itüdergisi/***d***
mühendislik
Cilt:9 Sayı 1, 121-133
Şubat 2010

Ayrık Olay Sistemlerinin kontrolü için bir modelleme ve gerçekleme yöntemi

İbrahim Tolga HASDEMİR*, Salman KURTULAN, Leyla GÖREN

İTÜ Fen Bilimleri Enstitüsü, Kontrol ve Otomasyon Mühendisliği Programı, 34469, Ayazağa, İstanbul

Özet

Sonlu durum makineleri ya da otomatlar, Ayrık Olay Sistemlerinin (AOS) analiz, tasarım ve kontrolüne yönelik formal yöntemlerin yer aldığı uygulamalarda yaygın olarak kullanılmaktadır. Ayrık olay sistemlerinin geribeslemeli kontrolü için kuramsal bir yapı tanımlayan Üstdenetim Kuramı ve bu kurama ilişkin uygulamalar buna örnek olarak verilebilir. Otomatın standart tanımının, tasarıma ilişkin ayrık olay sistem davranışını ifade edebilmesine rağmen, zamanlama ve sayma gibi bazı davranışları bu modelleme biçimi ile ifade etmek kolay değildir. Bu çalışmanın temel amacı, AOS davranışlarının, özel olarak kontrole yönelik davranış kurallarının tasarımını ve ifadesini mümkün kılan durum tabanlı bir modelleme biçiminin geliştirilmesi ve bu modele dayanarak tasarlanmış davranışın uygulanabilmesi için bir yöntemin elde edilmesidir. ZS-otomat olarak anılan yeni modelleme biçimi, uygulamada sıklıkla karşılaşılan zamanlama ve sayma davranışlarının durum gösterimi ile ifade edilmesini mümkün kılan Zamanlama ve Sayma Yapısına sahiptir. Modelin önemli bir özelliği, gerçekleme aşamasında kullanılan teknolojik araçlarda doğrudan uygulanabilecek yapısal bileşenler içermesidir. Bu çalışmada, modelleme biçiminin yanı sıra, bir gerçekleme yöntemi de tanıtılmaktadır. Gerçekleme yöntemi, ele alınan bir AOS için tasarlanmış kontrolörün ya da üstdenetleyicinin Programlanabilir Lojik Kontrolörlerle (PLC) gerçeklenmesine yönelik adımları sistematik olarak tanımlamaktadır. Yöntemin, tasarım davranışını yanlış olarak gerçeklemeye neden olan "çığ etkisi" adlı bir problem için çözüm oluşturduğu da gösterilmiştir. Geliştirilen yöntemle, tasarımda öngörülen zamanlama ve sayma davranışları, PLC'lerde kullanılan Zamanlayıcı ve Sayıcı bloklarını doğrudan kullanılmasını mümkün kılmaktadır. Sistematik olarak tanımlanan bu yöntem, programlanarak otomatik kod üretimini mümkün kılan bir yapıdadır.

Anahtar Kelimeler: *Ayrık olay sistemleri, gerçekleme, Programlanabilir Lojik Kontrolör.*

*Yazışmaların yapılacağı yazar: İbrahim Tolga HASEMİR. tolga.hasdemir@siemens.com; Tel: (216) 459 31 50.
Bu makale, birinci yazar tarafından İTÜ Fen Bilimleri Enstitüsü, Kontrol ve Otomasyon Mühendisliği programında tamamlanmış olan "Ayrık olay sistemlerinin tasarımı ve kontrolü için yeni bir gerçekleme ve otomatik kod üretme yöntemi " adlı doktora tezinden hazırlanmıştır. Makale metni 03.02.2009 tarihinde dergiye ulaşmış, 25.02.2009 tarihinde basım kararı alınmıştır. Makale ile ilgili tartışmalar 15.08.2010 tarihine kadar dergiye gönderilmelidir.

A modeling and realization method for the control of Discrete Event Systems

Extended abstract

When formal methods are applied in control design of Discrete Event Systems (DES), it becomes necessary to model the system behavior and specifications by a formalism such as automata or Petri nets. For example, the supervisory control theory (SCT) introduced by Ramadge and Wonham in 1987, uses formal languages to model system behavior and, specifications and formal languages are often expressed by automata. The synthesized controllers or supervisors are also represented by automata which are then needed to be realized by a programmable device. Standard definition of the automaton is capable of describing the DES behavior of a design; however some behaviors like timing and counting can not directly or easily be represented in the standard formalism.

Implementing a timing mechanism is necessary for discrete event control systems when a certain amount of delay is required to make a decision after an event occurs. A green traffic light, turning on after a certain time period following the red light could be an example. Likewise, a counting mechanism is applied if it is necessary to count the occurrence of a particular event for a number of times before issuing a control signal. For the case of manufacturing systems, packaging the products when a certain number of products are reached could be an example for the requirement of counting mechanism. While it is possible to implement the timing and counting requirements easily with the technological elements, formal definition of the standard automaton does not include such kind of mechanisms. However, it is possible to realize a time delay when utilizing formal methods. It is straightforward to define a time delay by assigning an event to indicate that a predetermined time has elapsed. However, in this case, a timing mechanism is required in the realization stage which is not a part of the formal structure. For counting an event for n times, n successive states in the automaton representation of the supervisor could be used, but in the general case, this may necessitate using large number of states. Therefore, a structure that is capable of representing the timing and counting behavior without using external mechanisms and excessive number of states is required. Another requirement regarding the implementation of formal supervisors is an output mechanism that would issue control signals (enabling or disabling signals for the example of Supervisory Control Theory) or events to drive the DES system being controlled. This requirement is generally met by employing automata with outputs for the representation of the supervisor.

When implementing a DES control strategy for discrete event systems, timing and counting behaviors are frequently applied by using predefined objects of the technological device which is used for the realization of control strategy. Programmable Logic Controllers (PLCs) have been used in industrial applications for more than 35 years, and in today's industrial control systems Programmable Logic Controllers (PLCs) are extensively used for realizing control strategies. PLCs make it possible to realize timing and counting behaviors easily by utilizing ready-made objects called timers and counters.

In this study, a formalism that enables designing and expressing DES behavior is developed, and a methodology that implements the designed behavior based on the introduced formalism is obtained. The new formalism, named TC-automaton, has a so called Timing and Counting Structure that enables the designer to assign timing and counting behaviors to the state based representation. An output function structure which enables outputting events depending on the states and/or on Timing-Counting Structure of the TC-automaton is also defined. TC-automaton is defined in such a way that, in the realization stage, it is possible to make use of the tools provided by the physical realization platform, i.e. PLCs. The implementation methodology introduced for the new formalism systematically defines the steps for realizing the designed controller or supervisor via PLCs. This systematic setting makes it possible to program the methodology, which is a step toward automatic code generation.

It is also shown that, the methodology resolves "avalanche effect" problem, which might be encountered due to a particular structure of the automaton representation of the control behavior. PLC programs obtained by utilizing the methodology are also modular in structure which enhances program readability.

