AREA ESTIMATION OF LOOK-UP TABLE BASED FIXED-POINT COMPUTATIONS ON THE EXAMPLE OF A REAL-TIME HIGH DYNAMIC RANGE IMAGING SYSTEM

Michael Kunz, Martin Kumm, Martin Heide, Peter Zipf

Digital Technology Group
University of Kassel
34121 Kassel, Germany
email: {mikunz, kumm, heide, zipf}@uni-kassel.de

ABSTRACT

In many FPGA-based designs, fixed-point computations can be efficiently implemented by look-up tables. However, the precision of these computations has a great influence on the hardware costs. We present two simple models for fast but precise area estimation without synthesis, that can be used for word length optimization. As an example we analyze a look-up table based implementation of a circuit for processing image combination and tone-mapping of high dynamic range (HDR) images. The results can be generalized and be applied to any problem with soft accuracy requirements.

1. INTRODUCTION

Image processing algorithms, like tone mapping HDR, usually work with relatively low word lengths as each color of a CMOS camera has a word length from 8 to 12 bit. The low word lengths make the LUT-based realization of operations and functions attractive, as less FPGA resources are used compared to other function realizations like, e. g., Taylor series or Chebyshev approximation which need a large amount of multipliers [1]. However, intermediate values of the computation are non-integer. Therefore, the image quality is significantly affected by the impact of rounding errors. These depend on the word length of the calculated values. On the other hand, the hardware complexity depends on the word length of the circuit. So, quality measures are needed that accommodate the relation between the required implementation effort and the image quality. Based on this quality measure an estimation for the required implementation effort is derived in this work.

2. WORD LENGTH OPTIMIZATION

The objective of word length (or bit width) optimization is to find a word length in each section of the data path of a circuit such that the hardware costs are minimized and the introduced error at the output is below a user-defined criteria.

Word length optimization methods are often classified into analytical and statistical analysis [2]. Statistical analysis methods are based on simulating the circuit with given input stimuli and tracking the minimal word size in each section of the data path. This leads to designs that are close to the optimum but no error bound can be guaranteed as using different input stimuli may lead to larger errors or even overflows. Analytical analysis methods usually result in more conservative word lengths as they use worst case range and error models of operators in a design. They are often faster and can guarantee a lower bound for the error [2].

One of the most challenging tasks is to predict the hardware costs as a function of the word length. Targeting FPGAs, a measure for the number of LUTs (or slices) is usually used as cost function. Most methods rely on pre-synthesized models with resource usage stored in a database for accurate resource estimation [3, 4]. However, for arbitrary functions every new function has to be evaluated by synthesis runs for each word length which is very inconvenient. A more generic approach is estimating the LUTs or slices without synthesis during cost evaluation. One approach is to analyze the netlist after high level synthesis [5]. However, the computational effort is still high for large designs. Hence, two very simple analytical models are proposed here which produce an error estimation without high level synthesis. Using these models, a complete search is possible due to the fast evaluation.

3. AREA ESTIMATION

3.1. Worst Case Model

In a general way a cost function can be found by computing the worst case storage space from the input and output word lengths. For the analysis of the error, we have to define an error metric that specifies the quality of computation. This error metric is application specific and has to be defined carefully, because it represents the quality measure.

For the word length range of interest, the calculation of the error metric can be done by, e. g., a complete search using

Matlab. For these values the cost function is calculated as a measure of the required implementation effort. The values which have the lowest cost and the highest quality of computation are selected to form a pareto front.

The prediction is that a design with an appropriate quality can be realized with maximum estimated costs. Therefore, it is necessary that there is a linear – or at least a continuous – relationship between the estimated and the real costs. An high accuracy approximation of the absolute number of LUTs or slices like in [3]-[5] is not necessary, higher errors of the model should simply correspond with higher LUT costs. Thereby, the assumption is that if the word length of the LUT is changed (and therewith its size) all other parts of the circuit, e. g., downstream adders will be also changed almost linear, at least if it is not a dedicated block, e. g., an embedded multiplier.

As we will see in Section 5 the worst case model provides a very pessimistic estimation. In particular, when the LUTs can be well optimized the errors from estimation become large. A better estimation is disussed in the following.

3.2. Enhanced Model

As mentioned before, the optimization of the LUTs during the synthesis process leads to large estimation errors. For example, if many parts of LUT columns are equal, under certain conditions the LUTs can be simplified. This leads to smaller LUTs (respectively a lower number of LUTs) which has an impact on the required implementation cost.

