임베디드 컴퓨터에서 프로세서 변경에 따른 영향 분석

Impact Analysis of the Processor Alteration on Embedded Computer

김 형 문*
Kim, Hyung-Moon

ABSTRACT

The ubiquitous embedded computers are firmly established as the basic electronic component of design that control military systems. Such applications can be found everywhere in the field of military system. A embedded computer is required to redesign when system needs performance upgrade or production-state of processor is NRND or EOL. This paper describes a scheme about impact analysis of designing processor alteration on embedded computer. In this case, hardware architecture and interrupt source of target system must be considered. Also, performance and throughput of that must be analyzed.

주요기술용어(주제어) : Embedded Computer(임베디드 컴퓨터), NRND(Not Recommended for New Design), EOL(End of Life), CPU Throughput(명령어 처리량)

1. 서 론

시스템을 제어하거나 감시하기 위해 시스템에 내장하는 컴퓨터를 의미하는 임베디드 컴퓨터는 현대식 무기체계에서 필수적인 구성 장치로 자리 잡았다. 임베디드 컴퓨터를 기반으로 하는 무기체계는 범용 컴퓨터와는 달리 특별한 목적 및 기능에 적합하도록 중앙집중식에서부터 분산 병렬 구조의 임베디드 컴퓨터까지 최적화되게 설계되어 진다.

인텔사의 창립자인 무어(Gordon Moore)가 제언한 "새롭게 개발되는 마이크로 칩의 처리능력은 18개월마다 2배로 증가 된다"는 무어의 법칙과 같이, 초기 CPU(Central Processor Unit)가 개발된 이후로 수백 종류의 프로세서가 개발되었다가 사라지고, 앞으로도 새로운 모델 및 고성능 프로세서들이 우리 주위에 다가오고 또 멀어져갈 것이다. 현재에도 고속처리가 가능하고, 저소비 전력, 다양한 주변장치내장, 특수 전용 장비를 겨냥한 다양한 프로세서가 매우 빠른 속도로 변화되어지고 있다.

개발 및 운용에 수십 년의 수명주기를 지닌 무기체계에 비해 프로세서의 발전 속도가 너무 빨라 무기체계의 개발 및 유지에는 단점이 된다. 현재도 개발 또는 운용 단계에서 제품 단종으로 인하여 임베디드 컴퓨터 변경이 자주 발생하게 된다.

이러한 경우 기 개발 또는 운용되는 소프트웨어 재사용을 극대화가 가능한 임베디드 컴퓨터의 재설계가 요구되며, 또한 전체시스템에 대한 영향을 가능한 줄이기 위해서 설계 변경의 범위를 최소화해야 한다.

이러한 관점에서 본 논문은 프로세서 단종으로 인하여 임베디드 컴퓨터의 프로세서 변경 설계 시에

† 2007년 3월 21일 접수~2007년 5월 18일 게재승인
* 국방과학연구소(ADD)
주저자 이메일 : ahwa20@daum.net

고려해야할 사항들을 도출하고 프로세서 변경에 따른 임베디드 컴퓨터의 영향 분석 방안을 제시하고자 한다.

제작사에서 제시하는 프로세서의 성능지표를 비교하면 당연히 최근에 개발된 프로세서의 성능이 월등하나 이는 단순히 프로세서가 발휘할 수 있는 최대의 성능을 제시해 줄뿐이다. 실제 임베디드 컴퓨터에 구현된 상태에서의 성능은 시스템에서 필요로 하는 주변회로에 따라 결정되므로 프로세서 변경 설계에 따른 영향 분석은 반드시 필요하다.

본 논문에서는 SMJ320C40을 채택하고 있는 제어 컴퓨터의 프로세서를 같은 제작사의 최신 프로세서인 SMJ320C6701(이하 C6701)로 변경 설계 시 임베디드 컴퓨터의 영향 분석을 수행하였다. C40의 Industrial Application용 제품인 TMS320C40은 이미 단종 되었고, Military Application용 제품인 SMJ320C40(이하 C40)는 현재 생산 중이지만 제품 단종이 예정되어 있다. SMJ는 'MIL-PRF-38535'를 만족하여 Military Application에 사용할 수 있는 직접회로를 의미한다.

