타이밍도의 EMFG 표현에 관한 연구

김영운, 여정모
부경대학교 전자계산학과

A Study on the EMFG Representation of Timing Diagrams

Young-Un Kim, Jeong-Mo Yeo
Department of Computer Science
Pukyong National University
E-mail : yhkim@dol.pknu.ac.kr

요 약

디지털 시스템을 설계하고 분석할 때, 번지버스와 데이터버스 및 각종 제어 신호들을 타이밍도로 표현하는 경우가 많다. 그러나 디지털 시스템의 동작이 타이밍도로 표현되는 경우, 그 표현이 복잡할 뿐 아니라 동작 분석이 용이하지 못하다.

본 연구에서는 시스템의 타이밍도를 확장된 마크흐름선도(EMFG; Extended Mark Flow Graph)로 표현하는 방법을 제안하였다. 시스템의 동작이 EMFG로 표현되는 경우, 각종 신호들에 따라 변화하는 시스템의 상태가 도식적으로 표현되므로 시스템의 동작 분석이 용이해질 뿐 아니라 시스템의 설계에도 유용하게 이용될 수 있다.

적용 예로 NEC사의 μPD70320 CPU의 메모리 읽기 사이클 및 MCM60256A의 메모리 동작을 EMFG로 표현하였다.

ABSTRACT

A Timing Diagram is almost used to represent the various signals such as an address bus, a data bus, and the control signals during design and analysis of a digital system. But if so, its representation is somewhat complicated and also it is difficult to analyze the operation of the system.

In this paper, we proposed the representation method of timing diagrams with the EMFG(Extended Mark Flow Graph). In the EMFG representation of the system operation, the logical states due to the various signals of the system is graphically represented. Therefore the proposal method allows that it is easy to design as well as analyze the system.

As examples applied, we represented the memory read cycle of μPD70320 CPU and the read cycle of MCM60256A memory with the EMFG.

Ⅰ. 서 론

시스템 설계 및 분석에 있어 타이밍도는 설계자에게 중요한 정보를 제공한다. 하지만, 이러한 타이밍도를 이해하고 설계하기에는 많은 시간이 요구되며 또한 각각의 소자들을 서로 연결하였을 때 타이밍도간의 신호들이 어떻게 동작되는지 명확히 알 수 없다.

확장된 마크흐름 선도(EMFG; Extended Mark Flow Graph)는 시각적으로 알기 쉽고 정확한 표현으로 시스템을 기술할 수 있다.[1] EMFG의 이론[2-3]을 이용하여 타이밍도를 표현함으로써 시스템의 동작을 쉽고 명확하게 이해할 수 있으며 나아가 시뮬레이션도 가능하다.

본 논문에서는 동시성 및 병렬성을 잘 표현할 수 있는 EMFG의 성질을 이용하여 타이밍도를 EMFG로 표현하는 방법을 제안하였다.

본 논문은 Ⅱ장에서 EMFG의 정의 및 성질에 대해 기술하였고, Ⅲ장에서는 타이밍도를EMFG로 표현하는 방법에 대하여 기술하여, 제시된 방법에 따라 CPU의 메모리 읽기 타이밍도와 메모리 동작을 각각 EMFG로 표현하였으며, Ⅳ장에서는 EMFG로 표현된 CPU의 메모리 읽기 타이밍도와 메모리 동작을 하나로 결합하여 EMFG로 표현하였다. 끝으로 Ⅴ장에서는 결론 및 향후 연구과제에 대하여 기술하였다.

Ⅱ. EMFG 정의 및 성질

Ⅱ.Ⅰ. EMFG 정의

EMFG는 Petri Net[4]의 한 부류로써 박스(box), 트랜지션(transition) 및 아크(arc)들로 구성되는 마크(mark)를 갖는 방향성 그래프이다.

그림 1은 EMFG의 한 예를 보여주고 있다.

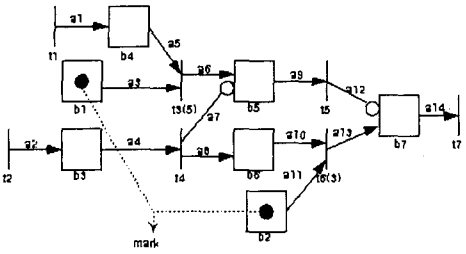

그림 1. EMFG

박스는 상태(개념적인 상태, 제어 상태, 신호 상태 등)를 나타내며, 상태의 만족 여부는 박스의 마크 존재유무로 결정된다. 즉 마크가 존재하면 상태가 만족되어 있으며, 마크가 존재하지 않으면 상태가 만족되어 있지 않다.