Keywords: *Discrete Event Systems, realization, Programmable Logic Controllers.*

Giriş

Davranışı anlık olaylara bağlı olarak ayrık durum değişimleri ile ifade edilebilen sistemler Ayrık Olay Sistemleri-AOS (Discrete Event Systems-DES) adını alır. Ayrık olay sistemleri için Ramadge ve Wonham'ın önerdiği üstdenetim kuramı (Supervisory Control Theory), kapalı çevrim kontrolü ve üstdenetleyici (Supervisor) tasarımı için kuramsal bir yapı oluşturmaktadır (Ramadge ve Wonham, 1987). Üstdenetim kuramında kontrol edilecek sistem davranışı ve üstdenetleyiciler biçimsel diller ile ifade edilir. Biçimsel dillerin ifadesinde ise sıklıkla otomat gösterim (modelleme) biçiminden faydalanılmaktadır.

Otomat gösterim biçimi ile uygulamada sıklıkla karşılaşılan iki tür davranışın ifadesi oldukça güçtür. Bunlardan "zamanlama" olarak adlandırılacak davranış türü uygulamada belirli bir durumda yeni bir olayın oluşması ya da yeni bir duruma geçişmesi için beklenmesi gereken süre ile ilgilidir. Diğer davranış ise "sayma" olarak adlandırılacaktır ve yeni bir duruma geçişin belirli bir durumda, belirli bir olayın, belirli bir sayıda olması koşuluna karşılık gelir. Zamanlama davranışı kontrol uygulamalarında, farklı gereksinimlerle ortaya çıkabilir. Trafik ışıklarının belirli sürelerde yanıp sönmesi, endüstriyel üretim süreçlerinde bir işlemin başlaması için başka bir işlemin bitmesinden sonra belirli bir sürenin geçmesi, ya da bir test işleminin sonucuna karar vermek için belirli bir süre beklenmesi gibi örnekler verilebilir. Sayma davranışına ise bir kontrol davranışının belirli bir durumda, belirli bir olayın, belirli bir sayıda olması koşulu ile oluşturulduğu uygulamalarda gereksinim duyulur. Bir paketleme ünitesinde belirli sayıdaki ürün için bir paket oluşturulması örnek olarak verilebilir.

Zamanlama ve sayma davranışlarını standart otomat gösteriminde doğrudan ifade edecek bir yapı bulunmamaktadır. Ancak, uygulamada standart otomat gösterimi ile ifade edilen bir tasarım için zamanlama ve sayma davranışlarını gerçeklemek imkânsız değildir. Örneğin zamanlama davranışının gerçeklenmesi problemi için, öngörülen bekleme sürelerinin dolması birer olay olarak tanımlanabilir ve kontrolör modeli bu olayları içerecek şekilde oluşturulabilir. Ancak gerçekleme aşamasında bu sürelerin dolması ile ilgili olayı üretecek bir mekanizmanın da bulunması gerekir. Bu durumda zamanlama işlemi formal kontrolör modeli içinde değil, gerçekleme için kullanılan yöntemlerle ifade edilmiş olur. Sayma davranışını ifade etmek içinse standart otomat yapısında bir olay *n* kez sayılacaksa durumlar arası geçişlerin bu olayla sağlandığı *n* adet durum, sayma davranışını karşılamak için kullanılabilir. Ancak bu, en genel halde durum sayısı çok fazla olan otomatların elde edilmesine neden olabilir.

Uygulamada zamanlama ve sayma davranışının gerçeklenmesinde kullanılan teknolojik aracın özelliklerinden yararlanılmaktadır. Örneğin, Programlanabilir Lojik Kontrolörler (PLC) yapılarında zamanlayıcı ve sayıcı adı verilen hazır program bloklarını içermektedir (Kurtulan, 2007). Tasarımda formal yöntemlerin kullanılıp kullanılmamasından bağımsız olarak günümüzün AOS kontrol uygulamalarında PLC'ler yaygın olarak kullanılmakta ve zamanlama ve sayma davranışları bu hazır bloklarla gerçeklenmektedir.

Bu çalışmada, kontrolör ya da üst denetleyiciyi gerçekleme amaçlı olarak modellemek üzere zamanlama ve sayma davranışlarını yukarıda bahsedilen zorluklar olmaksızın yapısında barındıran, teknolojik elemanların sağladığı fonksiyonları doğrudan kullanmayı mümkün kılan bir otomat yapısı elde edilecektir.

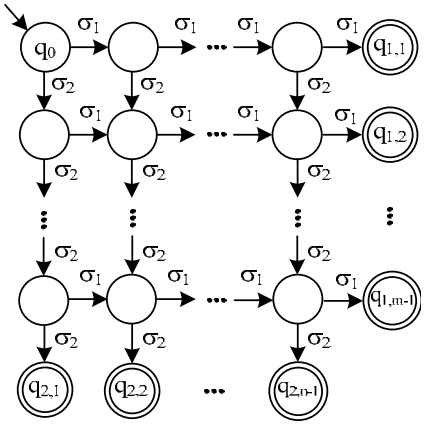
Zamanlama ve sayma davranışlarının yanında, belirli koşullara bağlı olarak belirli olayların üretilmesi önem taşır. Olay çıkışlarının üretilmesi farklı koşullara bağlı olabilir. Belirli bir kontrol davranışına karşılık gelen çıkışın, belirli bir duruma geçilmesi, bir durumda belirli süre beklenmesi ya da bir olayın belirli bir sayıda olmasına bağlı olarak üretilmesi gerekebilir. Sonuç olarak olay çıkışları otomatın bulunduğu duruma ya da durumla beraber zamanlama ve sayma davranışlarıyla doğrudan ilgilidir. Bu çalışmada elde edilecek otomat tanımında, durumlara ve zamanlama-sayma özelliklerine bağlı olarak olay üreten çıkış fonksiyonları da bulunmaktadır.

Formal tasarımla elde edilen ve durum gösterimi ile ifade edilen kontrolör ya da üstdenetleyicilerin gerçeklenmesi için, sonlu durum makinelerine karşılık gelecek PLC programlarının geliştirilmesine ilişkin yöntemlere gereksinim duyulur. Temel olarak, kontrolörlerin ya da sonlu durum makinelerinin gerçeklenmesi problemi PLC'nin kuramsal bir sonlu durum makinesi gibi davranmasını sağlayacak programının geliştirilmesinden ibarettir. Bu amaçla, PLC kodlarının geliştirilmesine yönelik yöntemler ve bu yöntemlerde karşılaşılabilecek problemler (Queiroz ve Cury, 2002; Fabian ve Hellgren, 1998) çalışmalarında incelenmiştir. Sonlu durum makinelerine karşılık gelen PLC programlarının oluşturulabilmesi için bilinen en basit yöntem, durumların ve olayların PLC bellek bitleri ile temsil edilmesine ve kurma (SET) ve silme (RESET) komutları ile durum geçiş fonksiyonlarının programlanmasına dayanır. Bu yöntem, kolay uygulanabilir olmasına rağmen, gerçeklenecek sonlu durum makinesinin bir tür yapısal özelliğine bağlı olarak davranışı beklenenden farklı olan yanlış çözümler verebilmektedir. Çığ etkisi (avalanche effect) adı verilen bu sorun için literatürde önerilen çözüm ise geliştirilen PLC programındaki komut sırasının değiştirilmesi gibi sistematik olmaktan uzak olan ve bazı durumlarda çözümü imkânsız kılan bir yönteme dayanmaktadır (Fabian ve Hellgren, 1998; Hasdemir vd., 2008).