2. 임베디드 컴퓨터의 아키텍쳐와 인터럽트 소스

임베디드 컴퓨터는 사용용도, 수행업무, 사람이나 다른 시스템과 어떻게 상호 작용하는가에 따라 아키텍쳐(Architecture)와 아키텍쳐에 따른 인터럽트 소스(Interrupt Source)가 결정된다.

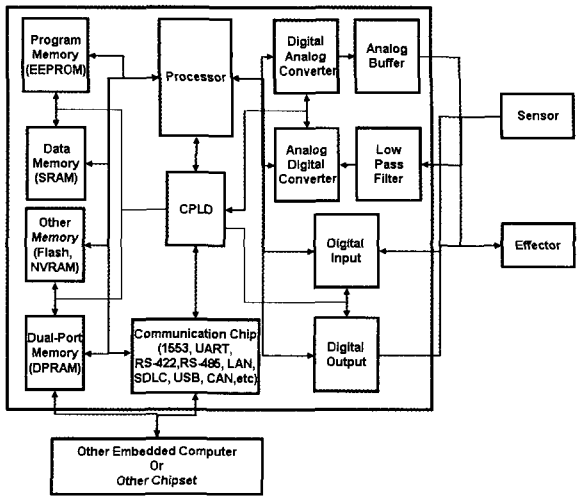
가. 아키텍쳐 및 기능[1,2]

임베디드 컴퓨터는 대상 시스템을 모니터하고 제어하기위해 Sensor와 Effector들을 사용하는 실시간용 하드웨어와 소프트웨어로 구성되는 특징을 지닌다.

제어용 하드웨어 아키텍쳐는 프로세서와 메모리를 기본 구성요소로 하며, 사용목적에 따라 여러 가지 형태의 입력과 출력을 가지는 주변회로와 인터페이스 된다.

일반적인 임베디드 컴퓨터의 구성도는 그림 1과 같고, 주변회로의 기능은 다음과 같다.

- Program Memory : 컴퓨터 초기화 프로그램 및 실행 프로그램 코드를 저장.



[그림 1] 임베디드 컴퓨터의 구성도

- Data Memory : 프로그램 연산에 필요한 변수와 데이터를 쉽게 쓸 수 있도록 임시 저장.
- Flash Memory : 시스템 매개변수, 환경설정, 파라미터 저장에 사용.
- NVRAM : 시스템 상태정보 및 서비스 기록.
- DPRAM : 다른 임베디드 시스템 혹은 칩셋과의 데이터 교환에 사용.
- CPLD : 프로세서와 주변회로의 버스/컨트롤 신호 타이밍 조절, 어드레스 디코더, 인터럽트 관리 등.
- Digital Input : 주변장치의 디지털 상태 입력, IO장치의 스위치상태, 버튼 누름 상태 입력.
- Digital Output : 시스템의 상태 전달, 외부 장치를 끄거나 켬. 다른 시스템과 통신에서 Input/Output의 조합을 통해 프로토콜 및 인터페이스를 동기화.
- Analog Input : 아날로그의 Sensor 값을 디지털 값으로 변환(온도, 속도, 기압, 수압, 습도, 전압, 전류 등).
- Analog Output : 시스템의 디지털 출력값을 아날로그 값으로 변환하여 Effector에 전달.
- Communication Chip : 호스트 컴퓨터, 모뎀, 다른 임베디드 시스템, 네트워크 등과 통신.
- Sensor : 대상 시스템의 측정된 물리적인 특성을 컴퓨터의 입력을 위해 상응하는 전기적인 신

호로 변환.

