AN OPTIMAL ALGORITHM FOR LOW POWER MULTIPLIERLESS FIR FILTER DESIGN USING CHEBYCHEV CRITERION

Georgios Karakonstantis and Kaushik Roy
School of Electrical and Computer Engineering, Purdue University
{*gkarakon, kaushik*}*@ecn.purdue.edu*

ABSTRACT

In this paper, we propose a novel finite impulse response (FIR) filter design methodology that reduces the number of operations with a motivation to reduce power consumption and enhance performance. The novelty of our approach lies in the generation of filter coefficients such that they conform to a given low-power architecture, while meeting the given filter specifications. The proposed algorithm is formulated as a mixed integer linear programming problem that minimizes chebychev error and synthesizes coefficients which consist of pre-specified *alphabets.* The new *modified coefficients* can be used for low-power VLSI implementation of vector scaling operations such as FIR filtering using computation sharing multiplier (CSHM). Simulations in 0.25um technology show that CSHM FIR filter architecture can result in 55% power and 34% speed improvement compared to carry save multiplier (CSAM) based filters.

Index Terms— Multiplierless digital filter design**,** low power, chebychev criterion, optimization methods

1. INTRODUCTION

With the continual need for improving the data rates at reduced power dissipation, the need for improved complexity reduction schemes for Digital Signal Processing (DSP) systems still persists today. One of the most widely used operations performed in DSP and most common in multiple constant multiplication (MCM) problems, referred to the multiplication of a variable by a set of constants is FIR filtering. Simple low-complexity implementations of digital filters using signed powers-of-two (SPT) [1], canonical signed digit (CSD) [2] and minimal signed digit (MSD) [3] number representations have been reported by many researchers. The optimization of MCM operation has often been accomplished by using shift-and-add multiplication algorithms with common sub expression elimination (CSE) [4]. Many approaches [5] start from a given optimal filter solution and find different quantization schemes that reduce the implementation cost in the vicinity of an optimal solution. Such schemes applied in post-synthesis phase of FIR filters, although simple, cannot guarantee an optimal solution with the desired frequency response.

In this paper, we explore complexity reduction in FIR filters from the point of view of removing computational redundancy in the *synthesis* phase of multiplierless filters, rather than following the traditional post-synthesis approach. Given a filter specification and a "pre-specified set of alphabets" (the details of which will be explained shortly) we synthesize "modified coefficients" so that proper trade-off between computation/communication complexity and filter quality can be made. Instead of trying to eliminate the common sub-expressions in the coefficient vector, we propose an algorithm that slightly modifies the coefficients and satisfies the filter design constraints. The new coefficient vector reduces redundant computation and is suitable for the implementation of filters with CSHM architecture which increases computation re-usability and power savings compared to other architectures [6].

The rest of this paper is organized as follows. In section 2 vector scaling operation and a review of CSHM architecture is presented. Section 3 presents optimal FIR filter design. Section 4 describes the proposed approach and compares the results of different synthesized filters. In section 5 the implementation of CSHM FIR filters using different alphabets and modified coefficients is compared to CSAM based filters. Finally, conclusions are drawn in section 6.

2. VECTOR SCALING AND CSHM ARCHITECTURE

The multiplication of vectors by scalars is referred as vector scaling operation. For instance, multiplication of a coefficient vector $C = [c_0, c_1, \ldots, c_{M-1}]$ with a scalar $x(n)$ is a vector scaling operation which is commonly used in DSP tasks such as filtering and matrix multiplication. In vector scaling operations, we can decompose any scalar s_i to smaller bit sequences a_k such that s_i can be rebuilt from these sequences by few shifts and adds. A decomposition of a simple vector scaling operation $[c_0, c_1]\cdot x$, is shown in Table 1. If $(0001)x$, $(0011)x$, $(0111)x$, $(1011)x$ are available, the entire multiplication is reduced to a few add and shift operations. We refer to these smaller bit sequences a_k as *alphabets*. Also, an alphabet set is a set of alphabets that spans all the coefficients in vector *C*. In the above example the alphabet set is {1, 3, 7, 11}.

