J Inf Process Syst, Vol.10, No.3, pp.429~442, September 2014
http://dx.doi.org/10.3745/JIPS.04.0004

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

Fault Detection in the Semiconductor Etch Process Using the Seasonal Autoregressive Integrated Moving Average Modeling

Muhammad Zeeshan Arshad*, Javeria Muhammad Nawaz*, and Sang Jeen Hong*

Abstract—In this paper, we investigated the use of seasonal autoregressive integrated moving average (SARIMA) time series models for fault detection in semiconductor etch equipment data. The derivative dynamic time warping algorithm was employed for the synchronization of data. The models were generated using a set of data from healthy runs, and the established models were compared with the experimental runs to find the faulty runs. It has been shown that the SARIMA modeling for this data can detect faults in the etch tool data from the semiconductor industry with an accuracy of 80% and 90% using the parameter-wise error computation and the step-wise error computation, respectively. We found that SARIMA is useful to detect incipient faults in semiconductor fabrication.

Keywords—Autoregressive Integrated Moving Average, Dynamic Time Warping, Fault Detection, Seasonal Autoregressive Integrated Moving Average, Semiconductor Process, Time Series Modeling

1. INTRODUCTION

Over the past decade of continuous improvements being made to semiconductor technologies, the device feature size has been scaled down tremendously. As a result, the processes have become more and more complex, and this calls for a much tighter process control than before. The faults in the manufacturing processes must be detected with precision to reduce the loss in manufacturing. Semiconductor equipment data has been shown to be helpful for the detection of any abnormal variation in the process. This is because any fault occurring during the process appears as some variation in the equipment tool data collected for that process.

There has been active research conducted in the fault detection area by applying different statistical techniques to the tool data. Neural networks have been one of the most researched techniques for the purpose [1-3]. Principal component analysis (PCA) and its variants have been widely employed as well [4,5]. Control charts have also been used for simultaneous fault detection and classification [6]. The use of autoregressive moving average based time series modeling techniques have also been shown to be quite useful in this regard [7,8]. These works have shown that the analysis of tool data gives greater insight into the process variations and can be used to detect abnormalities in the process.

In this research, the seasonal autoregressive integrated moving average (SARIMA) time series models are investigated for the detection of faults in etch tool data. The derivative dynamic time

Manuscript received September 16, 2013; accepted February 13, 2014; onlinefirst August 25, 2014.
Corresponding Author: Sang Jeen Hong (samhong@mju.ac.kr)
* Department of Electronic Engineering, Myongji University, Yongin 449-728, Korea
(mzeeshanarshad@gmail.com, javeriamnawaz@gmail.com, samhong@mju.ac.kr)

warping (DDTW) algorithm was employed to synchronize the data before the modeling step. The rest of the paper has been divided into three sections. Section 2 gives details about data collection, preprocessing, and synchronization. Section 3 provides a theoretical overview of SARIMA models. Section 4 explains the implementation of the technique and the results are discussed in Section 5.

2. SEMICONDUCTOR EQUIPMENT DATA

2.1 Data Preprocessing

Two sets of experimental data from an etch process were collected. The first set consisted of data from 10 acceptable runs of the etch process, while the second set consisted of 10 experimental runs. The experimental runs were composed of three acceptable runs and seven faulty runs. The faults were induced in the process for these seven runs. A total of eleven steps constituted a process run. All of the experiments were performed on an Applied Materials' DPS-II Centura dielectric etcher. The acquired data consisted of 55 parameters, where each parameter corresponded to the measurement of a specific attribute of equipment.

Table 1. Types of faults induced in an experimental data set

Run no.	Fault induced
1	None
2	–0.5 mT from base pressure
3	+0.5 mT from base pressure
4	–1% MFC conversion shift
5	+1% MFC conversion shift
6	Source RF cable: loss simulation
7	None
8	Bias RF cable: power delivered
9	None
10	Added chamber leak by 1.3 mT/min

MFC=mass flow controller, RF=radio frequency.

Table 1 provides the description of the experimental set and the type of faults induced. Our goal was to successfully distinguish these faulty runs from the acceptable runs of the experimental set. However, not all of these parameters carry useful information. Therefore, the 15 most important parameters were selected by incorporating PCA. The first principal component was observed and the parameters that correspond to the largest coefficients in the first component were selected. The selected parameters are listed in Table 2. Each parameter in the data had a separate range of values because of the difference in measurement units. As a necessary step before proceeding with modeling, the data normalization was performed using the min-max normalization technique.

Table 2. Selected data parameters

Parameter	Name	Parameter	Name
1	Throttle gate valve current	9	RF probe phase
2	RF source forward	10	E-chuck voltage
3	RF matcher current 1	11	Flow splitter-flow 1
4	RF matcher current 2	12	Flow splitter-total flow
5	RF bias forward	13	Gas flow-12
6	RF bias shunt	14	RF probe Vp-p
7	RF probe voltage	15	RF probe DC bias
8	RF probe current		

RF=radio frequency.

2.2 Derivative Dynamic Time Warping

Once the data was normalized, it was then synchronized. The need for data synchronization arose because the total no of observations collected from each run were unequal. Although all process runs followed the same steps, this mismatch appeared due to the fact that each run did not take exactly the same time to complete. Moreover, sometimes the delays in the data acquisition system also added to this misalignment of data.