If the LUT content is included within the cost function, it is possible to find a more precise estimation for the needed LUTs. However, the problem is to reproduce the synthesis steps which is very time-consuming and does not make much sense. So a less time-consuming but reasonably realistic estimation for the real LUT size is searched. A simple approximation of the real LUT size can be found by canceling the duplicate rows and reduce the LUT by the constant column parts. Note that in general the LUTs are not independent from each other, so the reduction potential for a LUT depends on the word length of other LUTs. This leads to a set of cost values for each LUT. This usually happens when an output word length of a LUT is the input word length of another LUT.

As in the worst case model the error metric is used to calculate the pareto front. This leads to a more realistic estimation, as we can see in Section 5.

4. HDR IMAGING

4.1. Overview

HDR images provide the possibility to represent synthetic or natural scenes with more than the 256 luminance levels present in regular digital images to make visible additional details in dark and bright areas by combining a sequence of low dynamic range (LDR) images with different exposures. The result is an image that can not be displayed with a common monitor and can not be printed. To do this, the image needs to be "tone mapped", i. e., the dynamic is fit to the original dynamic of an LDR image. These computations are quite time consuming. Although many HDR methods and tone mapping operators have been published in the recent years, only a few of them focus on real-time or an hardware efficient implementation and even less focus on the realization using FPGAs. Real-time HDR is done by a specialized ASIC ([6], [7], [8]) or a graphic processing unit (GPU) ([9], [10]).

4.2. HDR Core Algorithm

A core algorithm for HDR fusing from multiple exposures is given in Reinhard et al. [11] and is called "Luminance HDR". Based on the fact that nearly all information in an image exists in the luminance, the algorithm calculates the luminance of each HDR pixel. The HDR image is computed for all pixels L_{ij} from N exposures by

$$L_{ij} = \frac{\sum_{k=1}^{N} \frac{Z_{ij_k} \cdot w(Z_{ij_k})}{\Delta t_k}}{\sum_{k=1}^{N} w(Z_{ij_k})} \tag{1}$$

where Z_{ij_k} are the 8-bit pixel luminance values, $w(Z_{ij_k})$ is a weight function to exclude under- and overexposed pixels and Δt_k is the exposure time for exposure k.

In our case for tone mapping only Δt_k is set to '1' and the hat function $w(Z_{ij_k})$ is chosen to

$$w(Z_{ij}) = 1 - \left(\frac{2}{255} Z_{ij}\right)^4 \tag{2}$$

This method was selected because it is easily scalable in the number of images used, is well suited for other operators like removing ghosting artifacts, can provide a tone mapping image, is extensible for color images, and can be well partitioned into hardware.

As can be seen in (1) the weight function respectively the inverse of its sum is the essential part of the computation. These can be efficiently realized using LUTs. So with a small set of additional adders and multipliers the Luminance HDR was realized. Figure 1 shows the block diagram for the parallel processing of three images without pipeline registers.

5. IMPLEMENTATION AND RESULTS

5.1. Implementation

The design was synthesized for a Spartan 3e using ISE 13.4 with speed optimization without register duplication. Without any additional optimization the design is able to produce a real-time tone mapped HDR image up to a medium error of

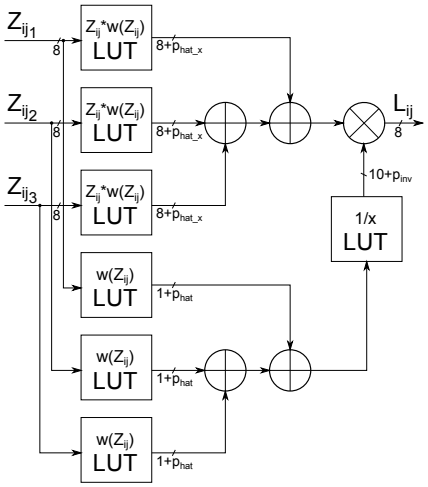

Fig. 1. Block diagram for HDR computation with three images without pipeline registers.

$28.61 \cdot 10^{-3}$ for 1600×1200 bit images with a frame rate of 30 fps, which means that the occourance of an error by one digit is very rare. The RGB version of the design was implemented in an Atlys board with camera module.

The word length optimization is separated into the analysis of the dynamic range and the precision. The analysis of the dynamic range is done analytical by a worst case scenario. In our case, the tone mapped HDR image has to satisfy the dynamic of 256 different values, so the dynamic range is 8-bit. The analysis of the precision is done also analytical by error models.

