실시간 운영체제환경을 고려한 SDL명세로부터의 C코드 생성

곽상훈* 이동익* 배영환**

*광주과학기술원 정보통신공학과
**한국전자통신연구원 집적회로설계연구부 시스템설계자동화팀
{shkwak, dilee}@kjist.ac.kr, yhbae@etri.re.kr

Translating SDL specification into C code in RTOS environment

Sang-Hoon Kwak* Dong-Ik Lee* Yeung-Hwan Bae**

*Dept. of Information and Communications, Kwang-Ju Institute of Science and Technology
**System Design Automation Team, IC Design Research Department, ETRI

요 약

내장형시스템의 복잡도의 증가, 하드웨어-소프트웨어 통합설계방법의 대두 등으로 인하여 시스템수준의 명세, 설계방법론에 관한 관심이 더욱 고조 되고 있다. 본 논문에서는 시스템기술언어로 ITU-T에 의해 표준으로 권고되어 널리 사용되고 있는 SDL (Specification and Description Language)로부터 실시간 운영체제에서 수행될 C코드를 자동으로 생성하는 방법론을 제시한다. C코드는 SDL 프로세스의 내부 행위(behavior)를 묘사하는 운영체제 비의존적인 코드와 다중프로세스수행, 프로세스간 커뮤니케이션과 같은 운영체제 의존적인 부분으로 나뉘어지며, 운영체제 서비스함수를 이용하여 생성된 C코드가 운영체제와 더욱 밀접하게 통합되어 수행된다.

1. 서론

최근 디지털시스템의 복잡도의 증가, 하드웨어와 소프트웨어가 혼재된 시스템의 동시설계 등으로 인하여 시스템수준 설계방법론, 시스템수준 사양기술언어 등에 관한 관심이 고조되고 있다[1]. 시스템수준명세언어를 이용하여 시스템을 기술할 경우 복잡한 시스템의 세부사항들을 나타내지 않고, 구현과 독립적인 기술이 가능하며, 최상위 수준에서 조기에 시스템의 동작을 검증할 수 있는 장점이 있다. 추상적인 시스템기술언어로부터 실제 구현언어로 변환시키는 과정은 점차 방대해지는 시스템규모와 조기시장진입이 관건인 내장형시스템 설계분야의 흐름으로 볼 때 전체 시스템설계과정 중 필수적인 부분이라 할 수 있다. 시스템수준기술언어로는 SDL, LOTOS, SystemC, SpecChart 등과 같이 여러 종류의 언어들이 존재하나 본 논문에서는 정형적으로 구문의 의미가 잘 정의되어 있고, 국제기관에 의해 표준으로 권고 되고 있으며, 구현독립적인 장점을 가지는[2] SDL로부터 내장형시스템에서 수행될 C코드를 생성하는 방법을 제안하고자 한다.

SDL은 1976년 CCITT(ITU-T의 전신)에 의해 처음 표준으로 권고된 이래 주로 통신시스템의 정형적 명세에 널리 이용되어 왔으며 최근에는 내장형시스템, 실시간시스템의 설계에 적용하려는 연구가 활발히 진행되고 있다. 본 논문에서는 제어중심의 내장형시스템 설계시에 SDL을 이용하여 시스템사양을 기술하고 그 사양으로부터 실시간운영체제상에서 수행될 소프트웨어 파트를 C코드로 자동 생성해주는 방법을 제안한다. 현재 많은 내장형시스템의 전형적인 아키텍쳐는 하드웨어적으로는 CPU 또는 MCU코어와 응용에 필요한 ASIC(Application Specific Integrated Circuit)과 주변장치를 부착한 형태이며, 소프트웨어파트는 커널의 크기가 작고 실시간 요구사항을 만족해 주는 실시간 운영체제 상에서 응용프로그램이 수행되는 형태이다 .

내장형시스템 설계에 SDL을 이용한 기존의 연구로는 [3][4]등이 있으나, 본 논문에서는 소프트웨어 구현을 위한 C코드를 생성시 운영체제에 의존하지 않는 프로세스내부 행동부분과 운영체제에 의존하는 프로세스간 통신부분을 분리하여 코드를 생성한다. 따라서 앞서 설명한 전형적인 내장형시스템 아키텍쳐에 적합하도록 대부분의 실시간운영체제에서 채택하고 있는 우선순위기반 선점형 스케줄링방식에 대한 고려와, 운영체제 시스템서비스를 이용하는 *tight integration*을 통하여 내장형 시스템 개발에 실질적인 도움이 되는 시스템을 구현하였다.

