Continuous-flow variable-length memoryless linear regression architecture

Mario Garrido Gálvez and J. Grajal

Linköping University Post Print

N.B.: When citing this work, cite the original article.

Original Publication:

Mario Garrido Gálvez and J. Grajal, Continuous-flow variable-length memoryless linear regression architecture, 2013, Electronics Letters, (49), 24, 1567-1568.

<http://dx.doi.org/10.1049/el.2013.2106>

Copyright: Institution of Engineering and Technology (IET)

<http://www.theiet.org/>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-103881>

Continuous-flow variable-length memoryless linear regression architecture

M. Garrido and J. Grajal

This letter presents a pipelined circuit to calculate the linear regression. The proposed circuit has the advantages that it can process a continuous flow of data, it does not need memory to store the input samples, and supports variable length that can be reconfigured in run time. The circuit is efficient in area, as it consists of a small number of adders, multipliers and dividers. These features make it very suitable for real-time applications, as well as for calculating the linear regression of a large number of samples.

Introduction: The linear regression [1] is one of the most important tools in statistical data analysis. It is used to determine the statistical relation between a dependent variable and an independent one, assuming that this relation can be modeled by a line.

The linear regression is used in many applications, ranging from data base processing [2] or augmented reality [3] to face recognition [4] or signal classification [5]. These applications are usually run in software [3, 4], and specific software programs to speed up the calculations of the linear regression have been proposed [6]. Nowadays, another alternative to handle a large number of computations is to resort to graphics processing units (GPUs) [2]. Finally, hardware platforms such as field programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) are a very suitable for real-time applications that demand both high throughput and low power consumption. Currently, more and more applications demand such real-time requirements [5]. For this reason, the design of efficient hardware architectures [8, 9] plays an important role in current and future applications.

Generally, the linear regression is calculated iteratively: A processor reads groups of data from memory, processes them and writes them back in memory until all the computations of the regression are carried out [7]. In these memory-based circuits, the input samples are first loaded to memory, then the linear regression is calculated iteratively and, finally, the results are read from memory. As new data cannot be stored in memory until all the computations have finished, this approach is not suitable for processing continuous flows of data. Furthermore, this approach requires to allocate all the inputs in memory. This leads to significant demands for memory when the linear regression is applied to large amounts of data.

Pipelined circuits [8, 9] are an alternative to memory-based ones. They support continuous data flow and reach high throughput and low latency. In spite of this, the use of pipelined circuits for the linear regression has hardly been considered. To the best of the authors' knowledge, the first pipelined linear regression was presented in [8]. Later, a pipelined linear regression that makes use of the embedded DSP blocks in FPGAs was proposed [9]. However, this approach is only suitable for a fixed number of samples and only if this number is a power of two.

This letter presents a novel pipelined architecture for the linear regression. The circuit has been specifically designed for a real-time signal classification system [5], but it can be easily adapted to other applications that demands real-time processing and few hardware resources. The proposed design has multiple advantages. First, the circuit does not require any memory, but just a few registers. This is very relevant for calculating the linear regression on large amounts of data and leads to significant savings with respect to memory-based approaches. Second, it can process a continuous flow of data. Third, it supports variable-length, which can be configured dynamically in run time. Fourth, the proposed circuit calculates all the parameters of the linear regression, including the error of the approximation, which is necessary for signal classification [5]. Finally, the design has been optimized in area by reducing the number of adders, multipliers and dividers.

The Linear Regression: The linear regression [1] is used to determine the relation between a dependent variable, Y , and an independent variable, X , based on a set of N pairs of samples, (X_i, Y_i) , where $i = 1, \dots, N$. The variables are supposed to be related by a line

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \quad (1)$$

where ϵ_i is the error of the i th sample.

The line that best fits the data is calculated by minimizing the mean square error (MSE). This provides b_0 and b_1 , which are the estimators of

Fig. 1 Proposed circuit for the linear regression. (a) Accumulators. (b) Main computations. (c) Dividers.

