENT

5.1 The Intuition

Now that there are two DSP48E1 slices per stage, a floorplan like that in Figure 5 can be achieved. For instance, the absolute value block occupies the DSP48E1 slice in the left column while the multiply-add block occupies the DSP48E1 slice in the right column. Since the LUTs to the left and the right of the two DSP48E1 slices may not be sufficient, not all DSP48E1 slices may be utilized in the DSP48E1 columns, resulting in a floorplan similar to that in Figure 6, in which a 5th-degree polynomial requires seven DSP48E1 slots across two rows.

Figure 6 Floorplan for Two DSP48E1 Slice Per Stage

A 5th-degree polynomial has been implemented on a -1I grade (the slowest grade) on a mid-range Xilinx Kintex™-7 XC7K410T FPGA in the Vivado™ 2013.1 environment. A 5th-degree polynomial is chosen because it is the most common polynomial degree for digital predistortion. The maximum operating frequency is 519MHz and has sufficient margin above

491.52MHz for wireless base station applications. Table 1 summarizes the resource usage.

Table 1 Implementation Results of a 5th-Degree Polynomial

	Total Used	Used Per Polynomial Degree	Available	Utilization %
FF	2726	545.2	508400	0.54
LUT	2500	500	254200	0.98
Memory LUT	388	77.6	90600	0.43
DSP48	10	2	1540	0.65
Maximum Frequency at -1I	515MHz			

The per-degree results in Table 1 show fewer resources and a higher speed compared to the single-precision fused multiply-add unit in [5], which consumes 4 DSP48E1s, 802 LUTs, and 1233 FFs with a maximum frequency of 488MHz. Obviously, the two designs are not functionally equivalent, with the polynomial accelerator presented here especially created for a hybrid fixed-point and high-dynamic-range application. These resource usage and frequency values nonetheless show it is worth the effort analyzing a high-dynamic-range design with XDR and architecting the Horner polynomial chain with only one normalizer and two shifters.

5.2 Expanding the Placement Search Space

Given a rectangular region in Figure 6 with two columns of seven DSP48E1 slices each, how good is the DSP48E1 placement in Section 5.1 for a 5th-degree polynomial?

Across the two DSP48E1 columns, there are 14 DSP48E1 slices available, from which 10 are used to implement the polynomial.

Thus there are $\binom{14}{10}10! = \frac{14!}{4!}$ different DSP placements. Assuming 10 minutes per place-and-route run, this would take over 69,110 years of run time.

Because there are two DSP48E1 slices per stage, this large search place can be pruned so that exactly five DSP48E1 slices are used in each of the two DSP columns. This pruning generates only $\binom{7}{5}^2 = 441$ placements. Along the dataflow of the polynomial, the absolute-value DSP and the multiply-add DSP are visited alternately, resulting in three reasonable orderings of these DSPs across the two columns: ping-pong, U-shaped, and S-shaped.

All placements are done stage-by-stage, starting with stage 5 of the 5th-degree polynomial.

1. The ping-pong placement has all the absolute-value DSPs in one column and the multiply-add DSPs in the other (Figure 7a).
2. The U-shaped placement requires one column of DSPs to be placed first before using the other DSP column (Figure 7b). For instance, the absolute-value blocks and the multiply-add units of stages 5 and 4 plus the absolute-value block of stage 3 occupy one DSP column, leaving the other column to the multiply-add unit of stage 3 and all of stages 2 and 1.
3. Like the ping-pong placement, the S-shaped placement places the absolute-value block and the multiply-add unit DSPs in the same stage in different DSP columns but the absolute-value DSP of one stage and the multiply-add DSP of its adjacent stage must be in the same DSP column (Figure 7c).

Figure 7 Three Placement Strategies. The green boxes prefixed ‘A’ are DSP48E1s used as absolute value evaluators. The pink boxes prefixed ‘M’ are DSP48E1s used as multiply-add units. Each box is suffixed with the stage number. Unlabeled grey boxes are unmapped DSP48E1s. (a) Ping-Pong Placement (b) U-Shaped Placement (c) S-Shaped Placement

6. RESULTS

The distributions of the three placements are plotted in Figure 6. The ping-pong placement generally yields faster designs than the other two placement strategies.

Figure 8 Placement Results

All final place-and-routed results are plotted in a histogram in Figure 9. 46.5% of the results exceed the 500MHz requirement and are represented by the green bars. With only a rectangular region constraint from Figure 6 specifying where all cells of the netlist should be placed, and without specifying which 10 of the 14 DSP48E1 slots should be used, Vivado™ 2013.1 produced a design at 473MHz on the Xilinx Kintex™-7 XC7K410T device. The fastest design achieved 534MHz, and this last 13% of speed improvement was not straightforward to obtain. The initial manual placement from Section 5.1 achieved a respectable performance at 519MHz.

Figure 9 Placement Design Space

7. CONCLUSIONS

A polynomial accelerator has been shown to operate up to 534MHz at the slowest speed grade on a 28nm FPGA. It accepts an 18-bit fixed-point input and high-dynamic-range coefficients. The FPGA polynomial accelerator has been architected with a custom number representation known as XDR. Because the two’s complement significand of an XDR number models bits that are put directly on the wires of two’s complement integer arithmetic circuits, XDR is useful for analyzing high-dynamic-range designs using two’s complement integer arithmetic operators.

The polynomial accelerator is implemented with Horner’s method and is physically partitioned into a cascade of identical stages. To balance LUTs and DSP48E1 usage, two DSP48E1s are used in each stage. One DSP48E1 slice serves as the XDR absolute value evaluator and the other DSP48E1 slices is the multiply-add unit. Place-and-route experiments of a 5th-degree polynomial including three DSP48E1 placement strategies reveal a design space with a range of maximum clock frequencies from 460MHz to 534MHz using Vivado 2013.1 on a -1I (slowest) grade of the Xilinx Kintex™-7 XC7K410T device. These experiments provide space for placing 10 DSP48E1s into 14 DSP48E1 slots, which are partitioned into two 7-slot DSP48E1 columns. An intuitive placement (the ping-pong placement) with unused DSP48E1 slots in strategic locations to give more room for LUTs proves to be effective since it yields a maximum frequency of 519MHz, 13% above what automatic placement can achieve and very close to the high end of the 460MHz-to-534MHz design space.

8. REFERENCES

- [1] W. G. Horner, “A new method of solving numerical equations of all orders, by continuous approximation,” *Phil. Trans. R. Soc. Lond.*, Part II, vol. 109, pp. 308–335, July, 1819.
- [2] C. S. Burrus, J. W. Fox, G. A. Sittou, and S. Treitel, “Horner’s method for evaluating and deflating polynomials,” DSP Software Notes, Rice University, Nov. 26, 2003, <http://www.ece.rice.edu/dsp/software/FVHDP/horner2.pdf>
- [3] 7 Series DSP48E1 Slice User Guide (UG479), version 1.8, Nov. 10, 2014.
- [4] S. Boldo and M. Daumas, “Properties of two’s complement floating point notations,” *Int. J. Softw. Tools Technol. Transfer*, vol. 5, no. 2, pp. 237–246, Mar. 2004.
- [5] Xilinx LogiCORE IP Floating-Point Operator v6.2 Product Guide, PG060, Dec. 2012.

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available “AS IS” and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising from, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx’s limited warranty, please refer to Xilinx’s Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx’s Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.