- Effector : 컴퓨터의 출력인 전기적 신호를 대상 시스템의 Function을 제어하기위해 상응하는 물리적인 작용으로 변환.
- Wrap-around : Effector 고장을 검출.
- End-around : Input/Output의 무결성을 입증.

나. 인터럽트 소스[2~4]

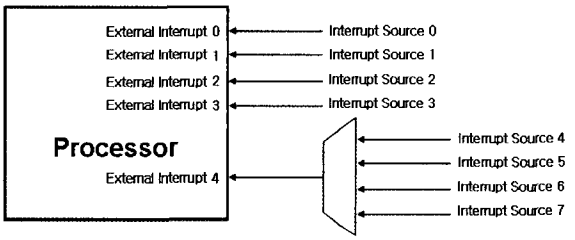
인터럽트는 시스템에서 발생하는 이벤트들을 처리하기 위해 CPU가 현재 수행중인 프로그램에서 분기하는 기법을 말한다. 발생할 수 있는 이벤트는 주변장치에서 발생한 오류일 수도 있고, 단순히 주변장치가 맡은 작업을 끝내고 다음 작업을 수행할 준비가 되었다는 신호일 수도 있다. 주변장치는 주의가 필요한 상황이 발생하면, 인터럽트 소스 중의 하나를 이용해서 CPU에 알린다.

아키텍쳐에 따라 인터럽트 소스가 결정되며, 인터럽트 소스는 하드웨어 소스와 소프트웨어 소스로 나누어진다.

1) 하드웨어 인터럽트 소스

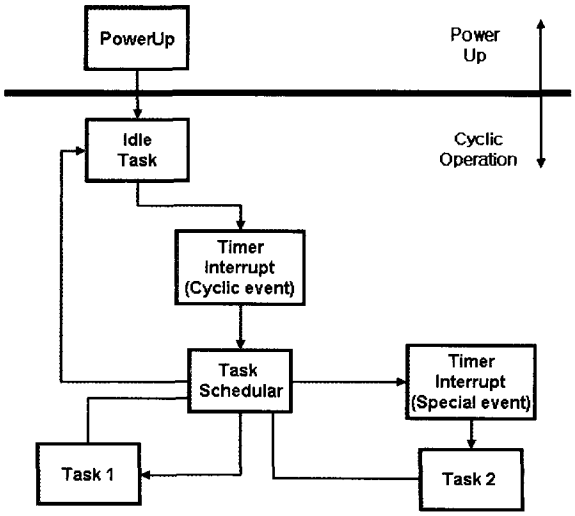
프로세서 내부모듈 및 외부의 주변회로에서 발생하는 이벤트들로 다음과 같은 인터럽트 소스들이 있다.

- Power Down : 공급전원이 정해진 기준 전압이하로 다운될 경우 Power Monitor에 의해 발생.
- Memory Parity Error : 주요정보가 저장되는 프로그램, 데이터 메모리의 쓰기 동작 시 각 메모리에 할당된 Parity 메모리에 Parity 정보를 저장하며, 읽기 동작 시 메모리 내용과 Parity 정보를 비교하여 오류 발생 시 발생.
- Watch dog timer : CPU의 비정상 동작을 감시하며 특정 시간 내에 timer가 clear되지 않을 때 발생. 프로세서의 내부 모듈을 사용할 수도 있다.
- Illegal address : 메모리 맵의 "unmapped" 또는 "not used" 영역에 CPU의 접근 시 발생.
- Self-test : 외부 모니터링 장비에 의한 Self-test요청 시 발생.
- DPRAM update : DPRAM을 통해 인터페이스 되어있는 다른 임베디드 시스템 혹은 칩셋의 데이터 업데이트 완료 시 발생.
- DMA R/W : DMA를 통한 메모리 Read/Write 관련 이벤트 시 발생.
- Communication chip Tx/Rx : 통신 칩의 송신/수신 관련 이벤트 시 발생.
- ADC EOC : 아날로그 입력의 디지털 변환 완료 시 발생.
- Capture : 비주기성 입력의 신호천이 시 발생.
- External Interrupt Multiplex : 프로세서가 제공하는 외부 인터럽트의 수를 초과하는 인터럽트 소스를 처리할 경우에 그림 2와 같이 사용.