본 논문은 다음과 같이 구성되어 있다. 2절에서는 SDL 프로세스의 내부행위를 동일한 동작을 수행하는 C코드로 변환하는 방법에 관하여 설명하고, 3절에서는 SDL의 다중프로세스수행, 프로세스간 시그널 및 데이터교환을 운영체제의 시스템서비스함수를 이용하여 C코드로 변환하는 방법에 관해 설명한다. 4절에서는 SDL사양으로부터 SDL2C 코드변환기를 통해 구현한 Access Control시스템 예제를 소개하고 5절에서는 결론을 제시한다.

2. SDL명세의 C코드로의 변환

SDL로부터 C코드를 생성하기 위한 전체 과정을 그림 1에 나타내었다. 이 때 중간형태로 SDL Access라는 자료구조 및 API를 이용하여 초기 SDL코드를 파싱한 후 얻어진 access파일로부터 그와 동등한 C코드를 생성한다.

C 코드생성시에 목적운영체제의 시스템서비스 함수를 이용하면 SDL시그널전송 등의 동작을 좀 더 효율적으로 구현할 수 있다. 본 논문에서는 목적운영체제로 uC/OS-II라는 실시간 운영체제를 사용하였다. 대부분의 운영체제에서 프로세스생성, 소멸, 메시지전송, 세마포어와 같은 대표적인 종류의 운영체제 시스템함수와 자료구조를 지원하므로 본 논문에서 제안하는 SDL2C 변환기는 약간의 수정을 통해 다른 운영체제를 환경을 위한 C코드를 쉽게 생성할 수 있다.

2.1 SDL수행모델

SDL기술에서 시스템의 동작을 나타내는 기본단위는 프로세

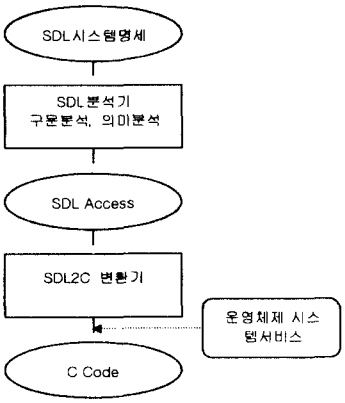


그림1. SDL2C 변환 흐름

스이며, 한 프로세스의 내부행위는 FSM(Finite State Machine)에 덧붙여 상태전이시의 데이터 연산을 표현할 수 있는 EFSM(Extended Finite State Machine) 모델을 기반으로 하고 있다[5]. 프로세스들 사이의 상호작용은 프로세스들 사이에 시그널전송과 그 시그널에 대해 종속적으로 정의된 데이터의 교환에 의해서 기술할 수 있다. 그리고 여러 개의 프로세스를 포함할 수 있는 블록이라는 단위로 시스템의 계층성을 표현한다. 따라서 추상적인 SDL모델로부터 실제 소프트웨어로 구현될 코드를 자동으로 생성하기 위해서는 실제 생성된 코드가 수행될 실행환경에 적합하도록 원래 사양기술에 나타나지 않은 묵시적인 요소들을 지정해 주는 것이 필요하다. 본 논문에서 제안한 SDL2C코드 생성기로부터 생성된 C코드는 다음과 같은 실행환경을 가정하고 있다.

- SDL기술에서의 계층구조 (프로세스-블록-시스템)는 평탄화(flattening)되어 C코드에서 표현된다.
- 값의 범위에 제한이 없는 SDL자료형은 변환된 C언어의 자료형의 범위로 값이 제한된다.
- SDL 수행모델에서 각 프로세스마다 존재한다고 가정되는 무한 크기의 입력버퍼는 생성되는 C언어에서 유한크기의 큐로 구현된다.

2.2 SDL프로세스의 C 구문으로의 변환

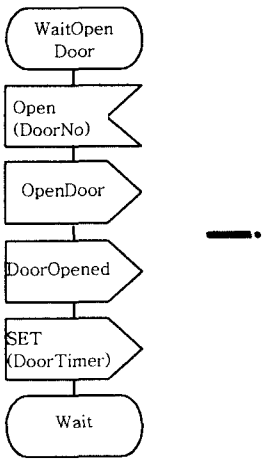
하나의 프로세스는 EFSM동작을 표현한다고 앞서 설명하였다. 이 FSM구조는 C언어로 표현할 때에 중첩된 switch - case 문을 이용하여 표현가능하고, 상태전이시에 발생하는 데이터연산은 그와 동일한 C언어에서의 연산으로 표현하여 한 SDL프로세스의 동작을 C로 표현하였다. SDL 프로세스 내부에서의 한 상태전이를 C코드로 변환한 결과를 그림2에서 보여 주고 있다. 그림2의 왼쪽 부분은 SDL프로세스에서의 한 상태전이를 그래픽표현으로 나타낸 것이다. 오른쪽의 변환된 C코드부분에서 GetSignal(), PutSignal()함수는 운영체제의 메시지메일박스와의 데이터 송수신을 위한 함수로 SDL시그널 교환을 구현한 것이며, Get_from_FIFO() 와 Put_into_FIFO() 함수는 한 시그널과 함께 교환되는 데이터를 프로세스의 입력큐로부터 읽거나 입력큐에 쓰는 함수이다.

