

# Efficient Bit-Serial Constant Multiplication for FPGAs

Florian Dittmann

Bernd Kleinjohann

Achim Rettberg

University of Paderborn/C-LAB

Fürstenallee 11, 33094 Paderborn, Germany

Phone: +49-5251-60-6110, Fax: +49-5251-60-6065

WWW: <http://www.c-lab.de/>

Email: {roichen, bernd, achim}@c-lab.de

## Abstract

*This paper describes how to realize place-effective synchronous bit-serial constant multiplications, which can be efficiently used for a block of constants (Multiple Constant Multiplication). The architecture combines traditional concepts and new approaches, which leads to the possibility of simultaneous fast multiplications of different input values. Fast multiplications are a core for up-to-date embedded systems, as they rely on fast input processing which can be offered by common transformations. As an example, we have realized the Discrete Cosine Transformation (DCT), where we can show that our architecture features an optimal trade-off between area and speed.*

## 1 Introduction

Up-to-date tasks for embedded systems are increasingly performance intensive and therefore need a fast underlying hardware architecture. An approach to design such hardware architectures is to combine gained and proven results which accrue from general purpose computers. Known architectures of modern general purpose computers are flexible concerning adoption to new operations, especially multiplication, including different signs and numbers. They are often optimal concerning speed, but usually need much chip area.

Embedded systems often cannot deal with the general purpose concepts, as the shortage of chip area and the limitation of operation speed is an important issue. However, as embedded systems mainly fulfill specialized tasks and only need a limited freedom of adaptability, many purpose-built architectures offer a variety of possibilities for improvements and optimization; especially the here discussed constant multipliers show an optimal trade-off between area and speed.

In the past, multiplication in embedded systems stood often for time-consuming and place-ineffective tasks,

general concepts for improvements are either very special or even often completely missing. The standard optimization methods established in practical usage combine well-known implementation tricks like shifting and avoidance of unnecessary adding.

Especially when implementing multipliers, the general trade-offs of embedded systems, speed and area, must be considered. To overcome these trade-offs new design concepts or the combination of different design paradigms are demanded. The past experience has shown, that changing from bit-parallel to bit-serial architecture leads to remarkable improvements concerning space, which is an important issue for FPGA and ASIC design. Bit-serial operators are smaller and mainly independent of word width. Furthermore, bit-serial architectures requires only an equally small number of input and output pins.

One of the current main fields of embedded systems is sound and video processing, where streaming data must be compressed and processed fast, reliable and in some cases with timing requirements. Standard solutions are transformations (Fourier, Laplace) and their new enhancements like FFT<sup>1</sup> or DCT<sup>2</sup>, which enable real-time behaviour. Nevertheless, the most optimized algorithms still rely on a noticeable amount of calculations, which is challenging concerning the increasing dimension of input data.

Considering a practical task in more detail, the DCT as the decoding part and IDCT as the inverse encoding part of the MPEG algorithm are two of the most computation intensive functions in compression. Therefore, a fast and optimized DCT/IDCT implementation is essential in improving the performance of the video coder and decoder in an embedded system.

The paper is structured as follows: First, we will look at related work, based on historical evolution of data computation both from theoretical and practical view. Section 3 describes the general ideas of bit-serial ar-

<sup>1</sup>Fast Fourier Transformation

<sup>2</sup>Discrete Cosine Transformation

chitecture followed by the optimization we used. As a practical example, the DCT implementation with bit-serial architecture is explained and results are listed. Finally, a conclusion will focus on future problems and tasks.

## 2 Related Work

All kinds of multiplications are important operations in computer technology. They are both in theoretical and practical aspects research fields perhaps since human discovered information technology. Therefore, we decided to split the trace of related work into two sections: architectures and algorithms.

Serious architecture for constant multiplication dates back to the beginnings of computer science in the 1950th where people like von Neumann [1] analysed different ways of data processing including multiplications under both general aspects, bit-serial and bit-parallel design. Later, several approaches were proposed for minimizing the number of operations after substitution of multiplication with constants by shifts and additions [2]. Algorithms for optimizing the number of shifts and additions have been described in compiler research [1], [8], [9]. To minimize the number of shifts is well studied in the digital signal processing (DSP) area. Furthermore, approaches can be found in particular in the digital filter design community [3]. The minimization of the number of shifts required for multiplications with constants is thoroughly studied, but not much attention has been paid to the simultaneous optimization of the multiplication of a variable by multiple constants. The work of Chatterjee et. al. [4] addressed this problem by describing two optimization methods for the minimization of the number of operations in vector-matrix product representation of linear systems. The methods are greedy and simulated annealing-based. Although those two algorithms, based on a number splitting technique, are theoretically of high interest.

