An Algorithm for VLSI Implementation of Highly Efficient Cubic-Polynomial Evaluation

Fan Mo, Yihua Zhang, Jun Yu and Qianling Zhang
ASIC & Systems State Key Lab, Fudan University, Shanghai, 200433, China
mo@ic.eecs.berkeley.edu, zhangyh@kali.com.cn, yujun@moboss.fudan.edu.cn, qlzhang@fudan.edu.cn

***Abstract* — In this paper, we present a novel cubic-polynomial evaluation algorithm. It is suitable for VLSI implementation and the computational cost is reduced to about 66% of the previously reported method.**

I. CBIC-POLYNOMIAL EVALUATION ALGORITHM

Cubic-polynomial evaluation is a commonly used method in measurement and instrumentation [1], [2]. Among the applications involving cubic-polynomial evaluation, there exist a lot of cases that the measurement is an iterative process, dual-integration analog-to-digital conversion for example. Direct evaluation after the measurement is of low efficiency, since the processing module is idle during the measurement. High computational efficiency can be achieved through iterative method.

An iterative cubic-polynomial algorithm was proposed by P.Mathias and L.Patnaik [3]. Their algorithm is based on the idea of systolic array and requires 3 operations at each step. In this paper, we propose an algorithm that employs only 2 operations at each step. So higher computational efficiency is achieved.

A general form of cubic-polynomial is:

$$z(x)=\sum_{i=0}^{3}a_i x^i = a_3x^3+a_2x^2+a_1x+a_0 \tag{1}$$

In [3], vector $P(x)$ is set up for the evaluation of the polynomial expression:

$$P(x)=[p_0(x),p_1(x),p_2(x),p_3(x)]^T \tag{2}$$

As shown in Fig.1 (a), from the initial vector $P(0)$:

$$P(0)=[p_0(0),p_1(0),p_2(0),p_3(0)]^T=\begin{bmatrix}a_0\\ a_1+a_2+a_3\\ 2a_2+6a_3\\ 6a_3\end{bmatrix} \tag{3}$$

iteratively execute

$$P(x+1)=A\cdot P(x) \qquad x=0,1,\ldots,M \tag{4}$$

until x reaches its final value M. The matrix A in (4) is:

$$A=\begin{bmatrix}1&1&0&0\\0&1&1&0\\0&0&1&1\\0&0&0&1\end{bmatrix} \tag{5}$$

The evaluating result at $x=M$ can be expressed as:

$$z(M)=p_0(M) \tag{6}$$

Notice that 3 addition operations are needed at each step. Hence, for a certain value M, the amount of operations is:

$$N=3M \tag{7}$$

In our algorithm, we attempt to reduce the number of operations at each step from 3 to 2. Vector $PP(x)$ is used:

$$PP(x)=[pp_0(x),pp_1(x),pp_2(x),pp_3(x)]^T \tag{8}$$

The iterative processes differ according to the parity of x:

$$\begin{cases}PP(x+1)=AA_o\cdot PP(x) & x=2d-1\\ PP(x+1)=AA_e\cdot PP(x) & x=2d\end{cases} \tag{9}$$

in which, d is an integer starting from 1. The matrixes for odd and even value of x are:

$$AA_o=\begin{bmatrix}1&1&0&0\\0&1&0&0\\0&0&1&1\\0&0&0&1\end{bmatrix},\quad AA_e=\begin{bmatrix}1&1&0&0\\0&1&1&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \tag{10}$$

Therefore, each step involves two additions as shown in Fig.1 (b). The evaluations of $pp_1(x)$ and $pp_2(x)$ are performed alternatively. The initial vector $PP(0)$ is chosen as:

$$PP(0)=[pp_0(0),pp_1(0),pp_2(0),pp_3]^T=\begin{bmatrix}a_0+3a_3\\ a_1+2a_2+4a_3\\ 4a_2\\ 24a_3\end{bmatrix} \tag{11}$$

Another remarkable difference between our algorithm and Mathias's is that errors are allowed to occur in $PP(x)$ at each iterative step. The error vector $\Delta(x)$ is defined as:

$$\begin{aligned}\Delta(x)&=PP(x)-P(x)\\ &=[\delta_0(x),\delta_1(x),\delta_2(x),\delta_3(x)]^T\end{aligned} \tag{12}$$

At step x, the error is:

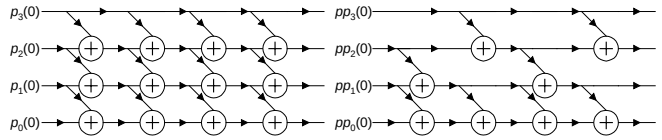
$$\begin{cases}\Delta(x)=\begin{bmatrix}0.5p_2(0)+(d-0.5)p_3(0)\\ -0.5p_2(0)-(d-1)p_3(0)\\ p_2(0)+(2d-1)p_3(0)\\ 3p_3(0)\end{bmatrix} & x=2d-1\\ \Delta(x)=\begin{bmatrix}0.5p_3(0)\\ 0.5p_2(0)+dp_3(0)\\ p_2(0)+(2d-2)p_3(0)\\ 3p_3(0)\end{bmatrix} & x=2d\end{cases} \tag{13}$$