[그림 2] External Interrupt Multiplex

- Timer : 그림 3과 같이 제어 루틴 수행을 위해 설정된 주기 경과 시 발생하며, Task Scheduler에 의한 특정 Task를 수행하기 위한 이벤트용으로도 발생.



[그림 3] Task Scheduler 구성도

2) 소프트웨어 인터럽트 소스

프로그램 수행 시 오류 검출 및 특정 루틴을 수행하기위해 사용하는 이벤트들로 다음과 같은 소스들이 있다.

- Stack Overflow : Stack의 Overflow 조건 시 발생.
- Trap : 프로그램 메모리의 "used" 영역 중에 사용하지 않는 부분에 CPU의 접근 시 발생.
- User-define : 특정 매개변수가 지정된 값에 일치할 경우 설정한 프로그램 루틴을 수행하기 위해 이벤트 발생.

3. 임베디드 시스템의 Throughput 예측

시스템의 제어주기와 그 제어주기 내에 실행되어야 할 소프트웨어의 양에 따라 CPU Throughput이 결정된다. 그리고 Throughput을 근거로 임베디드 컴퓨터에 사용할 프로세서와 동작 클럭(clock)을 결정하게 된다. 프로세서 변경 설계를 하고자 한다면, 제어주기와 소프트웨어의 양 이외에도 프로세서 변경에 따르는 다음 사항을 고려해야한다.

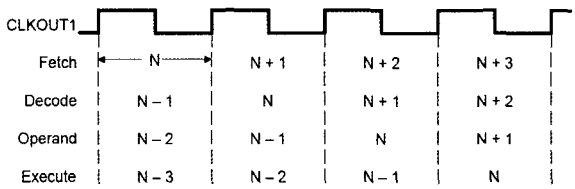
가. 메모리 타입, 속도, 인터페이스의 영향

메모리들의 타입, 속도, 인터페이스 방식에 따라 Throughput은 크게 달라진다. 규정된 시간 이상의 버스/컨트롤 신호 인가를 요구하는 비동기식 메모리는 클럭 속도에 따라 억세스 속도를 높일 수 있는 동기식 메모리에 비해 Throughput이 떨어진다. 같은 비동기식 메모리를 사용하더라도 보다 짧은 억세스 시간을 요구하는 메모리를 사용할수록 Throughput이 향상된다. 프로그램 메모리와 데이터 메모리를 동시에 접근할 수 있는 이중 버스 구조는 하나의 메모리만 접근할 수 있는 단일 버스 구조에 비해 Throughput이 향상된다.

나. Pipeline 구조에 따른 Conflict[3,4]

하버드 구조를 채용하고 있는 프로세서들은 하버드 구조의 이점을 활용할 수 있도록 CPU Core가 Pipeline 구조를 가지고 있고, 명령어 실행과정에 있어서 Pipeline을 최대로 활용하지 못할 때 Conflict가 발생할 수 있다. Pipeline Level이 클수록 Conflict는 다양하고 복잡하게 나타난다.

하버드 구조를 가지는 프로세서는 내부적으로 프로그램 버스와 데이터 버스가 분리되어있고, 이로 인해 CPU는 프로그램과 데이터에 동시에 접근할 수 있다. 따라서 하버드 구조의 이점을 활용할 수 있도록 프로그램 실행에 필요한 각 단계들을 쪼개어 다음 단계가 실행되는 동안 미리 이전의 단계를 수행시키는 방법을 통해서 여러 개의 명령어를 동시에 처리하는 Pipeline 구조의 채용을 통해 CPU의 성능을 최대한 발휘할 수 있다. 그림 4는 4-Level Pipeline Operation을 보여주고 있다.