CSHM architecture is based on the algorithm explained above. CSHM is composed of a precomputer, select units and final adder (SSA) as shown in Fig. 1. The precomputer computes the multiplication of alphabets and input vector *X* in advance and store them for re-use. When a scalar comes in, it is divided into smaller bit sequences that have the same length as that of an alphabet. The Shift unit, right shifts these sequences to determine the matching alphabet and generates control signals which are the index signal to the mux and the shift signal to the ISHIFT. A 2-to-1 mux takes care of the zero (0000) coefficient value determined by a select signal, sent by the Shift unit. The A:1 mux, where A defines the number of alphabets used, selects one precomputed value based on the index signal. The ISHIFT unit, correspondingly, left shifts the selected value from the A:1 mux. Finally, the output values are properly shifted and added to generate the final result.

Table 1: Vector scaling operation

Coefficients	Decomposition
c_0=01100111 (103)	$c_0 \cdot x = 2^5 \cdot (0011) \cdot x + (0111) \cdot x$
c_1=10001011 (139)	$c_1 \cdot x = 2^7 \cdot (0001) \cdot x + (1011) \cdot x$

This research was sponsored in part by Semiconductor Research Corporation (1122.001).

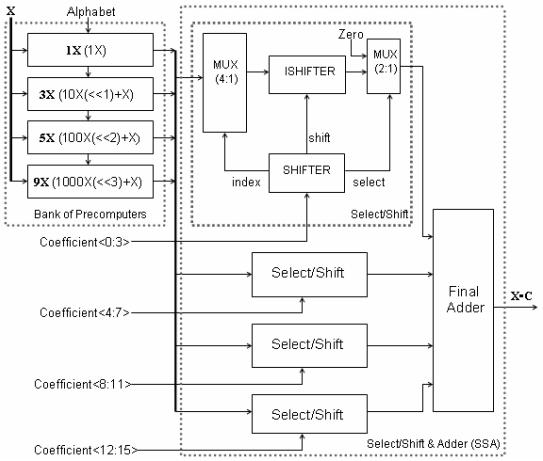

Fig. 1: Computation sharing multiplier implementation (A=4)

In order to cover all possible coefficients and to perform general multiplication operation efficiently, 8 alphabets {1,3,5,7,9,11,13,15} can be used [7]. It should be noted that the number of alphabets directly translates to power dissipation of the pre-computer unit, while the number of communication buses (coming out of the precomputer) is also proportional to the number of alphabets. However, for non-adaptive filters, the number of coefficients can possibly be reduced to achieve lower routing complexity and lower power dissipation and is the subject of discussions in section 4.

3. FIR FILTER DESIGN

FIR filters have been traditionally designed using optimization techniques with the objective of reducing the difference between the desired frequency response and the computer generated filter. Usually this difference is specified as a weighted function given by $E(\omega)=W(\omega)[H(e^{j\omega})-D(e^{j\omega})]$, where $H(e^{j\omega})$ denotes the frequency response of the digital transfer function to be designed, $D(e^{j\omega})$ is the desired frequency response and $W(\omega)$ is a positive weighting function [8]. A commonly used approximation measure is the Chebychev or min-max criterion which is the objective function (OE.1) of the optimal linear phase FIR filter design problem that can be stated as: find a set of coefficients $h(k)$ that minimizes the maximum-weighted absolute error defined as:

$$minimize \quad \|E(\omega)\| = \max_{\omega\in\Omega}\{W(\omega)|D(\omega)-G(\omega)|\} \qquad \text{(OE.1)}$$

where Ω: set of disjoint frequency bands in the range $0\le\omega\le\pi$

$$\text{s.t.} \quad \sum_{k=0}^{N-1} h(k)2\cos(2\pi(n-k-\tfrac{1}{2})\omega) - \frac{E(\omega)}{W(\omega)} \le D(\omega) \qquad (1)$$

$$-\sum_{k=0}^{N-1} h(k)2\cos(2\pi(n-k-\tfrac{1}{2})\omega) - \frac{E(\omega)}{W(\omega)} \le -D(\omega) \qquad (2)$$

The above formulation corresponds to design of even order (N) linear phase filters. There are four different kinds of transfer functions for linear phase filter design. The above formulation can be extended easily to all four cases by substituting constraints *1* and *2* with the appropriate equations, presented in [9].