As outlined in Section 3, an error metric and a cost function must be chosen. The error metric is chosen by the mean error of the output pixel compared to its floating point values. Another error metric could be, e. g., the maximum luminance error. The error metric has been calculated by a complete search from 4 up to 16 bit precision. The cost function $c(p_{hat_x}, p_{hat}, p_{inv})$ is given by the total bit size of all LUTs given as

$$c = 3 \cdot [f_{hat_x}(p_{hat_x}) + f_{hat}(p_{hat})] + f_{inv}(p_{hat}, p_{inv}) \quad (3)$$

with f_{hat_x}, f_{hat} and f_{inv} being the LUT size depending on the word lengths of the precision p_{hat_x}, p_{hat} and p_{inv}.

5.2. Worst Case Model

The parts of the cost function are given by the maximum bit size of all LUTs

$$\begin{aligned} f_{hat_x}(p_{hat_x}) &= 2^8 \cdot (8 + p_{hat_x}) \\ f_{hat}(p_{hat}) &= 2^8 \cdot (1 + p_{hat}) \\ f_{inv}(p_{hat}, p_{inv}) &= 2^{2+p_{hat}} \cdot (10 + p_{inv}) \end{aligned} \quad (4)$$

Figure 2 shows the pareto front for the worst case model. It is interesting that plateaus occur. These arise from changes in a word length of one of the LUTs which have a big influence on the costs but less on the error metric. However, these word lengths build the basis for a better quality of computation.

Figure 3 shows the relationship between the estimated bitsize and the realized number of LUTs in an FPGA. It can be seen that a close upper bound function exists. Indicating that the proposed estimation is closely related. However there are steps which denote that a design which was evaluated with higher costs can be realized with clearly less costs. This model forms an upper bound, but it can be assumed that this pessimistic estimation is not able to locate the best word length combinations. This will be improved by the enhanced model.

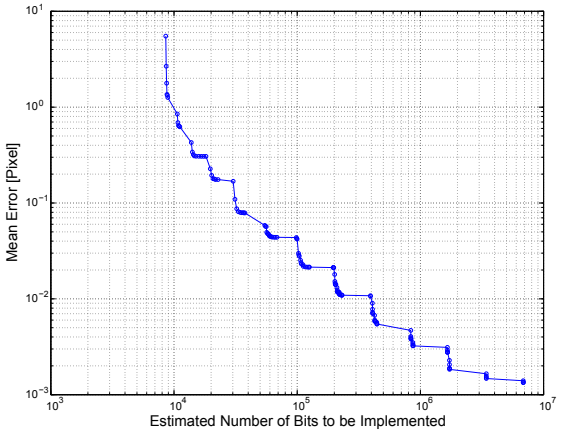

Fig. 2. Pareto Front for the worst case model.

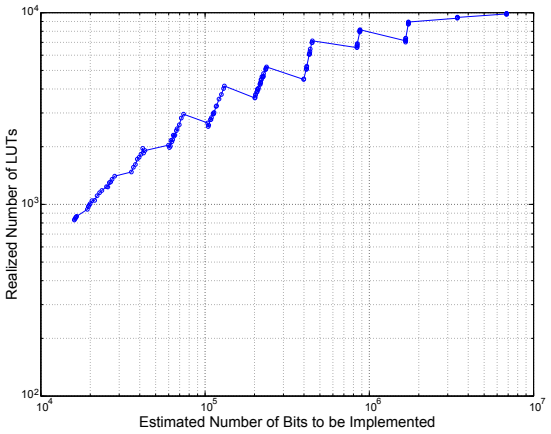

Fig. 3. Relationship between estimated and realized area for the worst case model.

5.3. Enhanced Model

As mentioned in Section 3 the cost function is calculated for a set of dependent LUT cost values. The cost function c in

this case is given by the total LUT sizes after minimization.

Figure 4 shows the pareto front for the enhanced model. In comparison with Figure 2 the plateaus are much less distinctive, parts of the front are quasi linear. Also the estimated required implementation effort is much more similar between neighboring points.

The relationship between the estimated bit size and the realized number of LUTs is shown in Figure 5. Compared with Figure 3 the relationship is nearly linear. The proposed estimation is very closely related. It is noticeable that the simplified LUT minimization produces a much better estimate. So it can be stated, that this model produces an acceptable estimation of the required implementation effort and is able to locate good word length combinations.