β_0 y β_1 for the y -intercept and the slope of the line, respectively:

$$b_0 = \frac{\sum Y_i - b_1 \sum X_i}{N} \quad (2)$$

$$b_1 = \frac{N \sum X_i Y_i - \sum X_i \sum Y_i}{N \sum (X_i)^2 - (\sum X_i)^2} \quad (3)$$

where the sums are defined for the interval $i = 1, \dots, N$.

Finally, the mean square error of the linear regression is calculated as

$$MSE = \frac{1}{N} \sum_{i=1}^N (Y_i - b_1 X_i - b_0)^2 \quad (4)$$

As a result, (2), (3) and (4) provide the values of the three parameters of interest, b_1 , b_0 and MSE respectively.

Proposed Architecture: The continuous-flow variable-length memoryless linear regression architecture is shown in Fig. 1. The circuit is divided into three blocks that calculate the accumulations, main computations and divisions, respectively. The first block in Fig. 1(a) calculates the summations

$$\begin{aligned} A &= \sum_{i=1}^N X_i & B &= \sum_{i=1}^N Y_i & C &= \sum_{i=1}^N X_i^2 \\ D &= \sum_{i=1}^N Y_i^2 & E &= \sum_{i=1}^N X_i Y_i & N &= \sum_{i=1}^N 1 \end{aligned} \quad (5)$$

This block only needs five registers, which are the only storage elements in the architecture. This is very little storage compared with the memory required in memory-based architectures, which needs to store all the input samples. These savings in memory are especially significant when calculating the linear regression on large amounts of data.

The second block in Fig. 1(b) calculates the main operations

$$\begin{aligned} F &= NE - AB \\ G &= BC - AE \\ H &= (NC - A^2)D + (2AE - BC)B - NE^2 \\ I &= J = NC - A^2 \\ K &= N(NC - A^2) \end{aligned} \quad (6)$$

For this block the architecture admits two options: fully pipelined and time-multiplexed. The fully pipelined architecture is the direct implementation of the operations in Fig. 1(b). The multiplication by 2 in Fig. 1(b) is carried out by the 1-bit shift represented by ($\ll 1$). This shift is hard wired and, therefore, does not need any hardware. Furthermore, the adders and multipliers are shared for different computations. For instance, the term $NC - A^2$ is reused to calculate H , I and K . This reduces the number of adders and multipliers in the circuit.

The time-multiplexed architecture takes into account the fact that the main operations in Fig. 1(b) must only be calculated once, just after the first stage of accumulators has processed the N input data. Therefore, the

Table 1: Register allocation procedure

REGISTER NUMBER	FIRST ITERATION	SECOND ITERATION
1	C	$NC - A^2$
2	N	$N(NC - A^2)$
3	D	$D(NC - A^2)$
4	E	NE^2
5	A	$2AE - BC$
6	B	$B(2AE - BC)$
7	NC	$D(NC - A^2) - NE^2$
8	A^2	F
9	AB	G
10	BC	H
11	AE	-
12	NE	-

Table 2: Components for fully pipelined/time-multiplexed regression

HARDWARE COMPONENT	MODULE			TOTAL COST
	Summations	Main	Divisions	
Adders	6/6	6/1	-	12/7
Multipliers	3/3	10/1	-	13/4
Dividers	-	-	3/1	3/1
Registers	6/6	-/12	-	6/18

operations in Fig. 1(b) can be multiplexed in time. By doing this, only one adder and one multiplier are needed, at the expense of a few extra registers. Table 1 shows the register allocation procedure. By writing partial results sequentially in these registers, the output results can be provided in two iterations. Note that by writing to the registers in order, the second iteration only overwrites registers with values that are not needed any more.

The third block in Fig. 1(c) calculates the divisions to obtain the parameters of the linear regression. The fully pipelined architecture uses the three dividers shown in Fig. 1(c), whereas the time-multiplexed approach uses only one divider. Furthermore, in applications where the outputs of the regression are compared to a threshold, such as [5], these dividers can be substituted by constant multipliers. This simplifies the hardware. For instance, in order to check if the error is bigger than a threshold value TH_{MSE} , the comparison $MSE = H/K \leq TH_{MSE}$ can be transformed into $H \leq TH_{MSE} \cdot K$. Note that the former requires a divider and the latter only needs a constant multiplier.