It's easy to find:

$$\begin{cases}\delta_0(x)=pp_2(x)/4 & x=2d-1\\ \delta_0(x)=pp_3(x)/8 & x=2d\end{cases} \tag{14}$$

So the precise value of $z(x)$ can be derived though an additional compensation step which involves only one subtraction operation. The total amount of operations of our algorithm is:

$$NN=2M+1 \tag{15}$$



(a) Iterative process in [3] (b) Iterative process we propose

Fig. 1. Iterative process

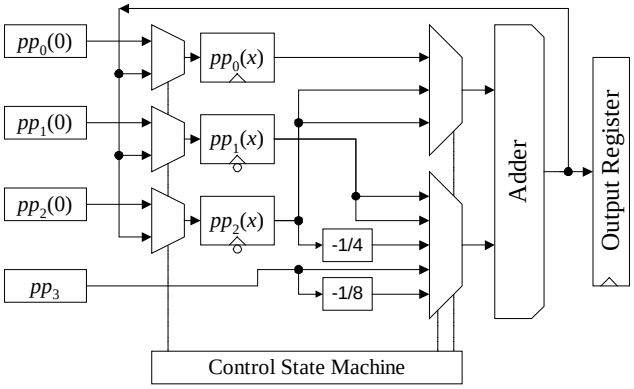


Fig. 2. Block diagram of the evaluation module

Roughly one third operations are saved as compared to (7).

II. Implementation of the Cubic-polynomial Evaluation Module

The structure of the cubic-polynomial evaluation module is shown in Fig.2. Three registers are employed to store intermediate values of $pp_0(x)$, $pp_1(x)$ and $pp_2(x)$. Since $pp_3(x)$ is constant, no register is assigned for it. When an evaluation round starts, the initial values are loaded into $pp_0(x)$, $pp_1(x)$ and $pp_2(x)$ register respectively. The module contains only one adder that all the operations defined in the algorithm share. During the iteration, the result of the new $pp_0(x)$ is stored at the positive clock edge, and the results of the new $pp_1(x)$ and $pp_2(x)$ that are alternatively generated are stored into the corresponding register at the negative clock edge. When the iteration is over, an additional compensating step is done, and the final value is stored into the output register. A state machine is used to control the data paths through multiplexers.

III. Experimental Results and Conclusion

A testing chip is fabricated in 0.6-micron double-metal double-poly CMOS technology, as shown in Fig.3. The 64-bit cubic-evaluation module highlighted in the micrograph is designed with standard cell library, occupying 0.45 mm^2. The achieved evaluation characteristics are given in Table I. Totally 160 bits are required for storage.

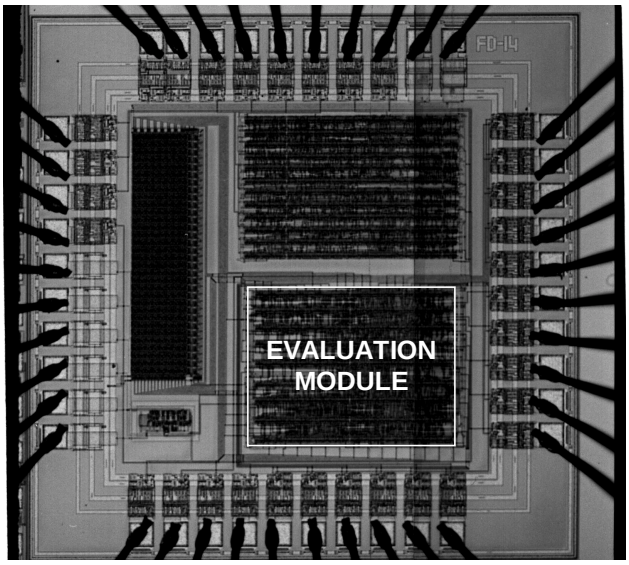


Fig. 3. Micrograph of the testing chip

TABLE I
EVALUATION CHARACTERISTICS

Polynomial Factor	Range min	Range max	Resolution
a_0	-1	1	1×10^{-19}
a_1	-3×10^{-5}	3×10^{-5}	1×10^{-19}
a_2	-4.6×10^{-10}	4.6×10^{-10}	1×10^{-19}
a_3	-7×10^{-15}	7×10^{-15}	1×10^{-19}

In this paper, we propose an algorithm for the cubic-polynomial evaluation. The iterative algorithm involves two addition operations at each step. The error incurred by the simplification of operations can be easily compensated with an additional subtraction. Hence, the total amount of operations is about one third less than that of previously reported method. Its feasibility has been verified by the experimental results.

References

[1] S. Garverick, K. Fujino, D. McGrath and R. Baertsch, "A programmable mixed signal ASIC for power metering", *IEEE J. Solid-State Circuits*, vol.26, no.12, Dec. 1991, pp. 2008-2016

[2] F. Mo, "The design of a DSP for the power-metering ASIC", Master Thesis, Fudan University, Jun. 1999

[3] P.C.Mathias and L.M.Patnaik, "Systolic evaluation of polynomial expressions", *IEEE Trans. Computers*, vol.39, no.5, May. 1990, pp. 653-665