In addition there exist several other concepts of multiplication, e. g. the architecture FLYSIG<sup>3</sup>[6], which uses asynchronous concepts and was motivated by establishing the possibility of power reduction. An interesting operator group of the FLYSIG architecture are the multiplier units. Two different types of multipliers have been implemented in form of repeated addition. The first type are constant coefficient multipliers, that means the input value is multiplied with a constant. The second type multiplies two different input values and is called full multiplier. Both multiplier types are based on binary multiplication. Each type of bit-serially implemented

<sup>3</sup>FLYSIG = data**FL**ow oriented dela**Y**-insensitive **SIG**nal processing

multiplication has to be adapted to the data-width linearly which reduces area complexity from  $O(n^2)$  for a parallel multiplier to  $O(n)$ . The multiplier is of linear complexity because exactly one adder block per bit of the data word is instantiated. Many numerically intensive applications, like filters in signal and video processing, have computations that involve a large number multiplications of one variable with several constants. An optimization of this part of the computation which is in literature known as the multiple constant multiplication problem (MCM), often results in a significant improvement in several key design metrics [5], [10].

## 3 Bit-Serial Architecture

### 3.1 General Concepts

Today's embedded system design uses often specialized architectures. In our approach we focused on a bit-serial synchronous architecture, where it is possible to cope with numbers in two different ways:

- MSB (Most Significant Bit) first
- LSB (Least Significant Bit) first.

We decided to use the LSB first alternative in our system as it is mainly necessary for our algorithms which accrue from paper-based mathematical operations, where LSB, i. e. beginning from the right is a common computation practice. Furthermore, the decision for bit-serial architecture leads to the following straight-forward design of the standard cells.

Bit-serial architecture makes it necessary to operate with special registers. The used registers are capable to store one bit for the period of a clock cycle. After this cycle the information is transmitted to the output and the next information can enter the register. These registers offer the main functionality of shifting numbers. The multiplexers we are using within our architecture are multiplexers from standard libraries (e. g. Cadence).

In order to implement the most important basic mathematical calculations plus and minus, two different elements are necessary, which in general are of nearly the same architecture. Clocked bit-serial adders work with a predefined empty internal latch (equivalent to the register described above), whereas subtractor elements consist of the same latch, which is set to one. Both consist of an adder cell without memory, which performs the mathematical operation of adding A and B considering the carry value. An example layout for an adder can be found in figure 1.

The next fundamental part of algorithms are multiplications which can be realized as a combination of

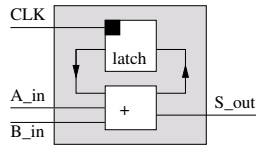


Figure 1: Clocked Adder with Internal Latch

adders and registers. We discuss this in the following section. Constant-free multiplication can lead to a non-linear architecture, which is a reason why especially in Digital Signal Processing (DSP) multiplication with constants is frequent.

Therefore, in this paper we will focus only on the optimization and implementation of constant multiplications. Our approach offers the freedom to fold those constant multiplications and to build up a field of constants, thus resulting in a very efficient hardware architecture.

### 3.2 Optimizing Bit-Serial Constant Multiplication

The optimization of constant multiplication was driven by several aspects. Firstly, we did not want to create more drawbacks than already inherited by using bit-serial architecture, which mainly means that processing speed should not increase. This went hand in hand with the intention to reduce space by saving core cells. Secondly, the new architecture should harmonize with up-to-date design aspects like low power consumption and fast prototyping. Therefore, we had to investigate the general problem from the theoretical (i. e. mathematical) and practical (i. e. hardware) view.

Multiplying a number with a constant means to sum up the number accordingly often, in binomial systems as often as the occurrence of ones in the constant, including respective shifts. Standard designs without any optimization therefore have long computation times and much overhead for the general organization, e. g. several registers and different counters.

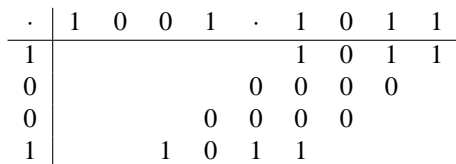


Figure 2: Traditional Paper-Based Multiplication

Considering traditional paper-based multiplication (see fig. 2), the core concepts of our architecture can be seen. After noting the problem in the top line, people use to write the intermediate results (multiplied lines,

here multiplied with 1 or 0) below each other, which includes beginning in the appropriate column. Subsequently, a vertical addition follows, where again traditional paper-based proceeding is used. These two steps – writing the needed lines below each other and adding up those lines afterwards – transformed into computer hardware architecture can now be merged into one general design performing both.