Bu çalışmada çığ etkisi problemi için kesin ve sistematik bir çözüm oluşturan bir yöntem önerilmektedir. Yöntem, temel olarak zamanlama ve sayma davranışlarını modelleyen bir gösterim şekli için verilmektedir, ancak genel durumda standart otomat gösteriminin gerçeklenmesi için de kullanılabilir. Öncelikle zamanlama ve sayma davranışlarını modelleyen otomat gösterimi oluşturulacak, daha sonra bu gösterimden hareketle bir gerçekleme yöntemi önerilecektir.

Zamanlama ve Sayma davranışlarının ifadesi

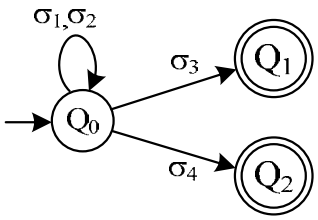
Genel durumda sayma davranışının standart otomat gösterimi ile ifadesi kabul edilemeyecek kadar çok sayıda durum kullanılmasına neden olabilir. Bu nedenle temel amaç sayma davranışını çoklu durum yerine az sayıda durumla ifade edecek bir yapı elde etmek olacaktır. Sonuçlar zamanlama davranışı için de kullanılacaktır. Öncelikle sayma işleminin standart otomat gösteriminden yola çıkılarak ifadesi elde edilecektir. Şekil 1'de bir q_0 durumundan itibaren σ_1 olayının *n*, σ_2 olayının *m* kez sayıldığını tespit eden bir otomat verilmektedir. İşaretli durumlar, ilgili olayın sayımının tamamlandığını göstermektedir.



Şekil 1. σ_1 *ve* σ_2 *olaylarının sayılmasını modelleyen standart otomat gösterimi*

Görüldüğü gibi sayılan olay sayısına bağlı olarak sayımı takip eden otomatın durum sayısı çarpımsal olarak büyümektedir. Bu problem için, davranış açısından eşdeğer olacak, durum sayısı az olan farklı otomatlar elde edilmeye çalışılacaktır. Şekil 1'deki otomatın durumları üç gruba ayrılabilir: İlk durumun da içinde bulunduğu işaretli olmayan durumlar bir gruba, işaretli durumlar hangi olayın sayımının tamamlandığını gösterecek şekilde diğer iki gruba karşılık getirilebilir. Buna göre, Şekil 1'deki işaretli olmayan durumların tamamı sayma işleminin devam ettiği eşdeğer tek bir durum olarak değerlendirilebilir. Sayma işlemi devam ettiği sürece σ_1 ve σ_2 olayları bu eşdeğer durumdan çıkışa neden olmayacağından bu olaylar özçevrim olayları olarak değerlendirilecektir. Başlangıç durumunun içinde bulunduğu eşdeğer durum Q_0 ile gösterilirse Şekil 2'deki gibi bir otomat elde edilebilir. σ_3 ve σ_4 olayları sırasıyla σ_1 ve

σ_2 olaylarının sayımının tamamlanmasına karşılık gelmektedir. σ_3 ve σ_4 olaylarının, ilgili sayma işlemi tamamlandığında otomatik olarak "üretildiği" varsayılmaktadır ve sayma olayı olarak adlandırılacaktır. Açıktır ki, standart otomat gösteriminde sayma özelliği ve sayıma karşılık bir olayın üretilmesi gibi bir işlev bulunmamaktadır. Bu çalışmada, otomat gösterimine bu özelliği kazandıracak bir yapı ile beraber, olayların üretilmesini mümkün kılan çıkış fonksiyonları tanımlanacaktır. Benzer bir özellik zamanlama davranışı için de geliştirilecektir. Tanımlanan otomat, hangi durumda hangi olayların kaç kez sayılması gerektiği bilgisini Sayma Yapısı adı verilen bir tanımla taşıyacaktır. Benzer şekilde hangi durumda ne kadar süre beklenmesi gerektiği bilgisi de Zamanlama Yapısı adlı bir tanımla verilecektir. Tanımında zamanlama ve sayma davranışlarını bulunduran bu otomat "ZS-yapılı otomat" ya da kısaca "ZS-otomat" olarak anılacaktır. Aşağıda ZS-otomatın formal tanımda kullanılacak vektör ağırlığı ve ağırlık kaybı tanımları verilmiştir.



Şekil 2. Sayma davranışına ilişkin eşdeğer otomat

Tanım 1. Bir **u** vektörünün ağırlığı vektörün sıfırdan farklı elemanlarının sayısına eşittir ve AG(**u**) ile gösterilir.

Tanım 2. $j,k \in \mathbf{N}$ olmak üzere her adımda sadece bir elemanının değeri değişen bir $(\mathbf{u}^1,\mathbf{u}^2,...,\mathbf{u}^k,...)$ vektör dizisinin k. adımında j. elemanının aldığı değer $u^k(j)$ ile gösterilsin. Bu vektör dizisinin j. elemanını için (k-1) ve k adımlarında

$$u^{k-1}(j) \neq 0$$
$$u^k(j) = 0 \qquad (1)$$

koşulları sağlanıyorsa (k–1). adımdan k. adımına geçişte **u** vektörünün ağırlığı bir azalır, yani

$$AG(\mathbf{u}^{k-1}) - AG(\mathbf{u}^k) = 1 \qquad (2)$$

koşulu sağlanır. Bu durumda k adımı ağırlık kayıp adımı, j tam sayısı da k adımındaki kayıp eleman olarak adlandırılır ve

$$AG_{\downarrow}(\mathbf{u}^k) = j \qquad (3)$$

ile gösterilir. Ağırlık kaybının olmadığı adımlar için kayıp eleman tanımsızdır.

ZS-otomat

ZS-otomat şu şekilde tanımlanmaktadır:

$$G_{ZS} = (Q, \Sigma, f, \Gamma, T, C, \Delta, q_0) \qquad (4)$$

Burada,

- Q sonlu durumlar kümesi,
- Σ sonlu olaylar kümesidir. $\Sigma = \Sigma_S \cup \Sigma_T \cup \Sigma_C$ şeklinde üç alt kümenin bileşimi olarak ifade edilir. $\Sigma_S = \{\sigma_1, \sigma_1, ..., \sigma_n\}$ sistem olayları kümesidir. Sistem olayları standart otomat tanımı ile verilen olaylar kümesindeki olaylara eşdeğerdir. $\Sigma_T = \{\sigma^T\}$, tek elemanlı bir kümedir ve elemanı zamanlama olayıdır. $\Sigma_C = \{\sigma_1^C, \sigma_2^C, ..., \sigma_n^C\}$ sayma olayları kümesidir. Bir ZS-otomatta sistem olayları sayısı kadar sayma olayı bulunur.
- $f : Q \times \Sigma \rightarrow Q$ durum geçiş fonksiyonudur ve özellikleri standart otomat tanımında verildiği gibidir.
- $\Gamma : Q \rightarrow 2^Q$ aktif olay fonksiyonu adını alır ve özellikleri standart otomat tanımında verildiği gibidir.
- T Zamanlama Yapısı (ZY) olarak adlandırılır ve

$$T = \{\tau_q,\ q \in Q\} \qquad (5)$$

şeklinde tanımlanır. Burada $\tau_q \in \mathbf{R}_{\geq 0}$, ZS-otomatın q durumunda üretilecek zamanlama olayına ilişkin gecikmeyi tanımlar.