[그림 4] 4-Level Pipeline Operation

그러나 Pipeline 구조로 인해 Branch Conflict, Register Conflict 그리고 Memory Conflict의 3가지 Pipeline Conflict가 발생한다. Pipeline Conflict의 발생빈도는 프로그램에 의해 좌우되므로, 작성한 실행 프로그램을 최적화하여 발생빈도를 가능한 줄이는 노력을 기울여야 한다.

1) Branch Conflict

프로그램 카운터를 읽거나 혹은 읽어서 변경하는 명령어의 실행 혹은 동작의 경우 발생한다. Branch, Call 같은 명령어가 이에 해당하며, 명령어 실행 클럭 수보다 하나 적은 클럭 수만큼의 Conflict가 발생한다.

2) Register Conflict

연속되는 명령어의 OP code에 어드레스 발생을 위해 사용하는 CPU의 보조 레지스터 중에 같은 레지스터의 읽기나 쓰기가 동시에 포함되어 있을 때, Conflict가 발생하게 된다.

3) Memory Conflict

프로세서 내부 메모리 혹은 외부 메모리의 억세스가 완료되지 않은 상태에서 다음 명령어에 의해 다시 억세스를 시도하는 경우 이전 메모리 억세스가 완료 될 때까지 Conflict가 발생한다. 그리고 억세스 시간이 긴 외부메모리의 경우 억세스가 완료될 때까지 전체 Pipeline 동작이 중지되는 Conflict가 발생한다.

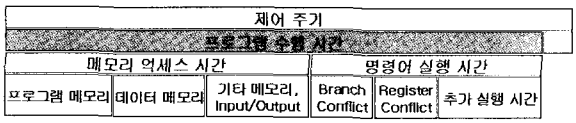
CPU의 Pipeline 구조가 복잡할수록 위 3가지의 Pipeline Conflict가 다양하고 복잡하게 나타나게 된다. 게다가 CPU의 버스구조, 사용하는 메모리의 타입과 억세스 시간, 프로그램 최적화에 대한 Pipeline Conflict의 영향은 더욱 커지게 된다.

아래의 3가지 프로세서 변경 설계 조건하에서는 2), 3)항의 Conflict 변화량은 무시할 수 있을 정도로 매우 적을 것으로 판단되며, CPU의 Pipeline 레벨에 따른 Branch Conflict 변화량만 고려하면 된다.

① 기존 임베디드 컴퓨터의 프로세서만 변경 설계.
② *PowerUp 시의 하드웨어 초기화를 제외한* 모든 프로그램의 재사용.
③ 같은 제작사의 프로세서로 변경함으로서 프로그램 컴파일 최적화 정도를 동일하게 유지.

다. CPU Throughput 예측 방안

앞에서 언급한 프로세서 변경 설계 조건하에서 Throughput 예측 방안을 제시한다. 일반적인 시스템의 제어주기내의 프로그램 수행시간은 그림 5와 같이 구성된다. Memory Conflict는 각 메모리별 억세스 시간에 포함되어 있다.



[그림 5] 프로그램 수행시간 구성도

메모리 억세스 시간(t_{mem})과 명령어 실행시간(t_{inst})은 다음과 같이 표현된다.

$$t_{mem} = \sum_{i=0}^{n-1} (R_{ci} \times t_{racci} + t_{ci} + W_{ci} \times t_{wacci} + t_{wmci}) \quad (1)$$

$$t_{inst} = B_c \times B_{dk} \times t_{dk} + R_c \times t_{dk} + N_{inst} \times t_{dk} \quad (2)$$

여기서, i는 메모리 및 I/O 소자, R_{ci}/W_{ci}는 i의 읽기/쓰기 개수, t_{racci}/t_{wacci}는 i의 읽기/쓰기 억세스 시간, t_{rmci}/t_{wmci}는 i의 읽기/쓰기 시에 발생하는 Memory Conflict 시간, B_c/R_c는 Branch/Register Conflict 개수, B_{clk}는 Branch 관련 명령어 수행에 필요한 추가 클럭 개수, t_{clk}는 메인 클럭 속도, N_{inst}는 명령어 수행을 위한 추가 클럭 수이다.