4. PROPOSED OPTIMAL ALGORITHM

4.1 Problem definition

As noted above, earlier approaches have tried to optimize MCM operation starting from a given coefficient vector (post-synthesis phase). Given a filter specification, our aim is to generate filter coefficients in the synthesis phase that can be used for the implementation of multiplierless filters, specifically CSHM filters which have shown increased power and speed improvement than earlier implementations. However, earlier approaches to CSHM based filter [7] used pre-specified fixed coefficients and the complete set of 8 alphabets for its implementation. *The novelty of our approach lies in the generation of coefficients such that they conform to a given low-power architecture, while meeting the given filter specifications by using the pre-specified "good alphabets."* Hence, proper trade-off between computation/ communication complexity and filter quality can be made and CHSM based filters can be implemented using fewer alphabets. For instance in the example of section 2, if we select to use the alphabet set {1, 3}, then the coefficient vector has to be slightly modified to c_0=01101000 (104), c_1=01100110 (138). As a result we need only two precomputer banks instead of four. This reduces the power required to operate these units. Also, power dissipation depends on the signal switching activity [10]. By selecting alphabets that have fewer number of *ones*, such as {1, 3, 5} we expect power consumption to be further reduced.

Two types of filter coefficient synthesis methodologies are more commonly used than others. They include i) optimal techniques, such as those based on Mixed-Integer Linear Programming (MILP) and ii) suboptimal techniques which include local search and stochastic optimization. As an exhaustive search method, MILP guarantees the optimal solution, but the computational time increases with the number of filter taps. On the other hand, local search methods can find a 'good' solution, in reasonable time, but might be far from the optimal. Therefore, we formulate the assignment of alphabets to coefficients as MILP which can be solved by using branch and bound technique [11]. However, in order to use branch and bound technique for the optimal solution, we need to impose certain constraints on the MILP formulation.

4.2 Algorithm Description and MILP Constraints

In this section, each step of the optimization algorithm and the corresponding MILP constraints are described. The proposed idea is illustrated in Fig. 2 and can be separated into two phases.

In phase 1, the filter characteristics and the parameters of the algorithm are given as inputs. These include the order N of the filter, the passband and stopband frequencies, the desired frequency response $D(e^{j\omega})$ and the number of bits L and W that represent the coefficients and the alphabets respectively. Note that we classify the alphabets into 4 groups (*Gset* in Fig. 2) based on the following considerations: (a) number of *ones* in each alphabet and (b) number of shifted values that each alphabet can produce. The vector of the input alphabets, *Aset* is selected from the *Gset* groups. Coefficients are represented in sign magnitude format.

In phase 2, each coefficient is synthesized using the pre-specified set of alphabets, *Aset*, conforming to the given filter characteristics. This phase can be divided into five steps:

Step 1: A vector *Vset* containing the shifted values of input alphabets *Aset* is *generated*. Consider for instance alphabets 1, 3, 5 and 7. The resulting *Vset* for these alphabets is shown in Fig. 2. We note that by selecting alphabets that have less numbers of *ones* (refer to Section 4.1), power consumption can be reduced.

Step 2: Each *L-bit* coefficient is represented by $S= L/W$ sections. Each section n_i, $i=0,..,S$ consists of a *W-bit* sub-expression which represents an alphabet or one of its shifted values.

Step 3: Assignment of the alphabets or their shifted values from *Vset* to each section n_i of each coefficient. To represent this step, in

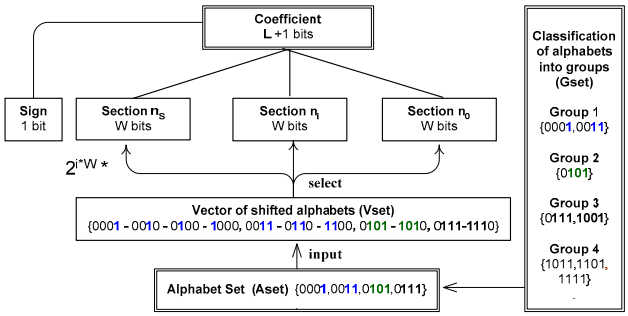

Fig. 2: Composition of each coefficient

the MILP formulation, we impose the following constraints:

For each section i :

$$\sum_{k=0}^{N-1}\sum_{a=0}^{A} sel_{k,a} = 1 \ , \quad i = 0,\ldots, S \qquad (3)$$

$$n_{i,k} = \sum_{k=0}^{N-1}\sum_{a=0}^{A} sel_{k,a} * Vset(a) \ , \quad \text{A: size of } Vset \qquad (4)$$

where sel_i are binary (0-1) variables, and n_i represent sections of each coefficient. We impose constraint *4* in order to ensure that only one shifted value of the alphabets is assigned to each section. For instance, if we consider the 1st section (*i=1*) of the 1st coefficient (*k=0*) and assume *A=4,* the constraint *4* will be:

$sel_{0,0} + sel_{0,1} + sel_{0,2} + sel_{0,3} = 1$. This expression together with constraint *5* ensures that only one of the 0-1 variables is equal to 1 and hence, only one alphabet *a* is assigned to section i=1.

Step 4: Each section is *multiplied* by a factor 2^{i*W} to determine the significance of each section (location) and for *shifting* them properly to compose each coefficient (Table 1). For instance if *L=12 bits* are used for the representation of the coefficients and each section is represented by *W=4* bits then each coefficient consists of *S=3* sections. The least significant section, n_0 is multiplied by 1, the next section n_1 is multiplied by 16 (shifted by 4) and the most significant n_2 by 256 (shifted by 8). We note that scaling the output preserves the filter characteristics, but results in an overall magnitude gain equal to the scale factor [8].

Step 5: Finally the shifted sections from step 4 are *added* together to represent the coefficients. The constraint that we have to impose in MILP formulation to implement steps 4 and 5 is given as:

For each coefficient k:

$$h(k) = sign(h(k))\sum_{i=0}^{S-1} 2^i * n_{i,k} \quad , k=0,\ldots, N-1 \qquad (5)$$

where *sign* denotes the signs of the coefficients. Note that steps 4 and 5 combined with step 3 can be seen as an iterative procedure where in each iteration only one alphabet is selected and assigned to each section, the combination of which compose new coefficients.

Following earlier post-synthesis approaches, we can start from a given optimal filter solution and apply the above procedure. In this case the iteration stops when the difference between the desired and the new composed coefficients is minimized. However such an approach does not have much control over any changes in filter specification due to the changes in filter coefficients.

4.3 Final MILP Problem Formulation

In this section, we present the overall optimization algorithm. As we noted, the aim of our approach is to guarantee the optimality of the filter response. Therefore, it is necessary to combine the optimal initial filter design procedure (*OE.1*, constraints *1*, *2*) with the *alphabet* restrictions (constraints *3-5*) described in the previous section. The problem now becomes a complex discrete optimization problem and can be stated as follows: *Determine the set of coefficients h(k) that consist of specific sub-expressions (or sections) by satisfying the alphabet constraints and minimizing the maximum-weighted absolute error.*

In order to reduce the complexity of this nonlinear problem and formulate it as MILP problem, we linearize the maximum weighted absolute error which is the objective function given in *OE.1*, by introducing two positive variables $e_1(\omega)$ and $e_2(\omega)$. As a result, the objective function 1 (*OE.1*) is substituted by constraints *6*, *7* and the final formulation with the new objective function (chebychev criterion) is given by:

$$minimize \quad E(\omega) \qquad (OE.2)$$

$$s.t. \quad E(\omega) \geq e_1(\omega) + e_2(\omega) \qquad (6)$$

$$D(\omega) - G(\omega) = e_1(\omega) - e_2(\omega) \qquad (7)$$

combined with constraints *1 to 5* which have already been discussed in previous sections.

4.4 Results

In this section we present results to demonstrate the power and potential of the proposed approach. We have implemented linear filters of even order, by using different alphabet sets and the errors in the objective function (*OE.2* - chebychev error) scaled by 10^3 are presented in Table 2. Also, different bit widths were used for the representation of the synthesized coefficients.

A closer observation of Table 2 reveals some interesting results. First, as expected, with a larger alphabet set, the difference between the initial frequency response and the acquired solution reduces. It is clear that the maximum-weighted absolute error reduces as the order of filter and the number of bits, used for the representation of the coefficients increases. Note that Initial is the error of the filter that results from the initial design procedure that approximates the desired frequency response $D(e^{j\omega})$, without any *alphabet* constraint. Most of the earlier algorithms that try to optimize MCM operations are heuristic, providing no indication how close the solutions are to the optimum in contrast to our approach. In case of filters 2, 4 and 7 we present the best possible solution (BP) that can be found from the branch and bound technique. We note that the difference between the error of the proposed algorithm and BP solution (reported by GAMS) is small.