In order to synchronize the data, we incorporated the DDTW [9,10] algorithm. DDTW is a variation of the DTW algorithm, which has been widely used in different fields for pattern recognition [11]. DDTW follows the DTW algorithm except that the inputs are differentiated in it before further processing, which enhances the performance of synchronization. The algorithm compares two independent time sequences, $X=(x_1, x_2, \ldots, x_N)$, having $N \in \mathbf{N}$ number of data samples and $Y=(y_1, y_2, \ldots, y_M)$, having $M \in \mathbf{N}$ number of data samples and finds the right alignment between the two by warping them. The process involves the construction of an $N \times M$ local cost matrix $\mathbf{C}$, where each element is $C_{i,j}=d(X_i,Y_j)$. The distance function d finds the Euclidian distance $(X_i - Y_j)^2$ between the points X_i and Y_j.

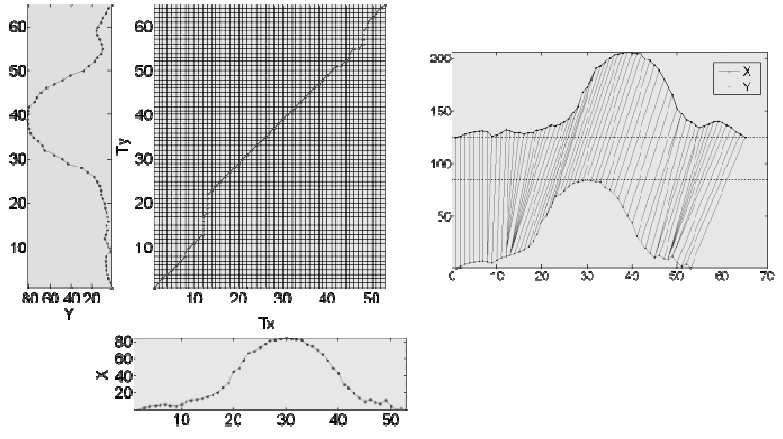


Fig. 1. The warping path and the alignment it returned for two sample time series X and Y with the unequal lengths of N=53 and M=65, respectively.

This matrix generates an alignment cost map of the two sequences. To find the right alignment, a path consisting of a set of coordinates across the map is computed, which is called the warping path *W*. Fig. 1 shows a DTW warping path as an example for two sample time sequences of *X* and *Y*.

The warping path defines a mapping between X and Y,

$$W = w_1, w_2, ..., w_k, ..., w_K \qquad \max(M,N) \le K < M+N-1 \tag{1}$$

The warping path must satisfy the following conditions:

1) boundary conditions: $w_1=(1,1)$ and $w_K=(M,N)$
2) continuity condition: for any $w_k=(a,b)$ and $w_{k-1}=(a',b')$, $a-a' \le 1$ and $b-b' \le 1$
3) monotonicity condition: for any $w_k=(a,b)$ and $w_{k-1}=(a',b')$, $a-a' \ge 0$ and $b-b' \ge 0$

There could be numerous paths that satisfy the above conditions, but there is a cost associated with each path and we are interested in the path with the lowest cost, which is called the optimal warping path.

i.e.,

$$\mathrm{DTW}(X,Y) = \min\left\{\frac{\sqrt{\sum_{k=1}^{K} wk}}{K}\right\}, \tag{2}$$

This can be found by using the dynamic programming based algorithm, which involves constructing the global cost matrix **D**, defined as [10]:

First row: $D_{1,j} = \sum_{k=1}^{j} C_{1,k}, \quad j \in [1, M]$

First column: $D_{i,1} = \sum_{k=1}^{i} C_{k,1}, \quad i \in [1, N]$

All other elements: $\mathbf{D}_{i,j} = \min\{\mathbf{D}_{i-1,j-1}, \mathbf{D}_{i-1,j}\mathbf{D}_{i,j-1}\} + \mathbf{C}_{i,j}$, where $i \in [1, N], j \in [1, M]$

After calculating the global cost matrix, the optimal warping path can be computed by tracking the path starting from $\mathbf{D}_{N,M}$ and by following the elements corresponding to the lowest cost at each step and then ending at $\mathbf{D}_{1,1}$. This path defines the alignment between the two sequences and can be used to create synchronized time sequences.

There are many modifications that have been made to DTW algorithm since its development, but all of them can be divided into two categories: symmetric and asymmetric [12]. Symmetric DTW algorithms give equal importance to both the input time sequence in such a way that the optimal path goes through all the points of both and hence results in two equal, synchronized time sequences. However, the total data samples of each of the synchronized sequences are greater than those before synchronization.

On the contrary, asymmetric DTW algorithms prefer one of the two input in such a way that the optimal path goes through all the points of the preferred sequence. However, it may skip

some of the points in the other input sequence. However, this results in synchronized sequences whose total data samples are equal to those of the preferred input sequence. Kassidas et al. [12] demonstrated the use of DTW for the synchronization of batch trajectories by combining both symmetric and asymmetric DTW algorithms. A variation of this methodology was used in this research. A run was fixed as the reference, and observations of all other runs from both data sets were synchronized with those of that run. In this way the number of observations for each of the runs was made equivalent to the number of observations of a reference run (2,411 observations).