따라서 기존 임베디드 컴퓨터에서 위 (1), (2) 식의 인자를 구해내면 프로세서 변경 후의 Throughput을 예측할 수 있다.

R_{ci}/W_{ci}, B_c는 기존 임베디드 컴퓨터를 동작시킨 후 제어주기 동안 프로그램이 메모리 및 I/O 소자를 억세스 하는 개수를 Logic Analyzer를 통해 어드레스 버스를 측정함으로서 카운팅할 수 있다. t_{racci}/t_{wacci}, t_{clk}는 메모리, I/O 소자 그리고 프로세서의 Datasheet를 통해서 계산할 수 있다. t_{mci}, R_c, N_{inst}는 기존 인자에 대한 변화량을 거의 무시할 수 있으므로 곱하는 t_{clk}을 프로세서 변경 후의 값으로 대입하여 계산할 수 있다. 추가로 B_c는 CPU의 Pipeline 레벨에 따라 Branch Conflict가 변할 수 있으므로 프로세서 변경에 따른 Pipeline 레벨의 변화를 고려해야 한다.

4. CPU 변경에 대한 영향 분석

변경할 프로세서 선정은 시스템의 목적, 성능, 요구조건을 고려해야한다. 프로세서가 선정되면 대상 임베디드 컴퓨터의 아키텍쳐 및 인터럽트 소스로부터 하드웨어/소프트웨어 관점의 변경 타당성에 대한 분석을 실시하고, Throughput 예측을 통해 프로세서 변경에 대한 영향을 분석한다.

가. 변경할 CPU 선정

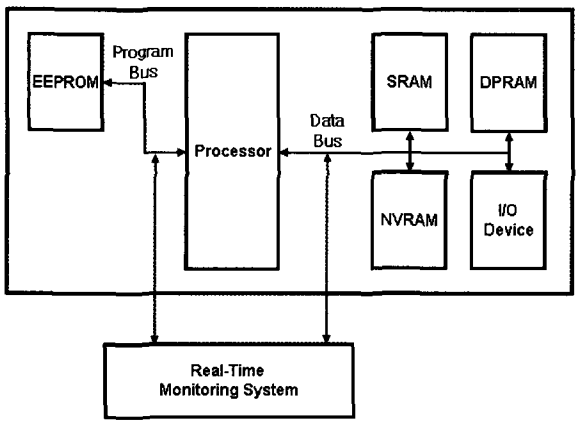
프로세서 변경으로 인한 전체시스템의 설계 변경 영향을 가능한 최소화하기 위해 프로세서 변경 설계

조건에 프로세서 선정 시 고려해야할 아래의 5가지 조건을 더해 판단한 결과, TI 社의 C6701로 선정하였다.

① 'MIL-PRF-38535'를 만족.
② C40 기본성능의 동등 혹은 이상의 성능.
③ 기존 인터럽트 소스 지원.
④ 프로세서 Self-test 프로그램 지원.
⑤ 기존 Unit-test 프로그램의 재사용 가능 극대화.

나. 대상 임베디드 컴퓨터의 아키텍쳐

대상 임베디드 컴퓨터의 아키텍쳐는 그림 6과 같으며 프로세서와 메모리 이외의 부분은 생략하였다.



[그림 6] 대상 임베디드 컴퓨터의 아키텍쳐

대상 임베디드 컴퓨터는 TI 社의 C40 프로세서를 채택하고 있으며, 사용하고 있는 메모리는 모두 외부 메모리이며, 비동기식이다. 실시간 외부 모니터링 시스템으로부터 컴퓨터 정상동작 여부를 검증하기 위해서는 외부 메모리를 사용해야 한다. 실시간 모니터링 시스템의 메모리 인터페이스 회로 역시 임베디드 컴퓨터에서 사용하고 있는 비동기 메모리에 적합하게 설계되어 있다.