박스는 자신의 조건에 의해 또는 자신과 다른 상태(들)의 조건(들)에 의해서 변화되는 어떤 상태를 나타내고, 사각형으로 표기한다. 특히 출력 트랜지션만을 가지는 박스를 소스 박스(source box)라 하고, 수동적이거나 기계적으로 그 상태가 결정되고 게이트 역할을 하는 박스이다. 그림 1에서 b1, b2가 소스박스이다.

트랜지션은 박스(들)의 상태전이(점화)가 일어나는 곳으로서 점화시간을 요하지 않거나 특히 시간에 무관하게 점화하는 일반 트랜지션(general transition)과 점화시간을 요하는 시간 트랜지션(time transition)이 있고, 막대(bar)로 표기한다. 그림 1에서 t3(5)와 t6(3)은 점화시간 5와 3을 가진 시간 트랜지션이고, 출력 아크만을 가진 트랜지션 t1, t2는 소스 트랜지션(source transition)이고, 입력 아크만을 가진 트랜지션 t7을 싱크 트랜지션(sink transition)이라하고, 나머지 t4, t5는 일반 트랜지션이다.

아크는 트랜지션의 점화조건을 결정하고 점화완료 후의 마크 상태를 결정하는 역할을 하고, 화살표로 표기되는 일반 아크(general arc)와 선과 작은 원으로 표기되는 역 아크(inhibit arc)가 있으며, 모두 트랜지션의 입출력으로 사용될 수 있다. 그림 1에서 a7과 a12는 역아크이고, 나머지 모든 아크는 일반 아크이다.

Ⅱ.Ⅱ. EMFG의 성질

Ⅱ.Ⅱ.Ⅰ. 트랜지션의 점화조건

트랜지션의 점화조건은 트랜지션의 모든 입력 박스(들)의 마크 상태가 만족될 때 만족된다. 일반 아크로 연결된 입력 박스에는 마크가 존재해야, 역 아크로 연결된 입력 박스에는 마크가 없어야 점화조건을 만족시킨다. 그리고 소스 트랜지션은 점화조건을 만족시킬 입력 박스가 없으므로 항상 점화조건이 만족되어 있다.

Ⅱ.Ⅱ.Ⅱ. 트랜지션의 점화 동작

트랜지션은 점화조건이 만족되면 즉시 점화를 개시한다. 일반 트랜지션은 점화시간을 요하지 않으므로 즉시 점화가 완료되지만, 시간 트랜지션은 점화 시간이 경과된 후 점화가 완료된다. 따라서 시간 트랜지션이 점화하는 도중에 어떤 원인에 의해 점화조건이 만족되지 않으면 즉시 점화가 정지된다. 점화가 완료되었을 때, 트랜지션에 연결된 박스들의 마크 상태는 다음과 같이 변화한다. 우선 일반/역 아크로 연결된 입력 박스의 마크는 소멸/유지되고, 다음에 일반/역 아크로 연결된 출력 박스에는 마크가 생성/소멸된다.

Ⅲ. 타이밍도의 EMFG 표현

일반적으로 디지털 시스템을 구현할 때는 디지털 소자들을 서로 연결하여 설계하게 된다. 그리고 각 소자들의 타이밍도를 사용하여 설계된 시스템의 동작여부를 판단하게 된다. 특히 디지털 시스템인 경우는 각종 신호들이 클럭 펄스에 동기되어 신호 상태들이 변화되므로 클럭 펄스에 따른 신호 상태의 변화를 검사하여 그 동작을 분석할 수 있지만, 시스템의 신호 상태들이 동시성, 직렬성 및 병렬성 등을 가지고 있으므로 그 분석이 복잡할 뿐 아니라 해석이 어려워 많은 경험을 필요로 한다.

따라서 동시성, 직렬성 및 병렬성 등을 잘 표현할 수 있는 EMFG를 사용하여 디지털 시스템의 동작을 표현한다면 분석 및 해석이 용이하게 된다.

Ⅲ.Ⅰ. 타이밍도의 EMFG 표현

동기 시스템에 사용되는 각종 소자들의 신호들은 클럭 펄스에 동기되어 변화하고 시스템의 상태를 변화시킨다. 따라서, 클럭 펄스에 따른 각 소자의 신호상태를 EMFG로 표현함으로써 시스템의 동작상태를 파악할 수 있다.