3. TIME SERIES MODELING

3.1 ARIMA Models

ARIMA models combine the autoregressive (AR) and moving-average (MA) models to form a time series model. The general equation for an AR model is:

$$x_t = e_t + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} \tag{3}$$

where x_t is the value of a time series at time t, $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive coefficients, and e_t is the white noise error term at time t [13]. As evident from Eq. (1), the AR model expresses the current value of the time series in terms of p regressed past values of the series, where p is called the order of the AR model. Similarly a MA model has the general equation of:

$$x_t = e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q} \tag{4}$$

where x_t is the value of a time series at time t, $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average coefficients, and e_t is the white noise error term at time t [13]. Unlike AR models, MA models express the current value of time series in terms of q regressed past innovations of the series, where q is called the order of the MA model. The AR and MA models are both combined to form the ARMA model [13], which has the equation:

$$x_t = e_t + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q}. \tag{5}$$

ARMA models are used for stationary time series. However, real life time series are usually non-stationary and hence, prior to applying the autoregressive moving average model on such data, this non-stationary must be removed by differencing the series d times. For example, differencing a time series d=1 can be defined as $x_t^{d=1} = x_t - x_{t-1}$. An ARMA model with this differencing step added to it is termed as the ARIMA model. It is denoted by ARIMA(p,d,q), which summarizes the number of each of the three parameters, p, d, and q, that are necessary to define it. The general equation for ARIMA model can be written as:

$$x^d_t = e_t + \phi_1 x^d_{t-1} + \phi_2 x^d_{t-2} + \ldots + \phi_p x^d_{t-p} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q}. \tag{6}$$

An alternate and more flexible general notation for the ARIMA model using the backshift notation B is:

$$\Phi(B)\,(1\text{-}B)^d x_t = \Theta(B)\,e_t \tag{7}$$

where

$$\Phi(B) = (1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p)$$

$$\Theta(B) = (1 + \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q).$$

3.2 Seasonal ARIMA Models

A time series that contains repeated patterns in its data after specific period *S,* is said to have a seasonal behavior. A non-seasonal ARIMA model as presented in the last section, cannot model this kind of series. Therefore, an extension of ARIMA models, which are called SARIMA models, are used. In addition to the three parameters for defining an ARIMA model, the SARIMA model incorporates four more parameters in order to cope with the seasonal components of the series. These parameters are defined in the notation, SARIMA$(p,d,q)\times(P,D,Q)_s$, where p is the non-seasonal AR order, d is the non-seasonal differencing, q is the non-seasonal MA order, P is the seasonal AR order, D is the seasonal differencing, Q is the seasonal MA order, and S is the repeating seasonal pattern period [14,15]. The general equation for SARIMA model is:

$$\Phi(B)\,\Phi_s(B^s)\,(1\text{-}B)^d(1\text{-}B^s)^D x_t = \Theta(B)\,\Theta_s(B^s)\,e_t \tag{8}$$

where

$$\Phi(B) = (1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p)$$

$$\Phi_s(B^s) = (1 - \phi_{s1} B^s - \phi_{s2} B^{2s} - \ldots - \phi_{sp} B^{ps})$$

$$\Theta(B) = (1 + \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q)$$

$$\Theta_s(B^s) = (1 + \theta_{s1} B^s + \theta_{s2} B^{2s} + \ldots + \theta_{sq} B^{qs}).$$

Table 3. Using the autocorrelation function (ACF) and partial-autocorrelation (PACF) to identify ARMA(p,q)

	AR(p)	MA(q)	ARMA(p,q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

AR=autoregressive, MA=moving-average.

SARIMA models are widely used for the modeling and forecasting of seasonal data. In order to apply and use them, the following general steps are involved

Step 1. *Identification*: In this step the values of the parameters p, d, q, P, D, Q, and S are identified using the autocorrelation and partial-autocorrelation of the time series. The

process can be summarized as shown in Table 3 [13].

Step 2. *Estimation*: All the coefficients of the model specified in the first step are then estimated in this step. If the model performance is not adequate, the process is restarted from Step 1.

Step 3. *Forecasting*: The model is used to generate future forecasts.

4. IMPLEMENTATION

In order to apply the SARIMA model, the three-step methodology previously explained was followed. The healthy data set was used for developing the model. As for each parameter of the data being a separate time series, a separate model was developed, making the total number of models equal to fifteen.

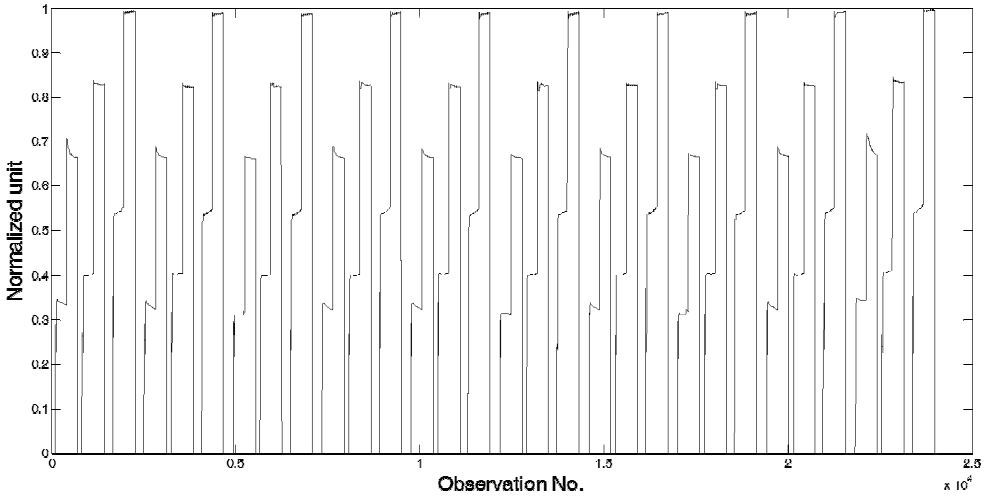


Fig. 2. Compiled sequence for parameter no. 8.