- C Sayma Yapısı (SY) olarak adlandırılır ve

$$C = \{\mathbf{c}_q, \quad q \in Q\} \tag{6}$$

şeklinde tanımlanır. $\Sigma_S = \{\sigma_1, \sigma_1, ..., \sigma_n\}$ sistem olayları kümesi olmak üzere, $\mathbf{c}_q$, q durumuna ilişkin n boyutlu doğal sayılar vektörüdür. $\mathbf{c}_q$ vektörünün $\sigma_j \in \Sigma_S$ olayına karşılık gelen $c_q(j)$ elemanı, q durumunda bu olayının kaç kez sayılacağını belirlemektedir.

- Δ Çıkış Fonksiyonları kümesidir ve

$$\Delta = \{\delta, \delta^T, \delta^C\} \tag{7}$$

şeklinde üç bileşenle verilir. Bunlardan $\delta : Q \rightarrow \Sigma_S \cup \{\varepsilon\}$, duruma bağlı çıkış fonksiyonudur ve $q \in Q$ için,

$$\delta(q) = \begin{cases} \varepsilon & q \text{ aktif degil} \\ \exists! \sigma \in \{\Sigma_S \cup \{\varepsilon\}\} & q \text{ aktif} \end{cases} \tag{8}$$

şeklinde tanımlanır. Burada ε boş kelime, ya da uzunluğu sıfır olan kelimedir.

δ^T : $Q \times \mathbf{R}_{\geq 0} \rightarrow \Sigma_T \cup \{\varepsilon\}$ duruma ve ZY'ye bağlı çıkış fonksiyonudur ve

$$\delta^T(q,t) = \begin{cases} \varepsilon & q \text{ aktif degil veya } \tau_q = 0 \\ \varepsilon & q \text{ aktif ve } t \neq t_q + \tau_q \\ \sigma^T \in \Sigma_T & q \text{ aktif ve } t = t_q + \tau_q \end{cases} \tag{9}$$

şeklinde tanımlanır. Burada t zamana, t_q ise q durumunun aktif olduğu ana karşılık gelir. $\tau_q \in T$'dir.

δ^C : $Q \times \mathbf{N} \rightarrow \Sigma_C \cup \{\varepsilon\}$ duruma ve SY'ye bağlı çıkış fonksiyonu adını alır. δ^C, Tanım 1 ve Tanım 2 kullanılarak,

$$\delta^C(q,k) = \begin{cases} \varepsilon & q \text{ aktif degil} \vee \ \mathbf{c}_q = \mathbf{0} \\ \varepsilon & q \text{ aktif} \wedge \ AG_{\downarrow}(\mathbf{c}_q - \mathbf{n}_q^k) = AG_{\downarrow}(\mathbf{c}_q) \\ \sigma_j^C \in \Sigma_C & q \text{ aktif} \wedge \ AG_{\downarrow}(\mathbf{c}_q - \mathbf{n}_q^k) = AG_{\downarrow}(\mathbf{c}_q) - 1 \\ & AG_{\downarrow}(\mathbf{c}_q - \mathbf{n}_q^k) = j \end{cases}$$

şeklinde tanımlanır. $\mathbf{c}_q \in C$'dir. Sayma adımı olarak adlandırılan k değişkeni q durumu aktif olduğu an sıfır değerini alır ve durum değişikliğine neden olmayan her olayda (özçevrim olaylarının meydana gelmesiyle) içeriği bir artar. $\mathbf{n}_q^k$ vektörünün j. elemanı $n_q^k(j)$, $\Sigma_S = \{\sigma_1, \sigma_1, ..., \sigma_n\}$ sistem olayları kümesindeki bir σ_j olayına karşılık gelir ve olayın oluş sayısıyla ilgili bilgiyi taşır. $\mathbf{n}_q^k$, elemanları üzerinden aşağıdaki şekilde tanımlanır:

$$n_q^0(j) = 0,$$
$$n_q^{k+1}(j) = \begin{cases} n_q^k(j) + 1 & f(q, \sigma^k) = q, \ \ \sigma^k = \sigma_j \\ n_q^k(j) & f(q, \sigma^k) = q, \ \ \sigma^k \neq \sigma_j \end{cases} \tag{10}$$

Burada σ^k, k. adımdan (k+1). adıma geçişe neden olan olaydır.

- q_0 başlangıç durumudur.

Görüldüğü gibi ZS-otomat, standart otomatın özelliklerine ek olarak zamanlama, sayma ve olay üretme özelliklerine sahiptir. Bu anlamda standart otomatın, burada tanımlanan ZS-otomatın özel bir durumu olduğu söylenebilir. Gerçekten de $\forall q \in Q$ için $\tau_q = 0$, $\mathbf{c}_q = \mathbf{0}$ ve $\delta(q) = \varepsilon$ sağlanıyorsa ZS-otomat davranışı, standart otomat davranışı ile aynı olacaktır.

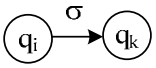
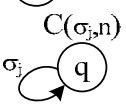
ZS-otomatta, zamanlama ve sayma yapıları durumlara ilişkilendirilmiş olarak tanımlanmaktadır. Benzer şekilde çıkış fonksiyonları da durumlarla ilişkilendirilmiştir. Bu, tasarımda ve gerçeklemede büyük esneklik sağlamaktadır. Tasarım aşamasında zamanlama davranışı, olaylardan bağımsız olarak belirli durumlar için öngörülür ve ZY ile bu durumlara belirli gecikme süreleri atanır. ZY'ye bağlı çıkış fonksiyonu ise ZY tarafından gecikme atanan durumlarda, ge-

cikme süresi sonunda zamanlama olayının üretileceğini garantilemektedir. Sayma davranışı ile ilgili olarak, belirli bir durum için, hangi olay ya da olayların o durumda kaç kez sayılacağı SY tarafından belirlenir. SY'ye bağlı çıkış fonksiyonu ise ilgili olayın sayımı tamamlandığında sayımın tamamlanmasına karşılık gelen sayma olayının üretilmesini garantilemektedir. Tasarımcı, gecikme süresinin, ya da ilgili sayımın tamamlanmasından sonraki kontrol davranışının ne olacağını, zamanlama ve sayma olaylarını kullanarak kolaylıkla belirleyebilir.

Görüldüğü gibi tasarım aşamasında ZY ve SY ile durumlara bir tür "özellik" kazandırılarak zamanlama ve sayma davranışlarının modellenmesi kolaylaştırılmaktadır. Bu davranışların durumlarla ilişkili olarak verilmesi gerçekleme aşaması için de avantaj sağlamaktadır. Örneğin PLC'ler için, hem gerçeklenme, hem de PLC programının okunabilirliğini kolaylaştırmaktadır.

Tablo 1'de, ZS-otomatın durum diyagramı gösterimindeki yapıtaşlarını ve bunların tanımında verilen karşılıklarını göstermektedir.

Tablo 1. Durum diyagramı gösteriminde ZS-otomatın yapıtaşları

Gösterimi	Karşılığı
	$q_0 = q$
	$f(q_i, \sigma) = q_k$
	$\delta(q) = \sigma$
	$\delta^T(q,t),\ \tau_q = t'$
	$\delta^C(q,k),\ c_q(j) = n$
	$\delta(q) = \varepsilon,\ \tau_q = 0,\ \mathbf{c}_q = \mathbf{0}$

Örnek 1. Bu örnekte bileşenleri aşağıda verilen $G_{ZS} = (Q, \Sigma, f, \Gamma, T, C, \Delta, q_0)$ ZS-otomatı ele alınacak ve ilgili durum diyagramı gösterimi elde edilecektir.