In Fig. 3 the scaled magnitude response of a bandpass filter is presented to demonstrate the quality of the filter designed using our approach. We have noticed that by reducing the alphabet set to four, including for instance the alphabets {1, 3, 5, 9}, the error can be very small and as a result the frequency response, resulting from the new coefficient vector is close to that of the desired frequency response (Fig. 3). A small alphabet set with less number of *ones* in its representation is suitable for the low-power implementation as noted earlier. The mixed integer package of CPLEX and GAMS [12] was used for the modeling and solving

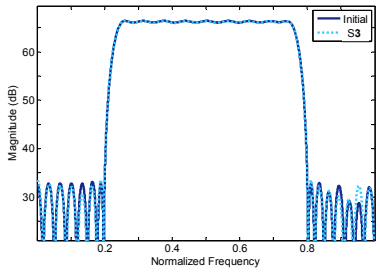

Fig. 3: Log magnitude response of *N=64* band-pass filter using S_3

Table 2: Error in Objective function (OE) and Best Possible solution (BP) of example filters, using different alphabet sets

Filter	Filter Specification					Results		S_7	S_6	S_5	S_4	S_3	S_2	S_1
	Type	Pass	Stop	Order	bits	($x10^{-3}$)	Initial	$\{S_6,11\}$	$\{S_5,15\}$	$\{S_4,13\}$	$\{S_3,7\}$	$\{S_2,9\}$	$\{S_1,5\}$	$\{1,3\}$
1	Low	0.45	0.47	128	8	OE	28.37	62.73	69.95	70.72	70.05	66.78	71.77	71.25
2	Band	0.35 0.75	0.30 0.80	64	12	OE	23.34	25.91	25.37	25.28	25.73	25.51	27.73	39.85
						BP	-	23.34	23.34	23.34	23.36	23.38	23.63	36.35
3	Low	0.40	0.50	128	12	OE	1.18	3.75	4.45	4.51	10.04	9.97	10.49	12.71
4	Low	0.25	0.30	84	12	OE	8.6	9.49	9.6	9.63	12.11	13.06	18.75	38.33
						BP	-	8.65	8.66	8.67	10.91	10.84	16.55	25.13
5	Low	0.40	0.48	64	12	OE	21.38	23.75	23.82	24.37	24.51	23.94	24.64	24.69
6	Low	0.50	0.65	32	12	OE	6.64	11.83	8.57	12.89	10.91	13.23	17.95	21.6
7	Low	0.50	0.58	44	12	OE	14.68	16.38	16.26	15.98	17.7	17.8	37.18	36.93
						BP	-	14.9	14.72	14.7	16.07	16.06	33.68	33.7
8	Band	0.33 0.66	0.25 0.75	32	12	OE	50.03	52.23	52.26	51.54	52.28	54.41	53.45	55.4
9	Low	0.50	0.58	12	16	OE	109.7	110.54	110.59	110.8	111.34	111.8	112.3	122.26
10	Low	0.25	0.30	64	16	OE	21.18	21.69	21.24	23.78	24.03	22.75	25.62	32.38
11	Low	0.45	0.47	128	16	OE	14.86	15.12	15.1	15.31	16.71	15.52	22.29	65.73
12	Low	0.40	0.42	256	16	OE	0.77	1.01	16.1	7.38	7.36	10.69	13.39	18.63

of the above problem and none of the solutions required more than a few minutes of CPU time resulting in stable, linear phase filters.

5. FIR FILTER IMPLEMENTATION

We implemented transposed direct form FIR filters of order N=12 in 0.25µm technology, using 17x17 CSHM architecture with different number of alphabets and CSAM [10]. The coefficients that were s*ynthesized* by the optimal algorithm were used in the implemented filters. Simulation results show that as we decrease the size of *Aset*, CSHM out-performs CSAM architecture with respect to power and performance, albeit with gradual filter quality degradation (Fig.4a). As we have shown, the *modified coefficients* are generated by maximum sharing of subexpressions by reducing the number of alphabets and as a result the overhead of muxes and precomputer is further reduced. In case of 2-alphabet CSHM, there is a 37% power and 11% performance improvement when compared to 8-alphabet CSHM and 11% increase of error.