일반적으로 디지털 시스템에 사용되는 소자는 크게 세 가지로 구분할 수 있다. 첫 번째로, 게이트와 같은 계열의 변환소자이다. 즉 입력신호를 가지고 출력신호를 만들어 내는 소자로서 변환(conversion)소자라 지칭하기로 한다. 여기에는 GAL, PAL 등이 속한다. 두 번째로, 제어신호 및 버스신호등을 만들어 내는 소자로서 능동적인(active) 소자라 지칭하기로 한다. 여기에는 CPU, CONTROLLER 등이 속한다. 세 번째로, 제어신호에 따라 동작하는 소자로서 수동적인(passive) 소자라 지칭하기로 한다. 여기에는 MEMORY, LCD 등이 있다.

위에서 말한 소자들의 신호는 입력 신호와 출력 신호로 구분할 수 있으며, 또한 신호들은 각각 타이밍도로 표현될 수 있다.

타이밍도를 EMFG로 표현하기 위해 박스, 트랜지션, 마크 등을 표 1과 같이 의미를 부여한다.

표 1. 박스, 트랜지션, 마크의 표현

종 류	의 미
박 스	제어신호 및 버스신호, 상태신호를 박스로 표현한다.
일반 트랜지션	지연시간이 없는 동작
시간 트랜지션	지연시간이 있는 동작
마 크	유효 상태 또는 논리레벨

타이밍도의 EMFG 표현에 앞서 먼저 능동적인 소자인지 수동적인 소자인지를 판별한다.

다음은 능동적인 소자를 EMFG로 표현하기 위한 방법을 설명한다. 그림1은 능동적인 소자의 출력 타이밍도를 나타내고 있다.

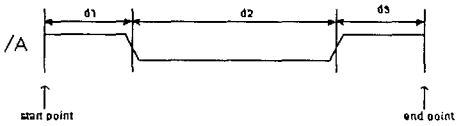

그림 2. 능동적인 소자의 타이밍도 예1

그림2에서 능동적인 소자는 시작시점(start point)과 끝 시점(end point)을 가진다.

방법 1) 능동적인 소자는 시작시점(start point)을 기준으로 상태가 변화됨으로 시작박스(start box)로부터 신호상태를 초기화한다.

그림2에서 신호 /A의 시작상태는 논리레벨 1을 가리키므로 EMFG로 표현하면 그림3과 같다.

그림 3. 그림2에서 시작상태 표현

방법 2) 시작상태이후의 신호상태 변화는 트랜지션의 점화 결과에 따른다.

그림2에서 신호 /A의 상태변화를 트랜지션으로 표현하면 그림4와 같다.

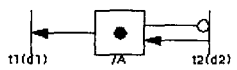

그림 4. 시작상태 이후의 상태표현

따라서, 그림2를 EMFG로 표현하면 그림5와 같다.

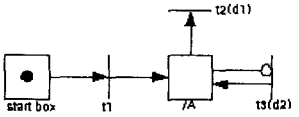

그림 5. 그림2의 EMFG 표현

방법 3) 시작시점(start point)을 기준으로 초기의 신호 상태가 정해지지 않는 경우(그림6 참조) 즉 논리레벨 0과1이 모두 나타나는 경우에는 시작박스(start box)를 두지 않는다.

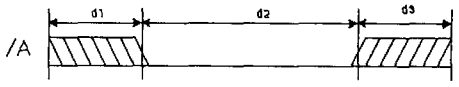

그림 6. 능동적인 소자의 타이밍도 예2

그림6을 EMFG로 표현하면 그림7과 같다.

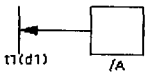

그림 7. 그림 6의 EMFG 표현

방법 4) 여러 개의 신호나 상태가 하나의 타이밍도에 함께 나타나면 각각의 박스와 트랜지션으로 나타낸다.

다음은 수동적인 소자인 경우에 타이밍도를 EMFG로 표현하는 방법에 대해 설명한다. 수동적인 소자는 동작을 일으키기 위한 내부지연 시간(그림8의 pt)을 가진다. 그림8은 수동적인 소자의 타이밍도를 나타내고 있다.

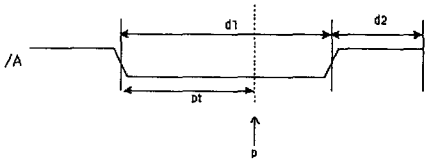

그림 8. 수동적인 소자의 타이밍도 예1
※ p : 동작 기준점(operation base point))

방법 5) 수동적인 소자인 경우 시작박스(start box)는 능동적인 소자의 제어신호 및 버스신호가 된다.