Durum ve olay kümeleri:

$$Q = \{q_1, q_2, ..., q_5\}$$

$$\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma^T, \sigma_1^C, \sigma_2^C, \sigma_3^C, \sigma_4^C\}$$

Durum geçiş fonksiyonu:

$$f(q_1, \sigma_1) = q_2,\ f(q_2, \sigma_1) = q_5,\ f(q_2, \sigma_2) = q_2,$$

$$f(q_2, \sigma_3) = q_2,\ f(q_2, \sigma^T) = q_1,\ f(q_2, \sigma_2^C) = q_4,$$

$$f(q_2, \sigma_3^C) = q_3,\ f(q_3, \sigma^T) = q_5,\ f(q_4, \sigma_4) = q_5,$$

$$f(q_5, \sigma_4) = q_2$$

Zamanlama Yapısı:

$$T = \{\tau_{q_1}, \tau_{q_2}, ..., \tau_{q_5}\} \text{ ve } \tau_{q_1} = \tau_{q_4} = \tau_{q_5} = 0;$$

$$\tau_{q_2} = 6.6,\ \tau_{q_3} = 2.3$$

Sayma Yapısı:

$$C = \{\mathbf{c}_{q_1}, \mathbf{c}_{q_2}, ..., \mathbf{c}_{q_5}\} \text{ ve}$$

$$\mathbf{c}_{q_1} = \mathbf{c}_{q_3} = \mathbf{c}_{q_4} = \mathbf{c}_{q_5} = [0 \quad 0 \quad 0 \quad 0];$$
$$\mathbf{c}_{q_2} = [0 \quad 4 \quad 2 \quad 0]$$

Çıkış fonksiyonları:

$$\Delta = \{\delta, \delta^T, \delta^C\}$$

$$\delta(q) = \begin{cases} \varepsilon & q \neq q_4 \\ \sigma_4 & q = q_4 \end{cases}$$

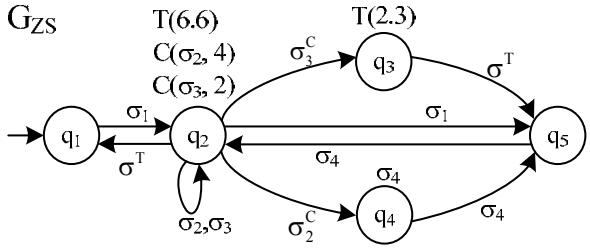
$$\delta^T(q,t) = \begin{cases} \varepsilon & q_2 \text{ ya da } q_3 \text{ aktif degil} \\ \varepsilon & q_i \text{ aktif ve } t \neq t_{q_i} + \tau_{q_i}\ (i = 2,3) \\ \sigma^T \in \Sigma_T & q_i \text{ aktif ve } t = t_{q_i} + \tau_{q_i}\ (i = 2,3) \end{cases}$$

$$\delta^C(q,k) = \begin{cases} \varepsilon & q_2 \text{ aktif degil} \\ \varepsilon & q_2 \text{ aktif degil} \\ \sigma_j^C \in \Sigma_C & q_2 \text{ aktif ve } AG(\mathbf{c}_q - \mathbf{n}_q^k) = AG(\mathbf{c}_q) - 1, \\ & AG_{\downarrow}(\mathbf{c}_q - \mathbf{n}_q^k) = j,\ (j = 2,3) \end{cases}$$

Başlangıç durumu:

$$q_0 = q_1$$

Bu ZS-otomatın durum diyagramı gösterimi Tablo 1 ile Şekil 3'teki gibi elde edilebilir.



Şekil 3. Örnek 1'de tanımlanan otomat

Şekil 3'teki ZS-otomatın örnek bir davranışı şu şekilde verilebilir: Başlangıçta q_1 durumunda olan ZS-otomat σ_1 olayı ile q_2 durumuna geçer. q_2 durumu için T zaman yapısı ile 6.6 saniyelik bir gecikme süresi, C sayma yapısı ile σ_2'nin 4, σ_3'ün 2 kez sayılacağı tanımlanmıştır. q_2 aktifken art arda σ_2 ve σ_3 olaylarının olması ile sayma adımı k'nın değeri 2, (10) ifadesi doğrultusunda $\mathbf{n}_{q_2}$ sayma vektörünün içeriği [0 1 1 0] olur. Ancak σ_2 ve σ_3 için tanımlanan sayma değerlerine ulaşılmadığından herhangi bir sayma olayı üretilmemektedir. q_2 aktif olduktan 6.6 saniye sonra ilgili çıkış fonksiyonu tarafından (9) ifadesi gereği σ^T zamanlama olayı üretilir ve otomat tekrar q_1 durumuna geçer. q_1 durumundayken σ_1 ile tekrar q_2 durumuna geçilir. q_2 durumunda σ_3 olayı ile k=1 değerini alır. σ_3'ün ikinci kez olması ile k=2 olur ve q_2 durumuna ilişkin sayma vektörünün (k=2 için $\mathbf{n}^2_{q_2}$) içeriği anlık olarak [0 0 2 0] olur. Bu ise ($\mathbf{c}_{q_2} - \mathbf{n}^2_{q_2}$) vektörünün ağırlık kaybetmesine ve δ^C fonksiyonunun σ^C_3 sayma olayını üretilmesine neden olur. σ^C_3 ile q_3 durumuna geçilir ve k sayma adımı sıfırlanır.

Otomat q_3'e geçtikten 2.3 saniye sonra duruma ve ZY'ye bağlı çıkış fonksiyonu tarafından (δ^T) σ^T üretilir. Bu olayla otomat q_5 durumuna geçer. Bu durumda σ_4 olayının olması G_{ZS}'nin tekrar q_2 durumuna geçmesine neden olur. q_2'de ZS yapısı ile σ_2'nin 4 kez sayılacağını tanımlamaktadır. σ_2 olayı 6.6 saniyeden kısa sürede 4 kez meydana gelirse ZS-otomat q_4'e geçecektir. q_4 durumunda, duruma bağlı çıkış fonksiyonu tarafından σ_4 olayının üretileceği tanımlanmaktadır. Dolayısıyla bu duruma geçilir geçilmez σ_4 üretilir, bu ise ZS-otomatın q_5'e geçmesine neden olur.

ZS-otomatın çalışmasına, bu şekilde örnek olaylarla devam edilebilir.

ZS-otomat için PLC ile gerçekleme yöntemi

ZS-otomat, durum diyagramı gösterimine ek olarak, tanımında ZS-Yapısı ve çıkış fonksiyonlarını da bulundurmaktadır. Aşağıda ZS-otomat için tanımlanacak yöntem durum geçişleri ve ilk durumun gerçeklenmesine ek olarak, ZS-Yapısı ve çıkış fonksiyonlarının gerçeklenmesini içerir. Durum geçişleri ve ilk durumun gerçeklenmesi için verilecek tanımlar kolaylıkla standart otomat için uyarlanabilir. Yöntem, aşağıda durum geçişlerinin ve ZS-Yapısının gerçeklenmesine ilişkin olarak iki aşamada tanımlanacaktır.