For each parameter, a sequence of time series of that parameter from each run was compiled together. So, as the pattern would repeat after every 2411 observations for each run, we found the seasonality parameter for the SARIMA model to be S=2411. Fig. 2 shows the compiled series for parameter no. 8, which contains the data from all ten runs of the healthy data set.

The first step is the identification of model parameters for defining the SARIMA model. This involves the use of the autocorrelation and partial-autocorrelation plots of the given series. The high value of initial lags in an autocorrelation plots suggests the need for differencing. The autocorrelation plot for the parameter no. 8 is given in Fig. 3, before and after the differencing of d=1. The rest of the parameters are identified using the autocorrelation and partial-autocorrelation plots of the differenced time series using the criteria mentioned in Table 3. For example, for the case of parameter no. 8, the plots are given in Fig. 4. The plots suggest a SARIMA $(0,1,1,0,1,1)_{2411}$ model for this particular parameter.

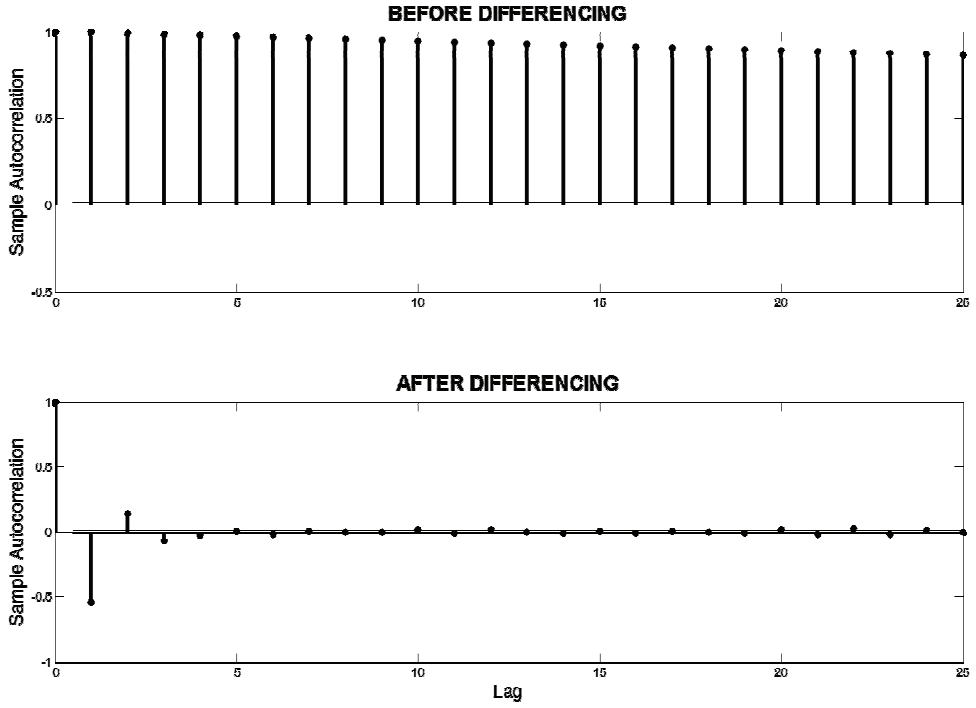


Fig. 3. Autocorrelation plots for parameter no. 8 before and after differencing.

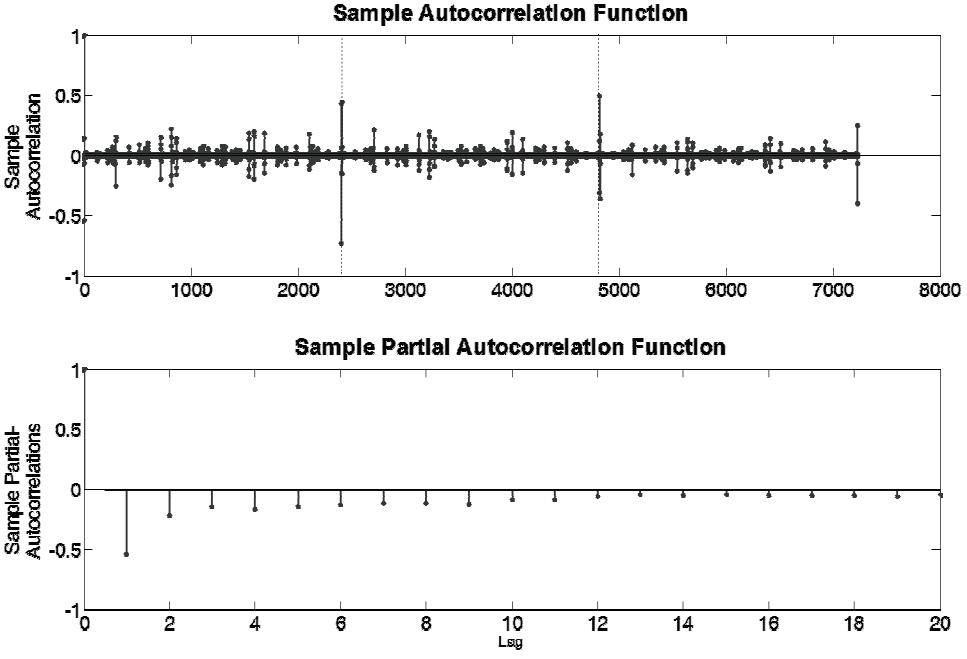


Fig. 4. Sample autocorrelation and partial autocorrelation for parameter no. 8.