그림8을 EMFG로 표현하면 그림9와 같다.

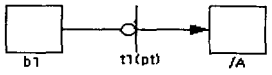

그림 9. 그림 8의 EMFG 표현

※ b1 : 능동적인 소자의 신호 상태
※ 수동적인 소자의 경우 d1시간의 의미는 능동적인 소자의 신호상태에 종속됨으로 그림9에는 표기하지 않았다.

지금까지 개별적인 신호에 대해 타이밍도를 EMFG로 표현하는 방법을 설명하였다. 이러한 개별적인 타이밍도는 특정 사이클 동안에 여러 개가 동시에 존재한다. 따라서, 각각의 EMFG를 하나의 EMFG로 합치는 방법에 대해 설명한다.

예를 들어 그림 10은 NEC사의 μPD70320(이하 V25로 약칭함) CPU의 메모리 읽기 타이밍도를 나타내고 있다.

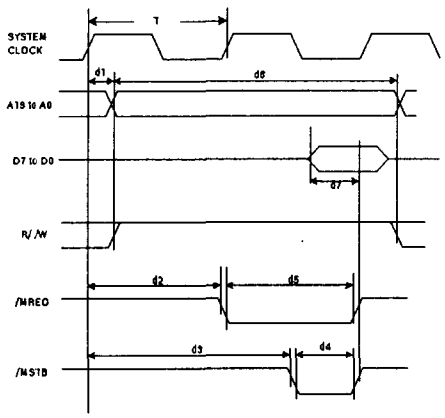

그림 10. V25 CPU의 메모리 읽기 사이클

CPU는 능동소자이므로 각각의 신호에 대한 EMFG 표현은 앞서 설명한 부분과 같다.

㉠ 번지신호는 그림11과 같이 표현된다.

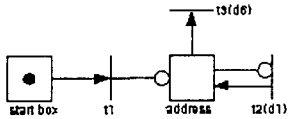

그림 11. 번지신호의 EMFG 표현

㉡ 데이터신호는 입력신호이므로 외부상태 신호에 의존한다.

㉢ R//W신호는 그림12와 같다.

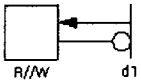

그림 12. R//W신호의 EMFG 표현

㉣ /MREQ 신호는 그림13과 같이 표현된다.

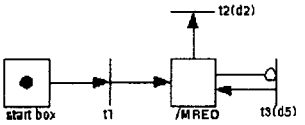

그림 13. /MREQ 신호의 EMFG 표현

㉤ /MSTB 신호는 그림14와 같이 표현된다.

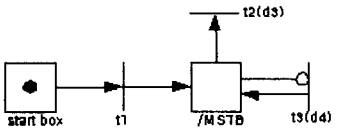

그림 14. /MSTB 신호의 EMFG 표현

이렇게 각각 표현된 EMFG는 메모리 읽기 사이클 동안에 동시에 나타나게 된다.

방법 6) 같은 성질의 박스 및 트랜지션은 한 개로 취급한다.

방법 7) CPU와 같은 능동적인 소자의 동작은 실행 사이클(execution cycle) 동안에 이루어지므로 진행중이라는 상태박스(progressing box)를 두기로 한다.

위와 같은 방법들을 이용하여 각각의 EMFG를 하나로 합치게 되면 그림15와 같다. 그림15는 그림10의 V25 메모리 읽기 타이밍도를 하나의 EMFG로 표현한 것이다.

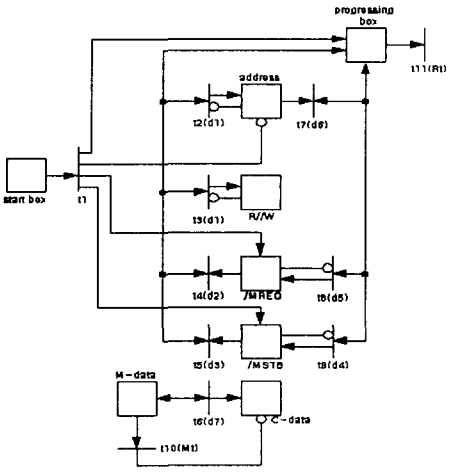

그림 15. 그림 10의 EMFG 표현

※ Rt : 메모리 읽기 사이클 시간
(V25에서는 두 클럭으로 정의된다.)
Mt : 메모리의 데이터가 무효하게 되는 시간

다음은 MCM60256A사의 메모리 읽기 동작을 그림16에 나타내었다.

