Resource optimised reconfigurable modular parallel pipelined stochastic approximation-based self-tuning regulator architecture with reduced latency

Varghese Mathew Vaidyan¹, Ashok Shankar²

¹Department of Computer Engineering, King Khalid University, Asir, Abha 61411, Saudi Arabia
²Department of Electrical Engineering, National Institute of Technology, Calicut 673601, India
E-mail: varghese86@gmail.com

Published in The Journal of Engineering; Received on 22nd December 2014; Accepted on 17th August 2015

Abstract: Present self-tuning regulator architectures based on recursive least-square estimation are computationally expensive and require large amount of resources and time in generating the first control signal due to computational bottlenecks imposed by the calculations involved in estimation stage, different stages of matrix multiplications and the number of intermediate variables at each iteration and precludes its use in applications that have fast required response times and those which run on embedded computing platforms with low-power or low-cost requirements with constraints on resource usage. A salient feature of this study is that a new modular parallel pipelined stochastic approximation-based self-tuning regulator architecture which reduces the time required to generate the first control signal, reduces resource usage and reduces the number of intermediate variables is proposed. Fast matrix multiplication, pipelining and high-speed arithmetic function implementations were used for improving the performance. Results of implementation demonstrate that the proposed architecture has an improvement in control signal generation time by 38% and reduction in resource usage by 41% in terms of multipliers and 44.4% in terms of adders compared with the best existing related work, opening up new possibilities for the application of online embedded self-tuning regulators.

1 Introduction

A self-tuning regulator (STR) is an important adaptive-control strategy for many applications. There has been several efforts to translate innovation in STR algorithms into practical implementations [1–6] and were realised using microcomputers or microcontrollers, single or multiple digital signal processors (DSPs) and faced problems in terms of performance and accuracy of computation during implementation. Enhancement in processing speed could be done by using high-performance DSP processors or multiprocessor schemes, but their cost is high and exceeds the benefits they bring [7]. General-purpose hardware in computer handles different tasks with very different computing patterns and trades resource efficiency for multitasking. It has very limited variety of architecture. Providing just the necessary memory bandwidth to keep the arithmetic units busy all the time, perfectly synchronising independent computations, avoiding or minimising contention on the computing and memory resources and achieving high computational efficiency can be made possible by designing a custom computing data path and a custom memory subsystem matching the data path. Redundant speculative circuits which burn power unnecessarily can be avoided completely by developing computing hardware specifically for a task. A custom-optimised architecture will extend the applicability of adaptive control to systems with tight real-time requirements in terms of fast dynamic response. With the advent of field programmable gate array (FPGA), designers can develop a fully reconfigurable hardware architecture dedicated to the control algorithm. In applications where required real-time capabilities cannot be ensured by software solutions, reconfigurable computers are very useful [8–13]. The cost per implemented function of an FPGA is also very less even though it is more expensive initially. In addition, other main advantages of FPGA-based reconfigurable technology are its ceaseless increasing density along with design flexibility of software, but with time performance closer to that of application specific integrated circuits.

To overcome the shortcomings of existing approaches in the implementation of the STR architectures using FPGA were developed such as scalar-based direct algorithm mapping-STR (SBDAM-STR) and multi-stage matrix multiplication-STR (MMM-STR) [14–17]. The SBDAM-STR architecture uses huge amount of resources and precludes its use in applications which run on resource constrained platforms. MMM-STR architecture which uses recursive elast squares (RLS) identification in its estimation stage consumes large amount of time in generating the first control signal (79 ccs) and precludes its use in applications that could benefit considerably from its advantages, especially in those that have fast required response times. Moreover, many intermediate variables are involved in the approach. In addition, five matrix multiplications (which consume time) are involved in every iteration. Owing to the advent of reconfigurable architectures, a stochastic approximation approach-based STR is a promising candidate. In this paper, a parallel, modular and pipelined stochastic approximation-based STR is proposed which has reduced latency in control signal generation than the best existing related work along with minimum resource usage. Distinguishing features of the proposed architecture are:

- Reduction in latency of control signal generation.
- Minimum resource usage.
- Reduction in number of intermediate variables involved.
- Reduction in number of matrix multiplications involved from 5 down to 2.