For checking the performance of the model, the model was estimated (i.e., all of the coefficients of the model were estimated). The values of the estimated coefficients, standard error, and *t*-statistic were then observed. For a good model, the value of the coefficients should be statistically significant. Also, a low value of standard errors and the resulting high value of *t*-statistic indicate that the model is adequate for the modeled time series. If that turns out to not be the case, the model is reselected.

Table 4. Selected models for each of the data parameters

Parameter	SARIMA(p,d,q,P,D,Q)$_s$	Parameter	SARIMA(p,d,q,P,D,Q)$_s$
1	SARIMA(4,1,1,0,1,1)$_{2411}$	9	SARIMA(0,1,1,0,1,1)$_{2411}$
2	SARIMA(0,1,4,0,1,1)$_{2411}$	10	SARIMA(0,1,1,0,1,1)$_{2411}$
3	SARIMA(0,1,6,0,1,1)$_{2411}$	11	SARIMA(0,1,2,0,1,1)$_{2411}$
4	SARIMA(1,1,1,0,1,1)$_{2411}$	12	SARIMA(0,1,2,0,1,1)$_{2411}$
5	SARIMA(1,1,1,0,1,1)$_{2411}$	13	SARIMA(0,1,2,0,1,1)$_{2411}$
6	SARIMA(0,1,3,0,1,1)$_{2411}$	14	SARIMA(0,1,1,0,1,1)$_{2411}$
7	SARIMA(2,1,4,0,1,1)$_{2411}$	15	SARIMA(0,1,1,0,1,1)$_{2411}$
8	SARIMA(6,1,1,0,1,1)$_{2411}$		

SARIMA=seasonal autoregressive integrated moving average.

This procedure was followed for all parameters of the data and the selected models are summarized in Table 4. The established model gave us the fully specified model, which was ready for forecasting. In this step, the model is then used to forecast the next run using its mathematical equation, which uses the data from the healthy runs for the past values of the series.

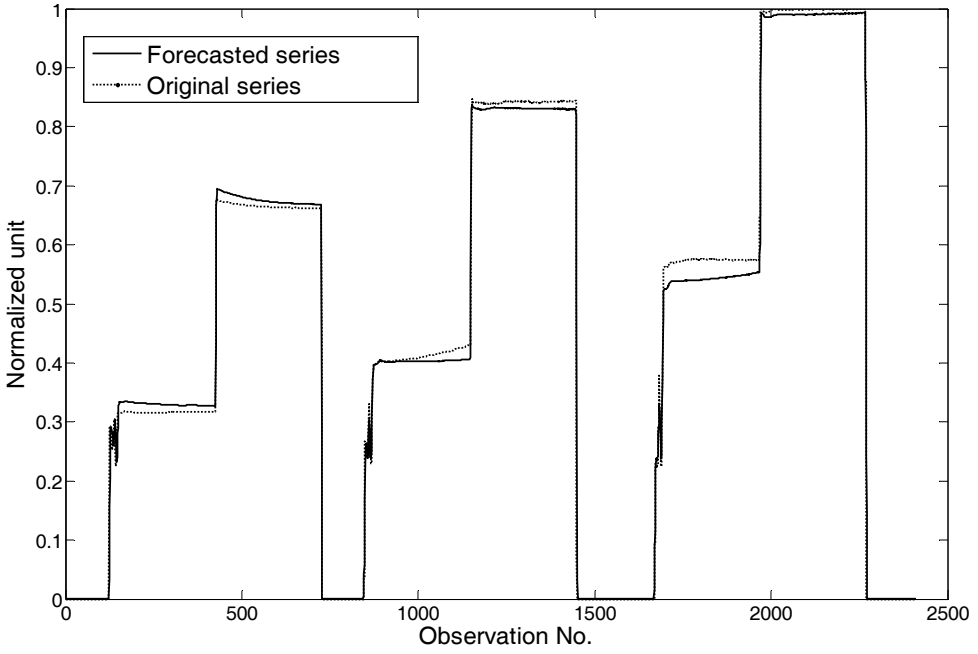


Fig. 5. Original and forecasted series for parameter no. 8 of experimental Run no. 10.

For each run in the experimental data set, the original series of all of the parameters were compared with the forecasted series for the corresponding parameters. Fig. 5 shows the comparison for parameter no. 8 in Run no. 10 of the experimental data set. The comparison yielded the root mean squared error (RMSE) for each parameter of each run. These results are summarized in Table 5. This table shows how much a parameter in each run varies from the model we developed for that parameter.