3. 실시간 운영체제 환경에 대한 고려

SDL2C 변환기는 SDL구문을 C의 구문으로 변환해주는 구문중심적(syntax oriented)번역을 수행하기보다는 생성될 C코드가 수행될 환경을 고려하여 C코드를 생성한다. 대부분의 실시간운영체제의 스케줄링방식이 선점형 우선순위기반의 스케줄링 기법을 사용하는 것을 고려하여 SDL의 다중프로세스 수행모델이 실제 운영체제상에서 프로세서 스케줄링을 통한 다중프로세스로 수행이 되도록 구현하였고, SDL의 시그널 송수신을 의미상 대응하는 운영체제의 IPC함수를 이용하여 구현하였다.

3.1 운영체제 IPC서비스 함수의 사용한 SDL시그널전송의 구현

SDL수행모델에서 프로세스간의 상호작용은 상태전이시 발생하는 출력시그널과 그 시그널에 종속되어 정의된 데이터를 다른 프로세스로 전송함으로서 이루어진다. 그림3과 같이 운영체




```
case  ST_WaitOpenDoor_DoorController :
    InSignal = GetSignal(SELF, &No_Arg);
    switch (InSignal)
    {
        case SG_OpenDoor :
            Get_from_FIFO((void *)(&SENDER), SELF);
            Dest_Pid =  Pid_tbl[P_Env];
            Put_into_FIFO(sizeof(SELF), (void *)&SELF, Dest_Pid);
            Put_into_FIFO(sizeof(DoorNo), (void *)&DoorNo, Dest_Pid);
            PutSignal(SG_Open, Dest_Pid, 2);
            Dest_Pid =  Pid_tbl[P_Controller];
            Put_into_FIFO(sizeof(SELF), (void *)&SELF, Dest_Pid);
            PutSignal(SG_DoorOpened, Dest_Pid, 1);
            TmrSetT(DoorTimer_ID, (int)(DoorTimer*10.0));
            TmrStart(DoorTimer_ID);
            NextState = ST_Wait_DoorController;
            break;
            default :
            CurrentState = ST_WaitOpenDoor_DoorController;
            delete_FIFO(No_Arg, SELF);
            break;
    }
break;
```

그림2. SDL 프로세스내에서 한 상태전이의 C코드 변환

제상의 메시지메일박스와 FIFO를 이용하여 이러한 프로세스간 상호작용을 구현하였다.

3.2 우선순위기반 스케줄링에서 SDL수행모델의 적용

SDL프로세스의 내부행위는 EFSM모델에 기반하고 있으므로 상태전이-입력시그널대기 의 두 단계를 반복하게 된다. 우선순위기반 스케줄링기법에서는 우선순위가 가장 높은 하나의 프로세스가 CPU시간을 얻어 수행되며, 실행중인 프로세스가 시그널 혹은 세마포어가 활성화되기를 기다리게 되거나 프로세스지연 함수등이 실행되면 대기상태로 전환된다. 그 후에는 다음 우선순위를 가지며 준비리스트에 속한 프로세스가 CPU시간을 할당받아 실행되게 된다. 이러한 우선순위기반 스케줄링기법에서 SDL 다중프로세스 수행모델은 다음과 같은 방식으로 실행된다.

1. 현재 우선순위가 가장 높은 프로세스가 수행도중 상태전이가 발생하여 출력시그널을 전송한다.
2. 출력시그널을 전송한 후 최우선순위 프로세스는 입력시그널을 대기하며 프로세스 대기상태로 전환되어 다음 우선순위를 가지는 프로세스가 수행될 수 있는 상태가 된다.
3. 차우선순위를 가진 프로세스가 대기상태에서 상태전이를 일으키는 시그널을 입력받았을 때에 프로세스의 상태는 실행상태가 되어 프로세스는 스케줄러에 의해 실행된다.
4. 차우선순위를 가진 프로세스가 대기상태에서 상태전이를 일으키는 시그널을 입력받지 못했다면 우선순위기반 스케줄링방법에 따라 그 다음 우선순위를 가지는 프로세스가 실행이 된다.