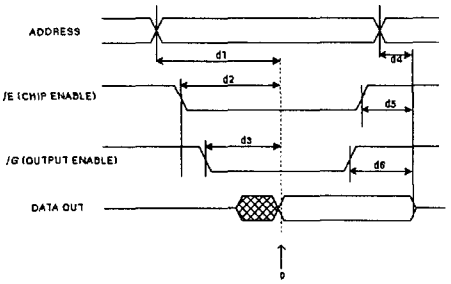

그림 16. MCM60256A의 읽기 사이클

일반적으로 타이밍도를 이해하기 위해서는 각각의 제어신호에 대한 이해가 요구된다. 또한, 타이밍도에서 신호의 우선 순위 즉 신호가 반드시 먼저 인가되어야 하는 곳과 우선 순위에 관계없이 임의로 신호가 인가되는 부분을 구별하기 어렵다. 위의 그림16에서는 /E 신호가 /G 신호보다 먼저 활성화된다. 하지만, 이것은 사용자에게 오해를 불러 일으킬 수 있다. 신호 /E와 /G는 순서에 관계없이 동작하기 때문에 반드시 /E신호가 먼저 활성화된다고 말할 수 없다. 하지만, EMFG로 표현하면 순서에 관계없이 동작 가능하다는 것을 알 수 있다.

메모리는 수동적인 소자이므로 각각의 신호에 대한 EMFG 표현은 다음과 같다.

㉠ 번지 신호는 그림17과 같이 표현된다.

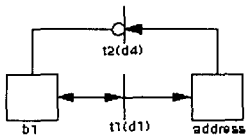

그림 17. 번지신호의 EMFG 표현

㉡ /E 신호는 그림18과 같이 표현된다.

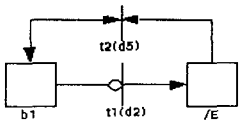

그림 18. /E 신호의 EMFG 표현

㉢ /G 신호는 그림 19와 같이 표현된다.

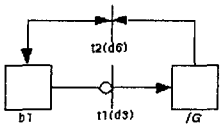

그림 19. /G 신호의 EMFG 표현

위에 나타난 b1 박스는 능동적인 소자의 신호상태 박스이다. 이렇게 각각 표현된 EMFG를 하나로 합치게 되면 그림20과 같다.

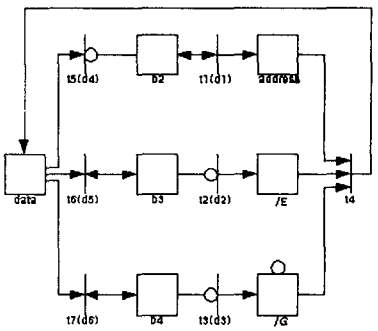

그림 20. 그림 16의 EMFG 표현

※ b2, b3, b4는 각각 능동적인 소자의 신호상태를 나타낸다.

IV. 적용 예

일반적으로 디지털 시스템은 여러 개의 소자들이 서로 연결되어 있고 이러한 소자의 동작은 능동적인 소자의 제어신호에 의해 동작된다.

예를 들어 V25 CPU 와 메모리의 연결된 모습이 그림21에 나타나 있다.

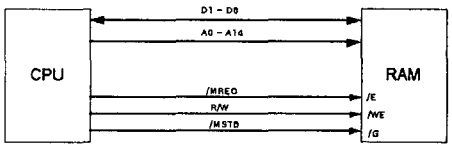

그림 21. V25와 메모리의 연결

그림21은 가장 이상적으로 연결된 모습을 보여주고 있다. 실질적인 디지털 시스템에서는 더 많은 소자들이 CPU와 연결되어 있으므로 CPU에서 나오는 제어신호 및 버스신호들이 직접 메모리에 인가되는 경우는 거의 없다.

그림21에서 CPU의 동작과 메모리 동작을 분석하기 위해 타이밍도를 가지고 동시에 해석하기란 어렵다. 또한 클럭을 기준으로 각각의 타이밍도를 하나로 그리기란 불가능한 일이며, 타이밍도를 이용하여 시뮬레이션 하기도 어렵다.

하지만, EMFG의 특성인 동시성 및 병렬성을 이용한다면 그림24의 CPU와 메모리 동작을 동시에 해석 할 수 있다.