Table 5. Parameter-wise root mean squared error results for each run

	RUN1	RUN2	RUN3	RUN4	RUN5	RUN6	RUN7	RUN8	RUN9	RUN10
Par1	0.0070	0.0084	0.0076	0.0088	0.0101	0.0075	0.0078	0.0082	0.0106	0.0095
Par2	0.0376	0.0639	0.0316	0.0562	0.0398	0.0467	0.0598	0.0432	0.0359	0.0578
Par3	0.0611	0.0582	0.0538	0.0595	0.0566	0.0574	0.0582	0.0492	0.0610	0.0567
Par4	0.0605	0.0572	0.0554	0.0580	0.0556	0.0614	0.0570	0.0483	0.0609	0.0571
Par5	0.0443	0.0407	0.0335	0.0405	0.0383	0.0362	0.0569	0.0316	0.0441	0.0457
Par6	0.0279	0.0180	0.0283	0.0183	0.0269	0.0511	0.0461	0.0255	0.0196	0.0266
Par7	0.0279	0.0362	0.0317	0.0285	0.0262	0.0396	0.0338	0.0350	0.0320	0.0358
Par8	0.0166	0.0225	0.0247	0.0240	0.0236	0.0386	0.0237	0.0191	0.0156	0.0238
Par9	0.0362	0.0478	0.0528	0.0407	0.0404	0.0800	0.0511	0.0597	0.0384	0.0543
Par10	0.0244	0.0202	0.0167	0.0124	0.0151	0.0097	0.0183	0.0171	0.0144	0.0179
Par11	0.0154	0.0105	0.0219	0.0197	0.0103	0.0219	0.0253	0.0170	0.0154	0.0132
Par12	0.0180	0.0122	0.0237	0.0225	0.0104	0.0237	0.0278	0.0194	0.0178	0.0157
Par13	0.0070	0.0068	0.0153	0.0070	0.0070	0.0154	0.0154	0.0069	0.0070	0.0068
Par14	0.0300	0.0429	0.0403	0.0364	0.0324	0.0294	0.0449	0.0664	0.0342	0.0416
Par15	0.0321	0.0460	0.0433	0.0389	0.0345	0.0295	0.0478	0.0756	0.0351	0.0465

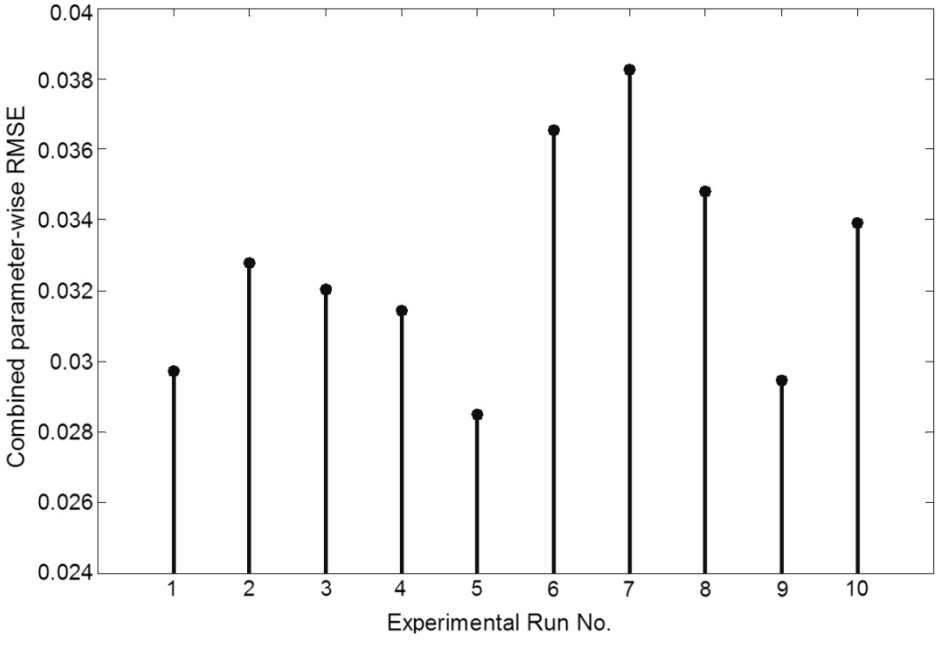


Fig. 6. The combined parameter-wise root mean squared error (RMSE) for each run.

In order to visualize the amount of errors in each run, the combined RMSE of each run was calculated as represented in Fig. 6. This can be used to quantify the amount of errors parameter-wise. However, in order to compare the runs from the perspective of errors in each of the eleven steps of the process, the step-wise RMSE was also presented in Table 6. This was done by comparing the forecasted series for each step of a parameter with the actual step in the original series of that parameter. This gives us insight into fault detection from another important perspective. The summary of the results for each run using this technique, which was done by calculating the combined stepwise RMSE for each run, is given in Fig. 7.