The rest of this paper is organised as follows. In Section 2, details about design of a stochastic approximation-based STR is discussed. Section 3 presents design flow of stochastic approximation-based STR. Functional verification is presented in Section 4. In Section 5, hardware implementation is discussed. Results of implementation and discussion are given in Section 6. Section 7 summarises this work.

2 Stochastic approximation-based STR design

The schematic diagram of an FPGA-based stochastic STR is given in Fig. 1.

2.1 Stochastic approximation

Important part of STR is the estimation block. To reduce the error in estimation of parameters, stochastic approximation-based adaptation of model parameters can be used [18, 19]. Stochastic
approximation is of the form

\[
\hat{a}_{n+1} = \hat{a}_n - \frac{k}{n+1} \frac{(\hat{a}_n y_n - y_n) Y_n}{\|Y_n\|^2} \tag{1}
\]

where

\[
Y_n = [u_0 \ u_1 \ldots u_{n-m} - y_{n-1} - y_{n-2} \ldots - y_{n-k}],
\]

\[
\hat{a}_n = [a_0 \ a_1 \ldots a_{m}],
\]

mean-square estimation error \((\|\hat{a}_n\|^2)\) is bounded by a quantity \(x_n\) given by

\[
x_{n+1} = \xi_n x_n - \frac{L}{(n+1)^2}; \quad x_0 = (\|\hat{a}_n\|^2) \tag{2}
\]

for \(p - n \to \infty\) and with variations of \(a_n\) vanishing asymptotically at a rate \(n^{-\nu}\) where \(w > 1\) with the condition that joint density functions of \(\hat{a}_n\) and \(Y_n\) satisfy [19]

\[
P(\hat{a}_n | Y_n) = P(\hat{a}_n) + o((p - n)^{-1})), \tag{3}
\]

\[
P(Y_n | \hat{a}_n) = P(Y_n) + o((p - n)^{-1})) \tag{4}
\]

Estimator converges in the mean according to (2) and the asymptotic behaviour is given by the equations

\[
(\|\hat{a}_n\|^2) \approx \frac{2L}{2k\beta - (v - 1)n^{-\nu - 1}}, \quad \text{for} \quad v - 1 < 2k\beta \tag{5}
\]

\[
= 0(n^{-2k\beta}), \quad \text{for} \quad v - 1 \geq 2k\beta \tag{6}
\]

where \(\beta = \inf \beta_k, \beta_k\) denotes the smallest eigenvalue of positive-definite matrix \((Y_n Y_n^T / \|Y_n\|^2)\). For all \(n \geq N_0, \xi_n < 1\) where

\[
N_0 = \text{integral part} \left[\frac{k}{2}\right] \tag{7}
\]

that is, transient part in (4) monotonically converges if \(n \geq N_0\) [18]. \(k\) is selected as \(> (1/2)\) since \(v - 1\) is utmost 1. \((Y_n Y_n^T / \|Y_n\|^2)\) is considered as a positive-definite matrix where \(\lambda_1, \lambda_2, \ldots, \lambda_s\) are the eigenvalues and \(Y_n\) is \(s\)-dimensional. Then

\[
\text{tr} \left( \frac{Y_n Y_n^T}{\|Y_n\|^2} \right) = \lambda_1 + \lambda_2 + \ldots + \lambda_s. \tag{8}
\]

However

\[
\text{tr} \left( \frac{Y_n Y_n^T}{\|Y_n\|^2} \right), \quad \text{tr} \left( \frac{Y_n Y_n^T}{\|Y_n\|^2} \right) = 1. \tag{9}
\]

From (8) and (9), \(\lambda_1\) is chosen as \(1/\), a conservative estimate of \(\beta\), assuming all eigenvalues to be equal. \(k\) is chosen equal top number of unknown parameters. Start of the process is at \(n = N_0\) where (7) represents \(N_0\).