In order to show the procedure, we attach a time line to figure 2 running at the bottom from right to left (deriving from LSB and bit-serial operation) and move the second step (the addition) to the right, thus operating horizontally. Now, the constant can be found at the left, and the multiplier is shifted diagonally from the top right to the bottom left side.

Our hardware architecture combines the two steps using the slightly changed steps, which are described in the last paragraph. Considering the addition step where summation is only performed if the constant on the left possess as 1, we represent the constant in our architecture by the existence or lack of corresponding adders. Numbers are shifted from left to right through the latches by using LSB first. In case there is a connection of the output of a latch and the input of an adder, the digit of the constant is 1, 0 in all other cases.

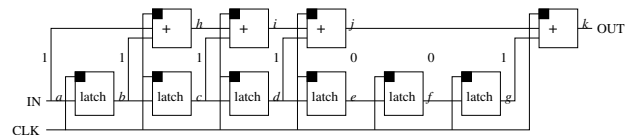


Figure 3: Constant Multiplication

An example of a constants shows figure 3. How this architecture works can be seen in table 1 by referring to the names of the output in figure 3. Assume, the number 1001 is multiplied with the constant 100111. 1001 can be found in column a, and is then shifted until column g. The result (10110011) is in column k, where again the LSB bit is calculated first.

time	a	b	c	d	e	f	g	h	i	j	k
1	1	0	0	0	0	0	0	1	1	1	1
2	0	1	0	0	0	0	0	1	1	1	1
3	0	0	1	0	0	0	0	0	1	1	1
4	1	0	0	1	0	0	0	1	1	0	0
5	0	1	0	0	1	0	0	1	1	0	0
6	0	0	1	0	0	1	0	0	1	0	0
7	0	0	0	1	0	0	1	0	0	0	1
8	0	0	0	0	1	0	0	0	0	1	1
9	0	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0	1

Table 1: Logic on Lines

The stated core concept fulfills our general requirements of no additional time costs and a reduction of space. Next, we optimized this basic architecture by looking at the structure of constants in binomical systems. Back-to-back occurrence of 1s offers this potential for optimization by using subtract statements. As in serial architecture it is similar to add a row of  $n$  1s (e.g. 1 1 1 1) or to add one 1 and subtract  $n$  ticks later 1 (1 0 0 0 -1) many constant multiplications can be reduced to a minimum of necessary operations. The process can also be seen as multiplying with a constant which is one bigger than the original constant and compensating this error by later subtracting one. These operations need additionally subtractors, which are in our architecture of equal costs.

Figure 4 shows the example above by using this optimization. We use the same constant (1001111), yet represented by 1010000 and the mandatory subtraction leading to 101000-1.

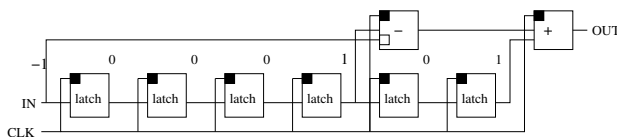


Figure 4: Constant Multiplication with Subtractor

Further, transformations often consist of multiple constant multiplications, which can be realized easily using our architecture. It is possible to use the chain of latches for several constants, which will save space by simply removing redundancy.

Furthermore, it is possible to combine the constants by splitting the single constants and finding equality. Independent of bit-parallel or bit-serial architectures the Multiple Constant Multiplication Problem deals with this idea. It uses algorithms like iterative pairwise matching. In order to show the general concept, we consider two numbers  $x = 10101$  and  $y = 10110$ . By using Boolean algebra, it is possible to find consensus and define the general layout.  $x$  and  $y$  match in the first and third digit, so we will use this as common basis (10100). The missing, but not in both numbers occurring 1s, are considered afterwards, which leads to  $x = 10100 \& 00001$  and  $y = 10100 \& 00010$ .

The combination of both ideas (Boolean algebra for field of constants and application of subtract statements) makes it possible to reduce the needed elements even further. An example of optimized multiple constant multiplication shows figure 5.