앞에서 타이밍도를 EMFG로 표현하는 방법과 능동적인 소자와 수동적인 소자에 대해 각각 EMFG로 표현하였다.

이렇게 각기 표현된 EMFG를 서로 연결하면 그림22와 같이 CPU 동작과 메모리 동작을 동시에

나타낼 수 있다.

그림22는 V25 CPU의 메모리 읽기 타이밍도와 메모리의 읽기 동작을 동시에 표현한 그림이다. 따라서 CPU 동작 및 메모리 동작을 동시에 이해할 수 있고, 전체적인 흐름도 쉽게 파악 할 수 있다.

V. 결론 및 향후 과제

본 논문에서는 타이밍도가 가지고 있는 문제점들을 해결하기 위해 EMFG의 성질을 이용하여 타이밍도를 EMFG로 표현하는 방법에 대하여 제안하였다.

예로써, V25 CPU의 메모리 읽기 사이클과 MCM60256A의 메모리 읽기 동작을 EMFG로 표현함으로 타이밍도로써 표현하기 어려운 부분들과 소자들간의 동작 설명이 가능하였다. 또한 EMFG로 표현된 각각의 타이밍도는 서로 연결 가능하였다.

앞으로의 연구과제는 타이밍도를 보다 간결히 EMFG로 표현할 수 있는 방법을 모색하고, EMFG로 표현된 각각의 타이밍도를 이용하여 시스템 전체 동작을 분석하고 컴퓨터 시뮬레이션을 통한 내부 동작을 확인하는데 노력할 것이다.

참고문헌

[1] 여정모, "이산 시스템의 설계와 해석을 위한 확장된 마크흐름선도의 재정의와 회로변환," 멀티미디어학회 논문지, 제1권 제2호, p.224-238, 1998. 12.

[2] 여정모, "이산제어시스템 설계를 위한 확장된 마크흐름선도의 동작해석," 정보처리 논문지, 제5권 제7호, p.1896-1907, 1998. 7.

[3] 여정모, "EMFG 동작의 부울함수 해석," 부경대학교 논문집, 제3권 제2호, p.17-28., 1998. 12.

[4] J.L. Peterson, "Petri Net Theory and the Modelling of Systems", Prentice-Hall,1981

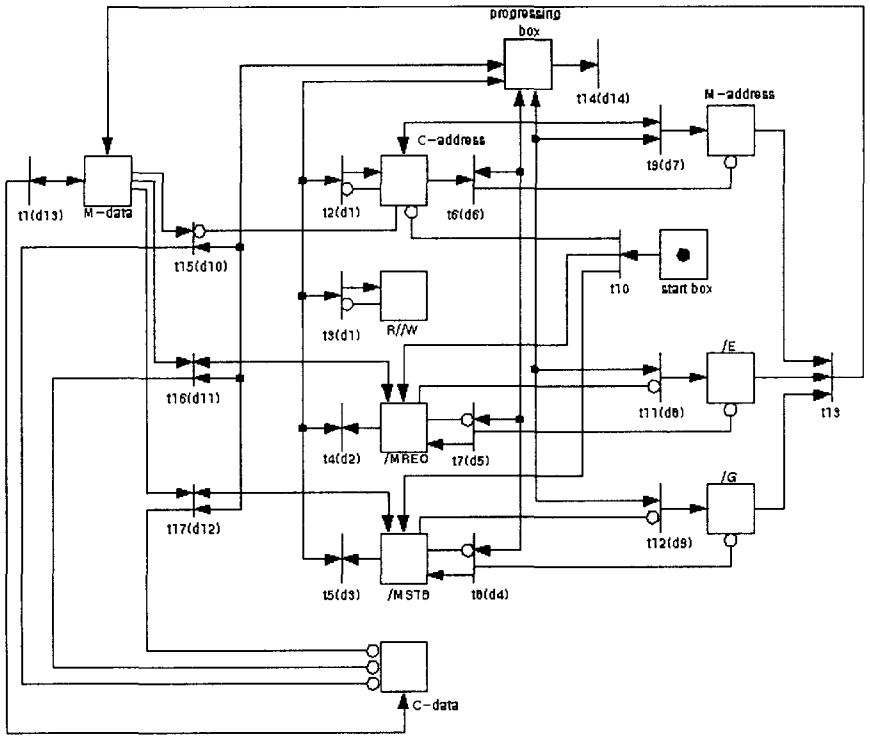

그림 22. EMFG로 표현한 V25와 메모리 읽기 사이클