2.2 Minimum-degree pole placement algorithm

Let \(A\) and \(B\) which do not have any common factors denote polynomials in either the forward shift operator \(q\) or differential operator, \(p = d/dt\) with \(A\) being monic.

Then a process can be described by the single-input, single-output system [1]

\[
Ay(t) = Bu(t) + v(t) \tag{10}
\]

Then, a general linear controller is given by

\[
Ru(t) = Tu(t) - Sy(t), \tag{11}
\]

where \(u\) denotes the command signal and \(R, S\) and \(T\) are polynomials. A feed forward with the transfer operator \(Tu\) and a negative feedback with the transfer operator \(-SR\) is represented by the control law. By eliminating \(u\) between (10) and (11) the following relations for the closed-loop system can be obtained

\[
y(t) = \frac{BT}{AR + BS} u(t) + \frac{BR}{AR + BS} y(t), \tag{12}
\]

\[
u(t) = \frac{AT}{AR + BS} u(t) + \frac{BS}{AR + BS} y(t). \tag{13}
\]

Thus, the closed-loop characteristics polynomial is given by

\[
AR + BS = A_c. \tag{14}
\]

In (14) called Diophantine equation, polynomial \(A_c\) is a design parameter that gives the desired properties of the closed-loop system. \(A\) and \(B\) are assumed to not have common factors for the equation to have solution. Equation (14) determines polynomials \(R\) and \(S\) only. To determine polynomial \(T\) in the controller given by (11), the response for the command signal \(u_c\) is described by

\[
A_m y_m(t) = B_m u_c(t). \tag{15}
\]

From (12) and (13), it then follows that the following condition must hold

\[
\frac{BT}{AR + BS} = \frac{BT}{A_c} = \frac{B_m}{A_m}. \tag{16}
\]

For all command signals, if the error is made zero then the perfect model following can be achieved. Factorisation of polynomial \(B\) yields

\[
B = B^+ B^- \tag{17}
\]

where \(B^+\) and \(B^-\) correspond to monic polynomial which has stable zeros and can be cancelled by the controller as it is well damped, and poorly damped or unstable factors that cannot be cancelled, respectively.
$B^r$ is a factor of $B_m$ and then

$$B_m = B^rB_m'$$

(18)

$B^r$ is a factor of $A_c$. Furthermore, $A_m$ must also be a factor of $A_c$ as can be seen from (46). Then, closed-loop characteristic polynomial is given by

$$A_c = A_0A_mB^r.$$  

(19)

From (14) it can be derived that $B^r$ is a factor of $B$ and $A_c$ also divides $R$. It then follows that

$$R = R'B^r$$

(20)

and Diophantine equation, (14) becomes

$$AR' + B'S = A_0A_m = A'_c.$$  

(21)

From (16) to (19)

$$T = A_0B_m$$

(22)

Minimum-degree pole placement algorithm is described briefly below [1, 2, 14].

Data: Polynomials $A$ and $B$.

Specifications: Polynomials $A_m$, $B_m$ and $A_0$.

Compatibility conditions: $\text{deg} \ A_m = \text{deg} \ A$

$$\text{deg} \ B_m = \text{deg} \ B,$$

$$\text{deg} \ A_0 = \text{deg} \ A - \text{deg} \ B^r - 1,$$

$$B_m = B^rB_m'.$$

Consider a monic $B^r$. On factorisation, $B$ becomes $B=B^rB^{-r}$. For $\text{deg} \ S < \text{deg} \ A$ find the solution $R'$ and $S$ by using relation

$$AR' + B'S = A_0A_m$$

(23)

Control signal is computed from the law

$$Ru = Tu_c - Sy$$

(24)

where $R = R'B^r$ and $T = A_0B_m^r$.

Consider a special case where all zeros are cancelled. Then, $\text{deg} \ A_0 = \text{deg} \ A - \text{deg} \ B - 1$. Choose $B_m = A_m(1)q^{d_m}$. On factorisation of $B$, we get $B = b_0$, $B = B/(b_0)$. Moreover, $B_m = A_m(1)q^{d_m}/b_0$ and $A_c = B'A_c$. Diophantine equation in (23) can then be expressed as

$$AR' + B_0S = A'_c = A_0A_m$$

(25)

Here, $R'$ is the quotient and $b_0S$ is the remainder on dividing $A_0A_m$ by $A$.