We show the procedure of generating the layout in figure 6. First we use the optimization applicable at back-to-back occurrence of 1s, then we look for equalities and split the constants into one common and two

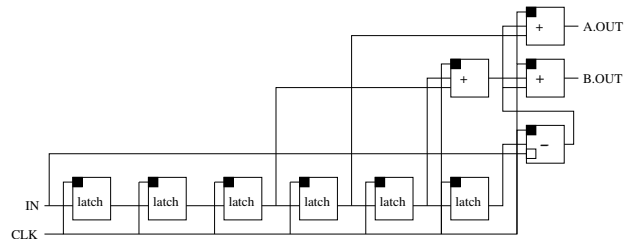


Figure 5: Optimized Field of Constant Multiplication

individual parts. These steps lead to the very small and compact layout depicted in figure 5. Equal results can be obtained for most practical multiple constant multiplication problems, an example is shown in the next section.

$$A = 1001111 = 101000-1 = 100000-1 \& 0010000$$

$$B = 1100111 = 110100-1 = 100000-1 \& 0101000$$

Figure 6: Generation of Constants

## 4 An example: the DCT Implementation

A very important part of the MPEG encoding and decoding process is the Discrete Cosine Transformation (DCT), which enables handling with compressed data. One of the best known solutions for the DCT is the Chen-Whang algorithm [7]. The DCT is reduced to a minimum of essential operations, grouped in a butterfly architecture. In order to use our architecture for the constant multiplications with all possible optimizations, a restructuring of the dataflow was necessary.

With help of the mathematical tool *Maple* and simple algebraic transformations (e.g. commutative law), it was possible to transform the algorithm to the shown structure (fig. 7), where all multiplications are done first. This transformation led to a layout which is suitable for the constant field multiplication we described in the section above.

In figure 8 we show a part of the whole DCT. The figure shows the constant multiplication field linked to the inputs X0 and X7 in figure 7. Here we can see the effectiveness of the presented approach to implement constant multiplication fields.

In order to realize the architecture, we implemented it in VHDL. The implementation starts from bottom-up. Therefore, we set up a library of components which consists of basic blocks at the bottom and register transfer components at the highest level. To simulate the bit-serial architecture components and the realized DCT we

used as a target platform an FPGA board and several tools from different EDA vendors. To synthesize and map the vhdl design on a Xilinx Virtex XE400 FPGA we used the ISE tool from Xilinx. The simulation was made with the ModelSim tool from Mentor Graphics.

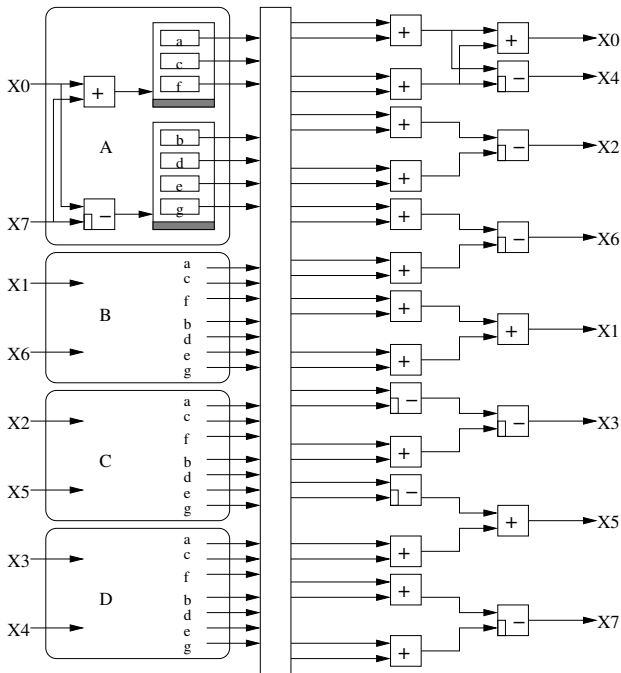


Figure 7: Discrete Cosine Transformation

## 5 Results

For compare reasons we implemented the DCT in two different manners. Firstly in the described architecture and secondly only on behavioural level. The behavioral level has no constant multiplication fields. The results of the mapping to a Virtex XE400 can be found in table 2.

In the given example we started with a 1D-DCT, that means we realized only the half part of a fully 2D-DCT used in MPEG encoder algorithm. A 1D-DCT operates only on 8 rows or 8 columns of the video stream. Therefore, a single DCT reads only one row or one column.

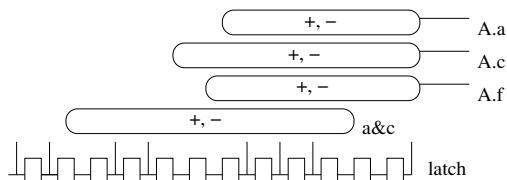


Figure 8: Outline of Multiple Constant Multiplication

We started with the implementation of a single DCT on the Virtex. The results given in number of slices and look-up tables (LUT) in comparison with the behavioral implementation are enormous, because the single DCT implemented with the described optimization uses only 9 % of the LUTs in opposite to 77 % used by the behavioral implementation. However, the whole 1D-DCT (which consists of 8 single DCTs) can be implemented in the target FPGA. Then, the amount of usage is 58 % of the LUTs and 99 % of the number of slices. It was not possible for us to implement the behavioral realization of the whole 1D-DCT by tool and size problems. Nevertheless four instances of a single DCT use 255 % of the Virtex FPGA. We could see, that results for the inverse DCT are of the same magnitude, thus using our architecture a 1D-IDCT fits into an FPGA as well.