Table 6. Step-wise root mean squared error results for each run

	RUN1	RUN2	RUN3	RUN4	RUN5	RUN6	RUN7	RUN8	RUN9	RUN10
Step1	0.0130	0.0067	0.0067	0.0127	0.0102	0.0064	0.0138	0.0142	0.0127	0.0117
Step2	0.0440	0.0384	0.0253	0.0306	0.0316	0.0249	0.0511	0.0307	0.0366	0.0408
Step3	0.0169	0.0154	0.0157	0.0159	0.0151	0.0278	0.0183	0.0242	0.0130	0.0142
Step4	0.0396	0.0487	0.0299	0.0466	0.0389	0.0332	0.0355	0.0199	0.0389	0.0379
Step5	0.0226	0.0245	0.0332	0.0303	0.0291	0.0353	0.0297	0.0382	0.0305	0.0254
Step6	0.0131	0.0148	0.0151	0.0157	0.0149	0.0335	0.0151	0.0234	0.0120	0.0162
Step7	0.0308	0.0470	0.0421	0.0466	0.0464	0.0396	0.0451	0.0437	0.0294	0.0480
Step8	0.0049	0.0047	0.0050	0.0065	0.0047	0.0050	0.0055	0.0051	0.0049	0.0050
Step9	0.0260	0.0286	0.0304	0.0265	0.0214	0.0353	0.0320	0.0310	0.0246	0.0346
Step10	0.0119	0.0144	0.0151	0.0172	0.0133	0.0282	0.0145	0.0215	0.0133	0.0167
Step11	0.0354	0.0501	0.0569	0.0482	0.0386	0.0465	0.0611	0.0425	0.0416	0.0448

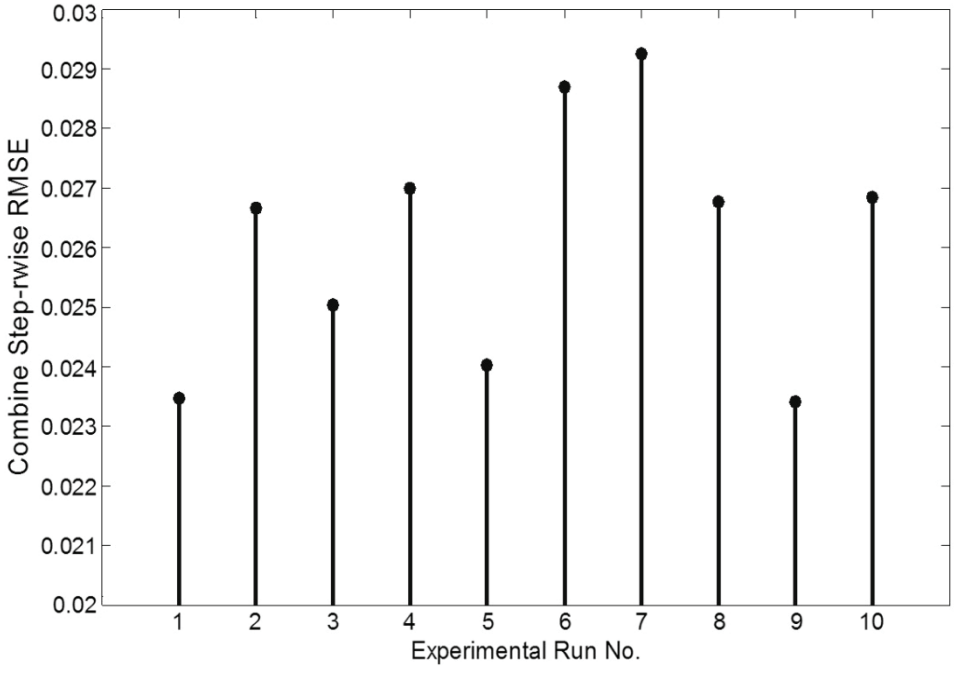


Fig. 7. The combined step-wise root mean squared error (RMSE) for each run.

5. RESULTS AND DISCUSSION

Observing the results summarized in chart in Fig. 6, except for Runs no. 5 and 7, all others came up with the expected results. Runs no. 1 and 9 were not induced with any fault, so they resulted in with very small errors. Similarly Runs no. 2–4, 6, 8, and 10 were the perturbed runs and clearly, they have been distinguished from the acceptable runs by resulting in a comparably higher error value. The exceptions were Runs no. 5 and 7, which returned with opposite results. Run no. 5 should have been detected with a large amount of error, while Run no. 7 should have had a small amount of error, but the results show otherwise. Since this technique resulted in eight successful detections, one missed detection and one false alarm; we can say that it can distinguish the faulty runs from the healthy runs with an accuracy of 80% by using the following equation for classification accuracy.

$$Accuracy = \frac{True\ Positives\ +\ True\ negatives}{True\ positives\ +\ False\ positives\ +\ False\ negatives\ +\ True\ negatives} \tag{9}$$

The second technique in which we calculated the step-wise error, as shown in Fig. 7, performed slightly better than the first one. Just like before, the acceptable Runs no. 1 and 9 showed the least amount of error. The fault-induced Runs no. 2-4, 6, 8, and 10 resulted in relatively higher error values. Similar to the first technique, Run no. 7 again came up with results that were contrary to expectations. Hence, we could argue that although there were no faults intentionally induced in Run no. 7, a fault might have occurred during the process of this run, which resulted in a high amount of error. However, unlike in the previous technique, the amount of error in Run no. 5 in these results was greater than the error in acceptable Runs no. 1 and 9. Considering this to be a near successful detection, we could use Eq. (9) to find the accuracy for this technique, which equaled 90%. A possible explanation for these slightly improved results might be that the faults induced in the process have an active effect on the data only for specific steps. Analyzing the error parameter-wise by combining the errors for the whole series of steps ended up in effectively blurring the error variance between individual steps.

It can be noticed in both charts for the combined parameter-wise and step-wise errors in the runs that the difference between the error values between the runs is very small. This is because the faults that were induced in the faulty experimental runs consisted of very little variation for evaluating the performance of the fault detection technique in detecting even the slightest of faults. Therefore, this small variation resulted in a very small difference existing between the faulty and acceptable runs.