다. 대상 임베디드 컴퓨터의 인터럽트 소스

대상 임베디드 컴퓨터의 하드웨어 인터럽트 소스는 Power Down, Memory Parity Error, Watch dog timer, Illegal address, External Interrupt Multiplex의 5개의 외부 인터럽트 소스가 필요하며, 프로세서 내부의 2개의 Timer 소스가 필요하다. 소프트웨어 인터럽트 소스는 Stack Overflow, Trap 인터럽트가 필요하다.

라. 하드웨어/소프트웨어 관점의 변경 타당성

표 1은 C40과 C6701의 성능 비교표이다.[5,6] 표 1과 같이 시스템에서 요구하는 기능에 대해 C40의 기본성능을 만족하며, 기존 인터럽트 소스도 지원할 수 있으며, 'MIL-PRF-38535'를 만족하고 있음을 알 수 있다. 같은 회사의 컴파일 프로그램을 사용함으로서, 컴파일 프로그램에 대한 의존도가 높았던 기존 Unit-test 프로그램의 재사용 가능성도 극대화할 수 있다.

[표 1] C40과 C6701 성능 비교표

Device	C40	C6701	비고
Type	Floating-Point	Floating-Point	
FLOPS (최대클럭기준)	• 명령어 수행시간 : 33-ns • 60 MFLOPS, 30 MIPS, 330 MOPS	• 명령어 수행시간 : 6-ns • 최대 1 GFLOPS	
최대 동작 클럭	60Mhz	167Mhz	
인터페이스 메모리	비동기	비동기, 동기	
내부 메모리	2K	Prog64Kb/ Data64Kb	
외부 메모리 확장	32bit(4G)	32bit (52Mb, 4CE)	
MIL-PRF-38535	만족	만족	
JTAG 지원	IEEE 1149.1 지원	IEEE 1149.1 지원	
타이머 인터럽트	2개(32bit)	2개(32bit)	
외부 인터럽트	5개	5개	
Trap 인터럽트	지원	지원	
Serail Port	통신포트 6개	McBSP 2개	사용안함
Peripheral Port	-	16bit HPI	사용안함
DMA(channels)	6 channels	4 channels	사용안함

프로세서 Self-test를 통해서 프로세서의 무결성 입증 및 고장 검출을 할 수 있으며, 시스템이 요구하는 안정성을 확보할 수 있다. 프로세서 Self-Test는 반도체 생산 공정의 모든 결함을 찾는 전체점검과는 달리 CPU 명령어 수행을 통하여 점검되며, 일종의 Confidence Check이다. 프로세서 Self-Test에서 점검되어야 할 항목은 CPU Core 및 Internal memory와 On-chip Peripheral(DMA, Timer 등)들이며, 각각 명령어 조합 수행을 통해 각 항목들이 점검된다.

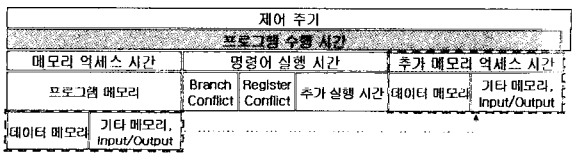
TI 社에서 제공하는 C6701의 Self-test 프로그램은 없으나, C6000계열 중 제공하는 C6201 및 C6416 Self Test 프로그램의 수정을 통해 C6701의 Self-Test 프로그램을 개발할 수 있다. 따라서 C6701로 변경함에 있어서 하드웨어/소프트웨어적으로 타당함이 확인되었다.[7~9]

마. CPU Throughput 분석

1) 메모리 인터페이스에 따른 Throughput 영향[3]

C40은 2개의 동일한 외부버스를 가지고 있다. Local-Bus는 프로그램 메모리와 Global-Bus는 데이터 메모리 및 기타 메모리와 각각 인터페이스 된다. 이 구조는 Local-Bus와 Global-Bus를 동시에 억세스 할 수 있도록 하며, 외부 메모리를 사용하는 아키텍쳐의 경우 더욱 높은 Throughput을 낼 수 있게 한다. 그러나 C6701은 내부 메모리 사용에 적합하게 설계되어있기 때문에 외부 메모리를 꼭 사용해야하는 현 아키텍쳐에 적용할 경우, 그림 7, 8과 같이 추가의 메모리 억세스 시간이 필요하게 된다.