그림4 에 이러한 시나리오가 표현되어 있다. 이 그림의 각 프로세스의 실행스레드에서 실선은 프로세스의 수행상태, 점선은 프로세스가 대기상태에 있음을 나타낸고, s1부터 s6는 각 프로세스가 주고 받는 시그널을 나타낸다. 이와 같이 우선순위기반 운영체제상의 소프트웨어로 구현된 SDL프로세스는 원래의 SDL사양기술에서 교착상태(deadlock)가 발생하지 않는다면 변환된 소프트웨어 환경에서도 교착상태가 발생하지 않는다는 것이 보장된다.

4. 코드변환 예제

SDL 사양기술로부터 C코드를 생성하는 SDL2C 변환기의 입력으로 Access Control 시스템을 사용하였다. 이 시스템은 출입문 접근시에 카드키를 이용하여 인증된 사용자에게 출입을 허용하는 시스템으로 전형적인 제어중심 시스템이다. SDL2C 변환기를 통하여 C코드를 생성한 후 ARM7TDMI를 코어로 하고 uC/OS-II 운영체제로 하는 내장형시스템 아키텍쳐에서 생성된 C코드를 내장형 시스템환경에서 디버깅 및 실행하였다. 생성된 C코드의 컴파일 및 디버깅에는 ARM Development Suite상의 C컴파일러와 AXD(Arm Extended Debugger)를

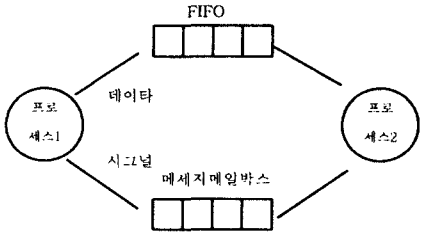


그림3 SDL시그널 전송의 소프트웨어 구현

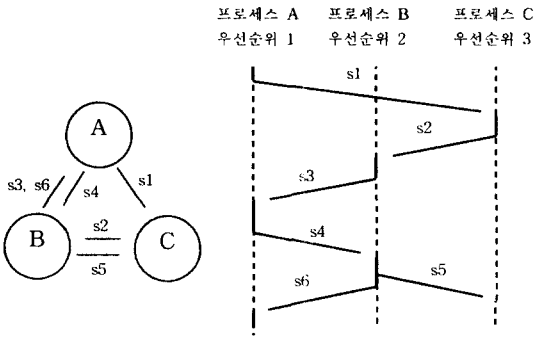


그림 4. 시그널송수신에 따른 우선순위기반 스케줄링 운영체제에서의 프로세스 상태변화

이용하였다.

5. 결론

SDL사양기술로부터 실시간 운영체제에서 수행될 C코드를 자동으로 생성해주는 방법을 제시하였다. 본 논문에서 제시한 SDL2C 변환 방법은 현재 내장형시스템에서의 소프트웨어 개발환경을 고려하여 실시간운영체제의 프로세스 스케줄링과 적절히 융화될 수 있는 기본틀을 제시하고 그에 맞게 실시간운영체제 서비스 함수들을 이용하여 C코드를 생성한다. 따라서 생성된 C코드는 원래의 SDL사양과 동일한 동작을 수행하며, 효율적으로 운영체제 기능들을 이용하며 수행된다. 현재는 초기사양을 동일한 동작을 수행하는 C코드 생성에 주안점을 두고 있지만, 생성된 소프트웨어 시스템의 응답성 및 CPU이용도 등을 척도로 하는 고성능 실시간 소프트웨어 생성이 향후 개선사항이다.

6. 참고문헌

[1] Daniel D. Gajski, Frank Vahid, Sanjiv Narayan, and Kie Gong, Specification and Design of Embedded System, Prentice Hall , 1994

[2] J. Staunstrup, W. Wolf, Hardware/Software Co-Design : Principles and Pracitce, Kluwer Academic Publishers, 1997

[3] Y. Huang M. Hughes, Using SDL in embedded systems design: a tool for generating real-time OS pSOS based embedded systems applications software, *Proc. of the 11th IEEE Workshop on Real-Time Operating Systems and Software*, pp. 39 -43, 1994

[4] J.M. Alvarez, M. Diaz, L. Llopis, E. Pimentel, J.M. Troya, Deriving hard real-time embedded systems implementations directly from SDL specifications, *Proc. of the 9th International Symposium on Hardware/Software Codesign, 2001*. pp. 128 133, 1994

[5] J. Ellsberger, D. Hogrefe, A. Sarma, SDL : Formal Object-oriented Langauge for Communication Systems, Prentice Hall, 1997