Durum geçişlerinin ve ilk durumun gerçeklenmesi

Daha önce bahsedildiği gibi standart gerçekleme yöntemi ile, çığ etkisi problemini ortadan kaldıran sistematik bir çözüm verilememektedir. Ayrıca, bu yöntem bazı PLC modellerinde normal bit işlem komutlarına göre program belleğinde önemli miktarda daha fazla yer kaplayan kurma ve silme (SET ve RESET) komutlarını yoğun olarak kullanmaktadır. Örneğin Siemens S7-200 PLC'lerde standart bir bit işlem komutu için program belleğinde 2 byte'lık, SET ve RESET komutlarının her biri için 7 byte'lık alan ayrılır. Ayrıca bu komutların kullanılması, geçiş fonksiyonlarının PLC programından takibini de güçleştirmektedir.

Aşağıda tanıtılacak yöntem, temel olarak kurma ve silme komutlarının kullanılmadığı, çığ etkisi problemi için kesin bir çözüm oluşturan ve takip

edilmesi, değişikliklerin uygulanması kolay olan bir yapı oluşturmaktadır.

Yöntem, PLC'nin çalışma prensibine uygun olarak zamanda ayrık anlarda işletilen mantıksal fonksiyonlarla ifade edilecektir. Bir mantıksal q değişkeninin kurma koşulu S ile, silme koşulu R ile temsil edilsin. $k \in \mathbf{N}$ olmak üzere q değişkeninin mevcut değeri q(k), bir sonraki değeri q(k+1) ile gösterilirse

$$q(k+1) = S + q(k).\overline{R} \qquad (11)$$

mantıksal fonksiyonunun q'yu S ile kurduğu, R ile sıfırladığı kolaylıkla gösterilebilir. Burada '+' mantıksal VEYA, "." VE işlemine, $\overline{R}$ ise R'nin mantıksal DEĞİL'ine karşılık gelmektedir. Bu fonksiyon verilen bir otomatın durum geçiş koşullarının gerçeklenmesi için de kullanılabilir. Aşağıdaki tanımlar ZS-otomatın durum geçiş koşullarının mantıksal fonksiyonlarla ifade edilebilmesine yönelik olarak verilmiştir. Tanımlar kolaylıkla standart otomat için uyarlanabilir (Hasdemir vd., 2008).

Tanım 3. $G_{ZS} = (Q, \Sigma, f, \Gamma, T, C, \Delta, q_0)$ ZS-otomatı verilsin. $Q = \{q_1, q_2, ..., q_{N_Q}\}$ durum kümesindeki her duruma 1 ile N_Q arasında bir tam sayı karşılık gelecek şekilde oluşturulan $I_Q = \{1, 2, ..., N_Q\}$ kümesine G_{ZS}'ye ilişkin *durum indis kümesi* denir.

Tanım 4. $G_{ZS} = (Q, \Sigma, f, \Gamma, T, C, \Delta, q_0)$ ZS-otomatı verilsin. $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_{N_\Sigma}\}$ durum kümesindeki her duruma 1 ile N_Σ arasında bir tam sayı karşılık gelecek şekilde oluşturulan $I_\Sigma = \{1, 2, ..., N_\Sigma\}$ kümesine G_{ZS}'ye ilişkin *olay indis kümesi* denir.

Tanım 5. Durum geçiş fonksiyonu f olan bir ZS-otomata ilişkin durum indis kümesi $I_Q = \{1, 2, ..., N_Q\}$, olay indis kümesi $I_\Sigma = \{1, 2, ..., N_\Sigma\}$ olsun. q_i *durumuna geçişe ilişkin durum indis kümesi* şu şekilde tanımlanır:

$$I_{SQ}(i) = \{j \in I_Q \mid \exists k \in I_\Sigma,\ f(q_j, \sigma_k) = q_i\},\ \forall i \in I_Q$$

$I_{SQ}(i)$ kümesinin elemanları q_i durumuna geçişin mümkün olduğu durum indislerine karşılık gelir.

Tanım 6. Durum geçiş fonksiyonu f olan bir ZS-otomata ilişkin durum indis kümesi $I_Q = \{1, 2, ..., N_Q\}$, olay indis kümesi $I_\Sigma = \{1, 2, ..., N_\Sigma\}$ olsun. q_i *durumuna geçişe neden olan olay indis kümesi* şu şekilde tanımlanır:

$$I_{S\Sigma}(i) = \{j \in I_\Sigma \mid \exists k \in I_Q,\ f(\sigma_k, \sigma_j) = q_i\},\ \forall i \in I_q$$

Tanım 7. Durum geçiş fonksiyonu f, aktif olay fonksiyonu Γ olan bir ZS-otomata ilişkin durum indis kümesi $I_Q = \{1, 2, ..., N_Q\}$, olay indis kümesi $I_\Sigma = \{1, 2, ..., N_\Sigma\}$ olsun. q_j *durumundan* q_i *duruma geçişe neden olan olay indis kümesi*:

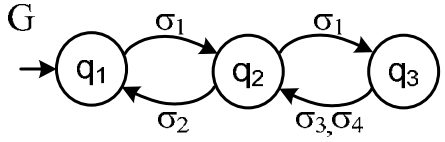
$$I_{T\Sigma}(i, j) = \{k \in \Sigma \mid k \in \Gamma(q_j) \wedge k \in I_{S\Sigma}(i)\},\ \ \forall (i, j) \in I_{SQ}$$

şeklinde tanımlanır.

Tanım 8. Durum geçiş fonksiyonu f, aktif olay fonksiyonu Γ olan bir ZS-otomata ilişkin durum indis kümesi $I_Q = \{1, 2, ..., N_Q\}$, olay indis kümesi $I_\Sigma = \{1, 2, ..., N_\Sigma\}$ olsun. q_i *durumundan çıkışa neden olan olay indis kümesi* şu şekilde tanımlanır:

$$I_{R\Sigma}(i) = \{j \in I_\Sigma \mid \sigma_j \in \Gamma(q_i)\},\ \forall i \in I_Q$$

Buraya kadar verilen tanımlar Şekil 4'te verilen örnek otomata uygulanacaktır.



Şekil 4. Örnek bir otomat

Örnek 2. Şekil 4'te verilen otomat için durum indis kümesi $I_Q = \{1,2,3\}$, olay indis kümesi $I_\Sigma = \{1,2,3,4\}$ 'tür. Tanım 5, 6, 7 ve 8, q_2 için uygulanırsa aşağıdaki kümeler elde edilir.

q_2 durumuna geçişe ilişkin durum indis kümesi $I_{SQ}(2) = \{1,3\}$; q_2 durumuna geçişe neden olan olay indis kümesi $I_{S\Sigma}(2) = \{1,3,4\}$, $q_j\ (\forall j \in I_Q)$ durumlarından q_2 durumuna geçişe neden olan olay indis kümeleri; $I_{T\Sigma}(2,1) = \{1\}$, $I_{T\Sigma}(2,2) = \phi$, $I_{T\Sigma}(2,3) = \{3,4\}$; q_2 durumundan çıkışa neden olan olay indis kümesi $I_{R\Sigma}(2) = \{1,2\}$

Tanım 9. q_i *durumuna ilişkin kurma koşulu*nun k adımındaki mantıksal ifadesi

$$S_i(k) = \sum_{j \in I_{SQ}(i)} q_j(k).\sigma(i,j)\ , \qquad \sigma(i,j) = \sum_{n \in I_{T\Sigma}(i,j)} \sigma_n$$

ile verilir. $\sigma(i,j)$, q_j durumundan q_i durumuna geçişe neden olan tüm olayların eşdeğerine karşılık gelmektedir.