Consider a discrete-time process expressed by pulse transfer function

$$H(q) = \frac{B(q)}{A(q)} = \frac{b_0q + b_1}{q^2 + a_1q + a_2}$$

(26)

where $a_1$, $a_2$, $b_0$ and $b_1$ are the process parameters [1, 2, 14]. They are estimated by the stochastic approximation algorithm [18].

Consider desired closed-loop system as

$$\frac{B_m(q)}{A_m(q)} = \frac{b_mq}{q^2 + a_mq + a_{m_2}}$$

(27)

where $q$ is the forward shift operator and $a_{m_1}$, $a_{m_2}$ and $b_{m_0}$ are the parameters of the closed-loop system. This model satisfies compatibility conditions. Factorise polynomial $B$ as follows

$$B^r(q) = q + \frac{b_1}{b_0}$$

(28)

$$B^r(q) = b_0$$

(29)

$$B_m'(q) = \frac{b_m(q)}{b_0}$$

(30)

The process being of second order, polynomials $R$, $S$ and $T$ are of first order. Thus, Polynomial $R$ is of degree zero. As the polynomial is monic, $R = 1$. As $B^r = 1$, from compatibility conditions it can be found that $\text{deg} \ A_0 = 0$. Choose

$$A_0(q) = 1.$$  

(31)

The Diophantine equation, (25) is then

$$(q^2 + a_1q + a_2) \times 1 + b_0(s_0q + s_1) = q^2 + a_{m_1}q + a_{m_2}.$$  

(32)

On equating coefficients of equal powers of $q$, we get

$$a_1 + b_0s_0 = a_{m_1},$$

(33)

$$a_2 + b_0s_1 = a_{m_2}. $$

(34)

![Fig. 2 Design flow of stochastic approximation-based STR](image-url)
If $b_0 \neq 0$. The solution is given as

$$s_0 = \frac{a_{m_1} - a_1}{b_0},$$  \hspace{1cm} (35)$$

$$s_1 = \frac{a_{m_2} - a_2}{b_0}$$  \hspace{1cm} (36)

Thus, the controller can be given by following polynomials

$$R(q) = B^+ = q + \frac{b_1}{b_0}$$  \hspace{1cm} (37)$$

$$S(q) = s_0 q + s_1$$  \hspace{1cm} (38)$$

$$T(q) = A_0 B_n + \frac{b_m q}{b_0}$$  \hspace{1cm} (39)$$

or

$$r_1 = \frac{b_1}{b_0}$$  \hspace{1cm} (40)$$

$$t_0 = \frac{b_m}{b_0}$$  \hspace{1cm} (41)$$

The controller polynomials (11) to implement the following control law can then be described as

$$u(t) + r_1 u(t - 1) = t_0 u_c(t) - s_0 y(t) - s_1 y(t - 1)$$  \hspace{1cm} (42)$$

3 Design flow of stochastic approximation-based STR

The flow of design algorithm for the stochastic approximation-based STR is shown in Fig. 2. The estimated error is updated and reduced at every iteration using the adaptation with stochastic approximation.

4 Functional verification

Functionality of the STR was verified in MATLAB, version 7.9.0.529 (R2009b). As shown in Fig. 3a, a square wave command signal of unit amplitude is used. Initially process output oscillates due to the estimation error as shown in Fig. 3b. As shown in Fig. 3c, estimated parameters converge to true parameters. Table 1 shows the estimated true parameters of the process. Fig. 3d shows the implemented controller tracks the command signal. Satisfactory performance of the STR sufficient for hardware design is obtained.

Fig. 3 Functional verification

\begin{itemize}
  \item[a] Command signal
  \item[b] Process output oscillations in the initial transient state
  \item[c] Estimation converges to true process parameters
  \item[d] Process output tracks the command signal
\end{itemize}
5 Hardware implementation

The parallel architecture design of STR incorporating MMM is described in this section.