The total number of components for the fully pipelined and time-multiplexed architectures are summarized in Table 2. As explained before, the small number of components is achieved by sharing terms in the calculations and reusing components. Furthermore, the number of components is fixed for any N . This is a significant advantage for large N , where other designs need large memories.

The latency also benefits from the proposed design. The results are provided a short time after the last sample has been collected. Contrary to memory-based approaches, this latency is independent of N . This can be observed in Fig. 1, where the time to calculate the main operations and the divisions does not depend on N and is constant once the registers in the accumulator block have been updated with the last sample.

The circuit can be used for any length of the regression, N , and the length can be reconfigured in run time. The reason for this is that the circuit provides incremental results of the regression. Therefore, the parameters of the regression are obtained just by collecting the values at the outputs when N samples have arrived. The calculations are restarted by just resetting the registers of the accumulators in Fig. 1(a).

Finally, the circuit can process a continuous flow of data at a rate of one sample per clock cycle, which allows for high throughput. In continuous flow, the time-multiplexed version has the limitation that the computations cannot start before those of the previous regression have been obtained. This sets a limit to the minimum number of samples of the regression. However, if N is large enough, the accumulation stage can process new samples, while stages two and three finish the calculations on the previous regression. This guarantees continuous flow. As a result, the fully pipelined architecture is more suitable when the number of samples is small, and the time-multiplexed architecture is preferable for large N , as it reduces the hardware components and guarantees continuous flow.

Conclusion: A circuit for calculating the linear regression is proposed in this letter. The circuit supports continuous flow and variable length, which can be configured in run time. The circuit uses few hardware components

and removes the need of a memory for the samples. The circuit is suitable for calculating the linear regression in real time, especially when the number of samples is large.

Acknowledgment: M. Garrido was supported by ELLIIT, Linköping University, Sweden.

M. Garrido (*Linköping University, Linköping, Sweden*) and J. Grajal (*Universidad Politécnica de Madrid, Madrid, Spain*)

E-mail: mariog@isy.liu.se

References

- 1 Draper, N., and Smith, H.: Applied Regression Analysis (John Wiley and Sons, 1980)
- 2 Kulkarni, J., Sawant, A., and Inamdar, V.: 'Database processing by linear regression on GPU using CUDA', in *Proc. Int. Conf. Signal Process. Comm. Comput. Networking Technologies* (2011), pp. 20–23
- 3 Ma, D.Y., Chen, Y.M., Li, Q.M., Huang, C., Xu, S., and Zou, Y.B.: 'Registration with linear regression model in augmented reality', in *Proc. Int. Conf. Science Autom. Eng.*, volume 2 (2011), pp. 520–523
- 4 Naseem, I., Togneri, R., and Bennamoun, M.: 'Linear regression for face recognition', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010, **32**, (11), pp. 2106–2112
- 5 Grajal, J., Yeste-Ojeda, O., Sánchez, M., Garrido, M., and López-Vallejo, M.: 'Real time FPGA implementation of an modulation classifier for electronic warfare applications', in *Proc. Europ. Signal Process. Conf.* (2011), pp. 1514–1518
- 6 Fenton, O., Beardmore, A., and Lane, D.: 'Software regression facility', Patent US 20080127121 A1, 2008
- 7 Washizawa, T.: 'Regression analysis apparatus and method', Patent US 20070288199, 2007
- 8 Garrido, M., and Grajal, J.: 'Procedimiento y arquitectura de circuito en pipeline para el cálculo de la regresión lineal', Patent ES 2365883, 2011
- 9 Royer, P., Sánchez, M., López-Vallejo, M., and López, C.A.: 'Area-efficient linear regression architecture for real-time signal processing on FPGAs', in *Proc. Conf. Design Circuits Integrated Syst.* (2011)