Tanım 10. q_i *duruma ilişkin silme koşulu*nun mantıksal ifadesi

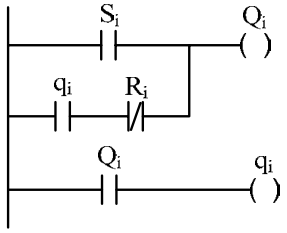
$$R_i(k) = \sum_{j \in I_{R\Sigma}(i)} \sigma_j(k)$$

ile verilir.

Tanım 9 ve 10 ile *i* indisli duruma ilişkin kurma ve silme koşulları verildiğinden, bu duruma ilişkin mantıksal eşitlik (11) ifadesinden faydalanılarak şu şekilde elde edilebilir:

$$q_i(k+1) = S_i(k) + q_i(k).\overline{R_i(k)} \qquad (12)$$

(12) eşitliğine ilişkin PLC programı kolaylıkla merdiven dili programlama tekniği ile elde edilebilir. $q_i(k+1)$ mantıksal değişkeni Q_i, $q_i(k)$ mantıksal değişkeni q_i ile gösterilirse (12) eşitliğine karşılık gelen PLC programı Şekil 5'teki gibi oluşturulabilir. Programın son basamağında Q_i değişkeninin içeriği q_i değişkenine atanmaktadır. Buna göre PLC programındaki q_i mantıksal değişkeni, q_i durumunun mevcut PLC tarama çevrimindeki; Q_i mantıksal değişkeni ise bu durumun bir sonraki PLC çevrimindeki gösterimine karşı geldiği yorumu yapılabilir.



Şekil 5. Eşitlik 12'ye karşılık gelen PLC programı

Örnek 3. Bu örnekte Şekil 4'teki otomat tekrar ele alınacak, bu otomata Tanım 9 ve 10 ile verilen ifadeler elde edilerek otomatı gerçekleyen PLC programı elde edilecektir.

Tanım 9 ve Tanım 10'a göre bu otomatın q_2 durumu için

$$S_2(k) = \sum_{j \in I_{SQ}(2)} q_j(k).\sigma(2,j) = q_1(k).\sigma_1 + q_3(k).(\sigma_3 + \sigma_4) \qquad (13)$$

$$R_2(k) = \sum_{j \in I_{R\Sigma}(2)} \sigma_j(k) = \sigma_1 + \sigma_2 \qquad (14)$$

ifadeleri elde edilir. Bu ifadeler Eşitlik 12'de yerine konulursa q_2 durumuna ilişkin mantıksal ifade

$$q_2(k+1) = q_1(k).\sigma_1 + q_3(k).(\sigma_3 + \sigma_4) + q_2(k).\overline{\sigma_1}.\overline{\sigma_2} \qquad (15)$$

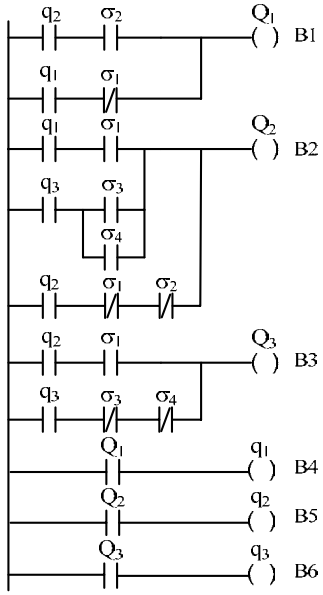
şeklinde elde edilir. Diğer durumlara ilişkin mantıksal ifadeler oluşturulursa, durum geçiş fonksiyonu gerçekleyen program parçası Şekil 5'e uygun olarak Şekil 6'daki gibi elde edilir.

İlk durumun programlanması

İlk durumun programlanması için PLC'nin ilk tarama çevriminde mantıksal 1 değeri alan özel belek alanlarından faydalanmak mümkündür. Örneğin S7-200 PLC'sinde SM0.1 ile adreslenen özel bellek alanı program işletilmeye başladığı anda ilk duruma karşılık gelen bit değişkenini kurmak için kullanılabilir. Bunun dışında, kullanılacak olan PLC'nin böyle bir özelliğinden bağımsız olarak "otomat hiçbir durumda değilse ilk durumdadır" mantığı ile de ilk duru-

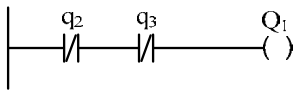
mun programlanması mümkündür. Bu sözel ifade, i_0 ilk durumun indis numarası olmak üzere, formal olarak aşağıdaki mantıksal eşitlikle verilebilir.

$$q_0(k+1) = \prod_{i \in I_Q \setminus i_0} \overline{q_i(k)} \qquad (13)$$



Şekil 6. Örnek 3'e ilişkin PLC programı

Buna göre Şekil 4'te verilen otomat için ilk duruma ilişkin mantıksal fonksiyon $Q_1 = \overline{q_2}.\overline{q_3}$ şeklinde olacaktır. Bu mantıksal fonksiyonu gerçekleyen PLC programı ise Şekil 7'de verilmiştir.



Şekil 7. İlk duruma ilişkin gerçekleme

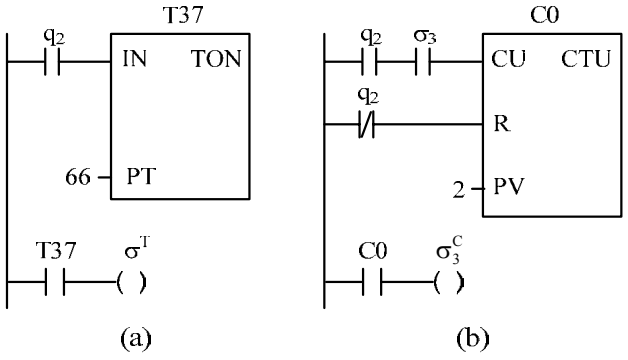
Şekil 6'daki B1 basamağı yerine Şekil 7'deki ilk duruma karşılık gelen program parçasının kullanılması gerçekleme açısından hiçbir probleme neden olmayacaktır. Çünkü, Şekil 7'deki programa göre q_2 ya da q_3 durumundan çıkılıp q_1 durumuna geçilen her koşulda Q_1 değişkeni mantıksal 1 seviyesine kurulacak, yani PLC gerçeklemesinde istenilen davranış sağlanmış olacaktır. Bu ise ilk durumun programlanması için PLC'de daha fazla program belleği kullanmak yerine, program belleğinden kazanım anlamına gelmektedir.