We note that the alphabets in each case are selected according to the number of *ones* from the *Gset*. For better trade-off between filter quality and power dissipation and performance, 4-alphabets (1, 3, 5, 9) can be used. In this case results show a 13% power and 7% performance improvement when compared to CSHM-8 and 8% power and 31% delay improvement compared to CSAM with only 2% increase of chebychev error in filter frequency response. Fig. 4b shows the area reduction of CSHM as the size of *Aset* decreases and the corresponding power and delay improvement.

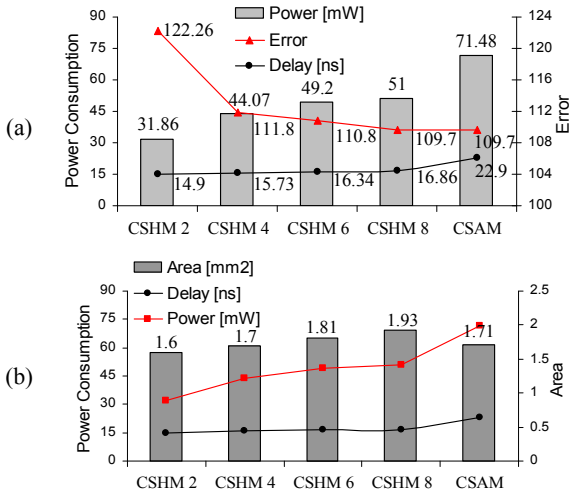

Fig. 4: Comparison of CSHM (with varying *Aset* size) and CSAM

6. CONCLUSION

A new approach to low-power filter design by constraining the synthesis of coefficients has been presented. By pre-specifying a few "good alphabets" we synthesize "modified coefficients" so that proper trade-off between computation/communication complexity and filter quality can be made. The number of alphabets representing/covering the coefficients can be reduced significantly and the coefficients are guaranteed to be near optimal satisfying the filter characteristics. We have implemented CSHM with reduced number of alphabets and 37% power reduction was shown over the CSHM that uses all the 8 alphabets and 55% over the CSAM based filter. Our approach offers flexibility in accommodating more implementation (or power) related constraints such as the number of ones in the coefficients. The idea presented in this paper can prove to be suitable for designing multiplierless DSP algorithms and implementing low power and high performance systems.

7. REFERENCES

[1] H. Samueli, "An Improved Search Algorithm for the Design of Multiplierless FIR Filters with Powers-of-Two Coefficients", *IEEE TCAS*, pp 1044-1047, July 1989.

[2] R. Hartley, "Optimization of canonic signed digit multipliers for filter design", *IEEE Int. Symp. Circuits and Systems*, pp. 1992–1995, June 1991.

[3] I. Park, et al., "Digital filter synthesis based on minimal signed digit representation", *IEEE DAC*, pp. 468-473, 2001.

[4] M.Potkonjak, et al., "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination", *IEEE TCAD* , 15(2):151-165, Feb 1996.

[5] M.D. Macleod, et al., "Multiplierless FIR filter dsign algorithms", *IEEE Signal Processing Letters,* vol 12, issue3, pp. 186-189, March 2005.

[6] J. Park, et al., "High performance FIR Filter Design Based on Sharing Multiplication", *IEEE TVLSI,* vol. 11, no. 2, April 2003.

[7] K. Muhammad, *Algorithmic and Architectural Techniques for Low Power Digital Signal Processing*, PhD thesis, Purdue University, 1999.

[8] J. G. Proakis, D. G. Manolakis, "*Digital Signal Processing: Principles, Algorithms and Applications*", McMillan Company, New York, 1992.

[9] M. K Dusan, "Design of Optimal Finite Wordlength FIR Digital Filters Using Integer Programming Techniques", *IEEE TASSP*, vol. ASSP-28, no. 3, June 1980.

[10] N.H.E. Weste, D. Harris, "*Principles of CMOS VLSI Design: A Systems Perspective*", 3rd edition, Addison Wesley, 2004.

[11]W.L.Winston, " *Introduction to Mathematical Programming Applications and Algorithms*", 2nd edition, ITP, 1995.

[12] Gams - A User's Guide, GAMS Development Corporation, 2006.