5.1 Architectural design

5.1.1 Parallel pipelined modular architecture of the stochastic approximation-based STR algorithm: The global parallel STR architecture design of STR incorporating MMM is described in this section.

Table 1 Parameter estimates

| Parameter        | $b_0$  | $b_1$  | $a_1$  | $a_2$  |
|------------------|--------|--------|--------|--------|
| true plant       | 0.1065 | 0.0902 | -1.6065| 0.6065 |
| estimated plant  | 0.1064 | 0.0905 | -1.6055| 0.6051 |

Fig. 4 Global parallel architecture of stochastic STR

Fig. 5 Pipelined stochastic approximation-based STR architecture structure
was partitioned into three modules: stochastic estimation module \( \text{stoch	extunderscore est	extunderscore module} \), the controller design module \( \text{cntrl	extunderscore design	extunderscore module} \) and the control law module \( \text{cntrl	extunderscore law	extunderscore module} \), as shown in Fig. 4. Furthermore, divided into three sub-modules: namely, the \( n\text{\textunderscore module} \), the \( nrm\text{\textunderscore module} \) and the \( est\text{\textunderscore module} \).

Global architecture: The global architecture of STR is shown in Fig. 4 and as three major modules: \( \text{stoch	extunderscore est	extunderscore module}, \text{cntrl	extunderscore design	extunderscore module} \) and \( \text{cntrl	extunderscore law	extunderscore module} \). Process parameters estimated using \( \text{stoch	extunderscore est	extunderscore module} \) were loaded in \( \text{cntrl	extunderscore design	extunderscore module} \) and controller design parameters were calculated. The \( \text{cntrl	extunderscore law	extunderscore module} \) generates the control signal. Temporary registers were loaded by enabling signals \( en_{\text{en}\textunderscore 1-\text{en}\textunderscore 4} \) and write signals \( w_{\text{w}\textunderscore 1-\text{w}\textunderscore 4} \). Different parts were triggered by \( t_{\text{t}\textunderscore 1-\text{t}\textunderscore 6} \). After 57 ccs the control signal was generated. Then, with the pipeline full, at every rising edge of the clock, the control signal was generated. The design flow of the pipelined design is shown in Fig. 5; signals \( 1-6 \) correspond to \( t_{\text{t}\textunderscore 1-\text{t}\textunderscore 6} \).

\( \text{Stoch	extunderscore est	extunderscore module} \): Two matrix multiplications were used in \( n\text{\textunderscore module} \) which computes \( (\hat{\alpha}_{n}Y_{n} - y_{n}) \). Matrix multiplications involved in \( n\text{\textunderscore module} \) were done using MMM architecture. Most of the time consumed is by matrix computation involved in every

![Flowchart of the pipelined stochastic approximation architecture](image)

Fig. 6 Flowchart of the pipelined stochastic approximation architecture

![Different modules of the stochastic approximation architecture](image)

Fig. 7 Different modules of the stochastic approximation architecture

- a \( n\text{\textunderscore module} \)
- b \( nrm\text{\textunderscore module} \)
- c \( est\text{\textunderscore module} \)

| Table 2 Hardware requirements |
|-----------------------------|
| Module          | Multiplier | Adder | Divider |
|-----------------|------------|-------|---------|
| \( n\text{\textunderscore module} \) | 8          | 3     | –       |
| \( nrm\text{\textunderscore module} \) | 4          | 1     | 1       |
| \( est\text{\textunderscore module} \) | 1          | 2     | 1       |
| \( cntrl\textunderscore law\textunderscore module \) | 4          | 1     | –       |
| \( cntrl\textunderscore des\textunderscore module \) | –          | 2     | 4       |
| **Total**       | **17**     | **9** | **6**   |

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)

\[ \text{J Eng}, 2015, \text{Vol. 2015, Iss. 9, pp. 286–292} \]

\[ \text{doi: 10.1049/joe.2014.0348} \]
iteration of estimation stage (1). The generalised MMM architecture for multiplying \( n \) number of \( (N \times N) \) matrices denoted by processing element was used for reducing the computation time required for matrix multiplications [14]. \( n_{\text{module}} \) computes the norm. Data from \( n_{\text{module}} \) and \( n_{\text{module}} \) were loaded in \( est_{\text{module}} \) and the process parameters were estimated online by \( stoch_{\text{est}}_{\text{module}} \). The flowchart of the pipelined stochastic approximation architecture is shown in Fig. 6. Different modules of the stochastic approximation architecture are shown in Figs. 7a–c.