ZS-yapısının ve çıkış fonksiyonların gerçeklenmesi

Gerçekleme yöntemi için buraya kadar verilen tanımlar durum geçişleri ve ilk durumun programlanabilmesi ile ilgilidir ve standart otomat gösteriminin gerçeklenmesi için de kolaylıkla uyarlanabilir. ZS-otomatı standart otomattan ayıran Zamanlama ve Sayma Yapıları (ZS-Yapısı) ile çıkış fonksiyonlarının gerçeklenmesine ilişkin yöntemlerin de tanımlanmasına gereksinim vardır. Bunların gerçeklenmesi için PLC'lerde zamanlama ve sayma işlemlerini yerine getiren hazır program bloklarından, yani Zamanlayıcı ve Sayıcılardan faydalanılacaktır. ZS-Yapısı ve ilgili çıkış fonksiyonlarının gerçekleme yöntemi de bu blokların tanımındaki giriş çıkışlar ile verilebilir. Bunun için Siemens S7-200 PLC'lerin komut kümeleri kullanılmıştır.

Sıfırdan farklı bir gecikme öngörülen bir durum için Gecikmeli Kapatan Zamanlayıcının (TON) "IN" girişine, bu durumu temsil eden mantıksal değişken, "PT" girişine ise gecikme süresine karşılık gelen bir tamsayı değeri atanmalıdır. Bir q durumu için Zamanlama Yapısında τ_q ile tanımlanan gecikme süresi, "PT" girişine $10 \times \tau_q$ değerini girerek T37-T63 ya da T101-T255 zamanlayıcılarından biri ile saniye cinsinden gerçeklenebilir. Şekil 3'te verilen ZS-otomat için ZY ile tanımlanan $\tau_{q_2} = 6.6$ değeri ve buna ilişkin çıkış fonksiyonunun PLC gerçeklemesi Şekil 8a'da verilmiştir. Zamanlayıcı çıkışı, zamanlama olayına karşılık gelen σ^T değişkenine atanmaktadır. Zamanlama olayının meydana gelmesi Şekil 3'e göre q_1'e geçişle sonuçlanacağından, PLC programında σ^T'nin mantıksal 1 değerini alması bir sonraki PLC çevriminde q_2'nin, dolayısıyla T37 zamanlayıcısının sıfırlanmasına neden olacaktır. Böylece σ^T değişkeni olayların gerçeklenme prensibine uygun olarak, bir tarama çevrimi boyunca mantıksal 1 değerini almış olacaktır.

Sayma Yapısı ile bir q_i durumu için tanımlanan $\mathbf{c}_{q_i}$ vektörü için $c_{q_i}(j) \neq 0$ koşulu, σ_j olayının q_i durumda $c_{q_i}(j)$ kez sayılacağına karşılık gelir. Bu koşul CTU sayıcısının "CU" girişine $q_i.\sigma_j$ mantıksal fonksiyonu atanmasıyla gerçeklenebilir. İlgili durumdan çıkılmasıyla da sayıcıyı sıfırlayan "R" girişine mantıksal 1 değeri atanmalıdır. Bu ise $\overline{q_i}$ ile gerçeklenebilir. Son olarak CTU sayıcısının "PV" girişi için Sayma Yapısı ile tanımlanan $c_{q_i}(j)$ değeri atanmalıdır. Şekil 3'te verilen ZS-otomat için Sayma Yapısı ile tanımlanan $c_{q_2}(3) = 2$ değerine ilişkin çıkış fonksiyonunun PLC gerçeklemesi Şekil 8b'de verilmiştir. Görüldüğü gibi C0 sayıcısının çıkışı σ_3^C sayma olayına atanmaktadır.



Şekil 8. ZS-Yapısına ilişkin çıkış fonksiyonlarının gerçeklenmesi

Sonuç

Bu çalışmada ayrık olay sistemlerinin kontrolüne yönelik tasarımı ifade etmek için kullanılabilecek, durum diyagramlarını temel alan bir modelleme biçimi ve gerçekleme yöntemi tanımlanmıştır. Tanımlanan gösterim biçimi uygulamada sıklıkla karşılaşılan davranış türlerinden zamanlama ve sayma davranışlarının ifadesini kolaylaştırmakta, aynı zamanda Programlanabilir Lojik Kontrolörlerle (PLC) gerçeklemede doğrudan kullanılabilecek bir yapı oluşturmaktadır. Zamanlama ve sayma davranışının ifadesi standart otomat gösterimine Zamanlama ve Sayma Yapısı olarak adlandırılan bir özelliğin kazandırılması ile sağlanmaktadır. Tanım açısından, Standart otomat gösteriminin ZS-otomat olarak adlandırılan bu modelleme biçiminin özel bir durumu olduğu söylenebilir.

Uygulamada sıklıkla karşılaşılan bir gerçekleme sorunu çığ etkisi olarak adlandırılır ve otomat gösterimindeki belirli bir yapısal özelliğin PLC'ler ile gerçeklemede hatalı sonuçlar vermesi ile ilgilidir. Literatürde bu sorun için sistematik olmaktan uzak, bazı uygulamalarında ise doğru bir gerçeklemenin elde edilmesinin imkânsız olduğu yöntemler önerilmiştir. Bu çalışmada önerilen ZS-otomat gösterim biçimi için, çığ etkisi problemi olmayan, sistematik bir gerçekleme yöntemi tanımlanmıştır. Yöntem uygulamacı tarafından doğrudan kullanılarak tasarıma karşılık gelen PLC programının oluşturulması için kullanılabilmektedir. Tamamen matematiksel ve sistematik bir şekilde ifade edilmesine rağmen, bu yöntem sezgisel olarak kolay uygulanabilir, elde edilen PLC kodu ise kolay okunabilir olma özelliğindedir. Tanımlanan yöntem uygulandığında, durum geçişlerine ilişkin koşullar, elde edilen PLC programında açıkça görülmektedir ve her bir duruma ilişkin geçiş koşulları için merdiven mantığı programında tek bir basamak yeterli olmaktadır. Bunun yanında gerçekleme yönteminin sistematik ifadesi, bu yöntemin programlanarak gerçeklenmesini de mümkün kılmaktadır. Bu ise PLC programlarının otomatik olarak üretilmesini mümkün kılan yazılım uygulamaların geliştirilebileceği anlamını taşımaktadır.

Kaynaklar

Brandin, B.A., (1996). The real-time supervisory control of an experimental manufacturing cell, *IEEE Transactions on Robotics and Automation*, **12**, 1, 1-14.

Fabian M. ve Hellgren A., (1998). PLC-based implementation of supervisory control for discrete event systems, *Proceedings*, 37th IEEE conference on Decision & Control, **3**, 3305-3310, Tampa, Florida, USA.

Hasdemir İ.T., Kurtulan S. ve Gören L., (2004). Implementation of local modular supervisory control for a pneumatic system using PLC, *Proceedings*, 7th International Workshop on Discrete Event Systems (WODES), 25-30, Reims, France.

Hasdemir İ.T., Kurtulan S. ve Gören L., (2008). An implementation methodology for supervisory

control theory, *International Journal of Advanced Manufacturing Technology*, 36, 373-385.

Kurtulan, S., (2007). Endüstriyel kumanda sistemleri, Nobel Yayın Dağıtım, İstanbul.

Queiroz, M.H. de ve Cury, J.E.R., (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell, *Proceedings*, 6th Int. Workshop on Discrete Event Systems (WODES), Zaragoza, Spain.

Ramadge, P.J. ve Wonham, W. M., (1987). Supervisory control of a class of discrete event processes, *SIAM Journal of Control and Optimization,* **25**, 1, 206-230.