It can be concluded that using SARIMA models can be useful for the detection of faults in semiconductor equipment data. By an improved selection of models, the presented technique could give even better results.

REFERENCES

[1] S. J. Hong, W. Y. Lim, T. Cheong, and G. S. May, "Fault detection and classification in plasma etch equipment for semiconductor manufacturing *e*-diagnostics," *IEEE Transactions on Semiconductor Manufacturing*, vol. 25, no. 1, pp. 83-93, Feb. 2012.

[2] M. D. Baker, C. D. Himmel, and G. S. May, "Time series modeling of reactive ion etching using

neural networks," *IEEE Transactions on Semiconductor Manufacturing*, vol. 8, no. 1, pp. 62-71, Feb. 1995.

[3] S. J. Hong, G. S. May, J. Yamartino, and A. Skumanich, "Automated fault detection and classification of etch systems using modular neural networks," *SPIE Proceedings*, vol. 5378, pp. 134-141, Apr. 2004.

[4] B. M. Wise, N. B. Gallagher, S. W. Butler, D. D. White, and G. G. Barna, "A comparison of principal component analysis, multiway principal component analysis, trilinear decomposition and parallel factor analysis for fault detection in a semiconductor etch process," *Journal of Chemometrics*, vol. 13, no. 3-4, pp. 379-396, May. 1999.

[5] G. Cherry, R. Good, and S. J. Qin, "Semiconductor process monitoring and fault detection with recursive multiway PCA based on a combined index," in *Proceedings of AEC/APC Symposium XIV*, Snowbird, UT, September 7-12, 2002.

[6] B. E. Goodlin, D. S. Boning, H. H. Sawin, and B. M. Wise, "Simultaneous fault detection and classification for semiconductor manufacturing tools," *Journal of the Electrochemical Society*, vol. 150, no. 12, Dec. 2003, pp. G778-G784.

[7] J. Zhao, L. Xu, and L. Liu, "Equipment fault forecasting based on ARMA model," in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, Harbin, China, August 5-8, 2007, pp. 3514-3518.

[8] C. J. Spanos, H. F. Guo, A. Miller, and J. Levine-Parrill, "Real-time statistical process control using tool data [semiconductor manufacturing]," *IEEE Transactions on Semiconductor Manufacturing*, vol. 5, no. 4, pp. 308-318, Nov. 1992.

[9] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proceedings of the 1st SIAM International Conference on Data Mining*, Chicago, IL, April 5-7, 2001.

[10] P. Senin, "Dynamic time warping algorithm review," 2008; http://www2.hawaii.edu/~senin/assets/papers/DTW-review2008draft.pdf.

[11] R. Bellman and R. Kalaba, "On adaptive control processes," *IRE Transactions on Automatic Control*, vol. 4, no. 2, pp. 1-9, Nov. 1959.

[12] A. Kassidas, J. F. MacGregor, and P. A. Taylor, "Synchronization of batch trajectories using dynamic time warping," *AIChE Journal*, vol. 44, no. 4, pp. 864-875, Apr. 1998.

[13] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and its Applications: With R Examples*, 3rd ed. New York, NY: Springer, 2011, pp. 83-162.

[14] S. Bisgaard and M. Kulahci, *Time Series Analysis and Forecasting by Example*. Hoboken, NJ: Wiley, 2011, pp. 83-92.

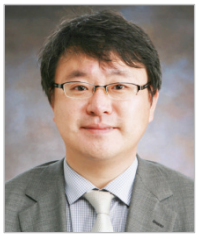
[15] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*. Hoboken, NJ: Wiley, 2008, pp. 231-285.

Muhammad Zeeshan Arshad

He received the B.S. degree in Electronics Engineering from Bahria University, Pakistan in 2009 and M.S. degree in Electronics Engineering from Myongji University, Korea in 2013. Currently he is undertaking a doctorate course as a member of the Microelectronics Manufacturing Technology Lab (MMTL) lab at Myongji University. Since 2013, Mr. Zeeshan has also been working as software developer at Rainbow Corporation (RBC). His research interests are real-time semiconductor process monitoring, data mining, data modeling and fault detection.

Javeria Muhammad Nawaz

She received her B.S. degree in Electronics Engineering from Bahria University, Pakistan in 2009 and M.S. degree in Electronics Engineering from Myongji University, Korea in 2013. Her research interest is semiconductor process data mining using various statistical analysis techniques.



Sang Jeen Hong

He received his B.S. in ECE from Myongji University, Korea in 1992 and his M.S. and Ph.D. degree in ECE from Georgia Institute of Technology, Atlanta, in 2001 and 2003. His research interests are APC with in-situ sensors and tool data driven FDC in volume semiconductor manufacturing. He is currently an associate professor in the Department of Electronic Engineering at Myongji University, where he serves as a director of Semiconductor Process Diagnosis Research Center (SPDRC). Dr. Hong is also serving a Director of Cooperative Center for Research Facilities at Myongji University. Dr. Hong was Editor-in-Chief of Transactions on Electrical and Electronic Materials from 2006-2007, and has been listed Marquis' Who's Who in the World from 2009-2013. He held JSPS Fellowship at Tohoku University, and has worked at IBM in East Fishkill as a process engineer.