Data controller module: The most critical part in the architecture is the data controller module. At appropriate instants clock, all processes were triggered with respect to the clock. The data controller module is shown Fig. 8. Control signals were generated by counters \((c_1,\ldots,c_6)\). The address lines were generated by finite state machines. For example, \( n_{\text{module}} \), in Fig. 4, completed its task at the 20th clock instant and \( est_{\text{module}} \) was triggered at the same instant by the data controller. Otherwise, entire computation will be ineffective.

5.2 Hardware utilisation

Table 2 lists the hardware requirements.

6 Implementation results and discussion

Implementation results of the developed architecture are presented and discussed in this section.

6.1 FPGA implementation

The architecture was designed in VERILOG. Verification and simulation were done in Xilinx ISE simulator. The hardware was mapped onto a XILINX Virtex4 FPGA device. Fig. 9 shows the first control signal was generated after 57 ccs.

6.2 Performance comparison

The performance comparison was made in terms of the control signal generation time and resource usage. The equations governing the recursive least-square algorithm were decomposed to scalar equations and implemented directly with multipliers and adders in the SBDAP approach [15–17]. The algorithm is advantageous in applications which require very precise estimation dynamic-data range. However, it requires huge hardware resources and needs extra hardware cost [14]. The hardware requirements were reduced using the MMM-based RLS architecture [14]. However, the control signal generation time is large and also large amount of resources are used and precludes its use in systems which have fast dynamic response and which have to work on resource constrained platforms. In the proposed paper, control signal generation time is improved by 38% when compared with best existing related work [14] as shown in Table 3. A comparison of hardware requirements with different architectures are shown in Table 4, which reveals the hardware requirements have reduced by 41% in terms of multipliers, 62.5% in terms of adders and comparable number of dividers in the proposed approach compared with the best existing related work [14]. The hardware requirements have reduced by 2.5\( \times \) in terms of multipliers, 5\( \times \) in terms of adders and 3.3\( \times \) in terms of dividers in the proposed approach compared with [15–17]. The processing speed is compared in Table 5. In case of complex algorithms, software implementations are further slower, and two to three orders of magnitude better performance can be obtained using parallel and pipelined hardware implementations [20].

7 Conclusion

A novel FPGA-based parallel and pipelined STR architecture based on stochastic estimation with faster control signal generation and reduced resource usage is proposed to overcome the shortcoming of the STR algorithm in real-time online applications. Compared with the best related existing work, the time for control signal generation was reduced by 38%, the resource usage was reduced by 44.4% in terms of adders and 41% in terms of multipliers. Moreover, the number of matrix multiplications and the number of intermediate variables involved at each iteration were reduced. Fast matrix multiplication, pipelining and high-speed arithmetic function implementations were used for improving the performance. The proposed architecture also has high processing speed and can be implemented in applications that have fast required response times and which run on embedded computing platforms with low-power or low-cost requirements with constraints on resource usage.

8 References

[1] Aström K.J., Wittenmark B.: ‘On self-tuning regulators’, Automatica, 1973, 9, (2), pp. 185–199
[2] Clarke D.W., Gawthrop P.J.: ‘Self-tuning controller’, Proc. IEE, 1973, 120, (9), pp. 929–934
[3] Sheirah M.A., Malik O.P., Hope G.S.: ‘Self-tuning microprocessor universal controller’, IEEE Trans. Ind. Electron., 1982, IE-29, (1), pp. 31–38

J Eng, 2015, Vol. 2015, Iss. 9, pp. 285–292
doi: 10.1049/joe.2014.0348

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)