An ASIC implementation of a 2D-DCT which is not comparable to our approach is described in [11]. That paper focused on the design of a complete processor and not on the optimization of the multiplications.

Recapitulating, our bit-serial synchronous architecture with constant multiplication fields leads to a remarkable reduction of space, mainly as few operator cells are needed, see table 2

## 6 Conclusion

In this paper we described a feasible architecture of synchronous bit-serial multiple constant multiplication. Therefore, we showed the combination of the design in paper-based multiplication and its possibilities of realization in hardware. The realization included the possibility to reduce the amount of needed core cells for one and especially multiple constants, with the optimization based on the general Multiple Constant Multiplication problem and its algorithm (iterative pairwise matching).

As a result, the paper shows that the constant multipliers are especially efficient for filter algorithms being used by signal preprocessing applications. These filter algorithms frequently use constants. Future work will concentrate on the extension of the bit-serial synchronous architecture. Obviously, dynamic re-configuration is a very interesting aspect.

For the design of e. g. a video processing unit in an embedded system, this new architecture represents a novel approach, especially regarding the multiple constant multiplication problem. One example is the DCT task of the MPEG algorithm where performance, cost and size of the implementation in hardware are of vital importance. The results of the described example show the effectiveness of the architecture.

	number of slices	percentage of 4800	number of 4 input LTUs	percentage of 9600
behavioural				
one	4,674	97%	7,426	77%
two	8,381	174%	13,126	136%
three	12,829	267%	20,032	208%
four	15,100	314%	24,568	255%
five	18,877	393%	30,705	319%
six	25,441	530%	39,620	412%
architecture				
DCT one	1,435	29%	873	9%
half (eight)	4,798	99%	5,624	58%

Table 2: Results

## References

- [1] R. Bernstein, *Multiplication by Integer Constants*, Software - Practise and Experience, Vol. 16, No. 7, pp. 641-652, 1986
- [2] G.W. Reitwiesner, *Binary Arithmetic*, in *Advances in Computers*, Vol. 1, Academic Press, pp. 261-265, New York, NY, 1960
- [3] F. Catthoor, et. al., *SAMURAI: s General and Efficient Simulated Annealing Schedule with Fully Adaptive Annealing Parameters*, *Integration*, Vol. 6, 1988
- [4] A. Chatterje, et. al., *Greedy hardware optimization for linear digital systems using number splitting and repeated factorization*, *IEEE Transaction on VLSI Systems*, Vol. 1, No. 4, pp. 423-431, 1993
- [5] Miodrag Potkonjak, Mani B. Srivastava, and Anantha P. Chandrakasan, *Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination*, *IEEE Transactions of Computer Aided Design of Integrated Circuits and Systems*, Vol. 15, No 2, 1996
- [6] Achim Rettberg, Andreas Hennig, Bernd Kleinhjohann, *Re-Configurable Multiplier Units of the Asynchronous FLYSIG Architecture*, 10th NASA Symposium on VLSI Design, Albuquerque, March 2002
- [7] Zhongde Wang, *Fast Algorithms for the Discrete W Transform and the Discrete Fourier Transform*, *IEEE Transactions on Acoustic, Speech, and Signal Processing*, Vol. ASSP-32, No. 4, 1994
- [8] Kenneth A. Dockser, *Method and Apparatus for Computer Implemented Constant Multipliation with Multipliers Having Repeated Patterns Including Shifting of Replicas and Patters Having at least Two Digit Positions With Non-Zeor Values*, United States Patent, 5,841,684, 1998
- [9] Nigel Pail Dyer, *Fixed-Coefficient serial Multiplication and Digital Circuits therefore* European Patent, 0213854, 1986
- [10] Michael Chen, Jing-Yang Jou, Hen-Ming Lin, *An Efficient Algorithm for the Multiple Constant Multiplication Problem*, *International Symposium on VLSI Technology, Systems, and Applications*, June 1999
- [11] David Johnson, Venkatesh Akella, Brett Stott, *Micropipelined Asynchronous Discrete Cosine Transform /DCT/IDCT) Processor*, *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 6, No. 4, 1998