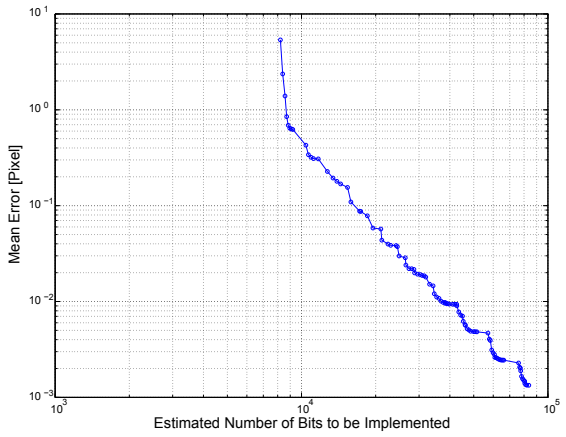

Fig. 4. Pareto Front for the enhanced model.

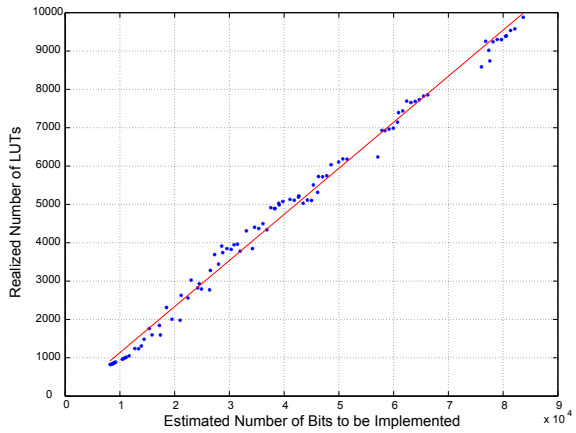

Fig. 5. Relationship between estimated and realized area for the enhanced model.

6. CONCLUSION

For many FPGA-based designs fixpoint computations are needed. Low word lengths make a LUT-based realzation of operations and functions attractive. For such designs two simple models for area estimation are presented. The worst case model delivers a pessimistic estimation whereas the enhanced model delivers a more realistic estimation. For both models cost functions and an error metric were shown. Both models are clearly much simpler than previous models enabling a complete search for word length optimization. It was shown that these simple models can be used for an approximation which is accurate for practical image processing. The models are used for a real-time HDR imaging system.

7. REFERENCES

[1] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays," Springer Verlag, 2007.

[2] K. Han, "Automating transformations from floating-point to fixed-point for implementing digital signal processing algorithms," Ph.D. dissertation, The University of Texas at Austin, Aug. 2006.

[3] N. Herve, D. Menard, and O. Sentieys, "Data wordlength optimization for FPGA synthesis," *Signal Processing Systems Design and Implementation, 2005. IEEE Workshop on*, pp. 623–628, 2005.

[4] G. A. Constantinides, "Word-length optimization for differentiable nonlinear systems," *ACM Trans. on Design Automation of Electronic Systems*, vol. 11, no. 1, pp. 26–43, Jan. 2006.

[5] M. Xu and F. Kurdahi, "Accurate prediction of quality metrics for logic level designs targeted toward lookup-table-based FPGAs," *IEEE T-VLSI*, vol. 7, no. 4, pp. 411–418, 1999.

[6] C.-T. Chiu, T.-H. Wang, W.-M. Ke, C.-Y. Chuang, J.-S. Huang, W.-S. Wong, and R.-S. Tsay, "A 100MHz real-time tone mapping processor with integrated photographic and gradient compression in μm technology," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, oct. 2008, pp. 25 –30.

[7] T.-H. Wang, W.-S. Wong, F.-C. Chen, and C.-T. Chiu, "Design and implementation of a real-time global tone mapping processor for high dynamic range video," in *Image Processing, 2007. ICIP 2007. IEEE Int. Conf. on*, vol. 6, 16 2007-oct. 19 2007, pp. VI –209 –VI –212.

[8] W.-M. Ke, T.-H. Wang, and C.-T. Chiu, "Hardware-efficient virtual high dynamic range image reproduction," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, nov. 2009, pp. 2693 –2696.

[9] N. Goodnight, R. Wang, C. Woolley, and G. Humphreys, "Interactive time-dependent tone mapping using programmable graphics hardware," in *Proceedings of the 14th Eurographics workshop on Rendering*, ser. EGRW '03. Eurographics Association, 2003, pp. 26–37.

[10] H. Zhao, X. Jin, and J. Shen, "Real-time tone mapping for high-resolution hdr images," in *Cyberworlds, 2008 Int. Conf. on*, sept. 2008, pp. 256 –262.

[11] E. Reinhard, G. Ward, S. Pattanaik, P. Debevec, W. Heidrich, and K. Myszkowski, *High Dynamic Range Imaging – Acquisition, Display, and Image-Based Lighting*, 2nd ed. Morgan Kaufmann, 2010.