[그림 7] C40 프로그램 수행시간 구성도



[그림 8] C6701 프로그램 수행시간 구성도

2) 메모리 억세스에 따른 Throughput 영향[3,4,10~15]

C40 사용 시 메인 클럭 주파수는 10MHz(100nS)를 사용하였고, 읽기에 한 클럭, 쓰기에 2 클럭이 필요한 특성을 가지고 있다. EEPROM, SRAM을 제외한 소자에는 Timing Margin을 고려하여 Wait-state를 한 클럭씩 더 인가하였다. 기존 임베디드 컴퓨터의 메모리 읽기/쓰기 억세스 시간은 표 2와 같다.

[표 2] C40 사용 시 메모리 억세스 시간

종 류	Part number	읽 기	쓰 기
EEPROM	AM29F800B-55nS	100nS	-
SRAM	EDI816256CA-17nS	100nS	200nS
DPRAM	IDT7025-20nS	200nS	300nS
NVRAM	STK11C68-25nS	200nS	300nS
I/O Device	-	200nS	300nS

[표 3] Timing Parameter

파라미터	정 의
t_{isu}	Data setup time, read D before CLKOUT1 high
t_h	Data hold time, read D after CLKOUT1 high
t_d	Output delay time, CLKOUT1 high to output signal valid
$t_{xw(m)}$	Time from control/data signals active to /AWE inactive
$t_{wp(m)}$	Write pulse width
$t_{ih(m)}$, $t_{wr(m)}$	Maximum of either write recovery time or data hold time
$t_{rc(m)}$	Length of the read cycle
$t_{wc(m)}$	Length of the write cycle
$t_{acc(m)}$	Access time, from EA, /BE, /AOE, /CE active to ED valid
$t_{oh(m)}$	Output hold time

C6701의 경우 비동기 메모리 읽기/쓰기를 위한 버스/컨트롤 신호는 Setup, Strobe, Hold의 구간으로 구성되며 각 구간을 결정하기 위한 식은 (3)~(9)와 같고, Timing Parameter는 표 3과 같다. 메모리 억세스 시간은 Setup, Strobe, Hold의 시간을 모두 합한 시간이 된다.

- 읽기 억세스 시 Setup, Strobe, Hold 계산식

$$\text{Setup+Strobe} \geq (t_{acc(m)} + t_{su} + t_{dmax})/t_{cyc} \quad (3)$$

$$\text{Setup+Strobe+Hold} \geq (t_{rc(m)} + t_{skew})/t_{cyc} \quad (4)$$

$$\text{Hold} \geq (t_h - t_{dmin} - t_{oh(m)})/t_{cyc} \quad (5)$$

- 쓰기 억세스 시 Setup, Strobe, Hold 계산식

$$\text{Strobe} \geq t_{wp(m)}/t_{cyc} \quad (6)$$

$$\text{Setup+Strobe} \geq (t_{xw(m)} + t_{skew})/t_{cyc} \quad (7)$$

$$\text{Hold} \geq (\max(t_{ih(m)}, t_{wr(m)}) - t_{skew})/t_{cyc} \quad (8)$$

$$\text{Setup+Strobe+Hold} \geq (t_{wc(m)} + t_{skew})/t_{cyc} \quad (9)$$

C6701의 메인 클럭을 100MHz(10nS)로 선정하고 Timing Margin을 고려했을 때, 식 (3)~(9)를 이용해 계산한 메모리 억세스 시간은 표 4와 같다.

[표 4] C6701 사용 시 메모리 억세스 시간

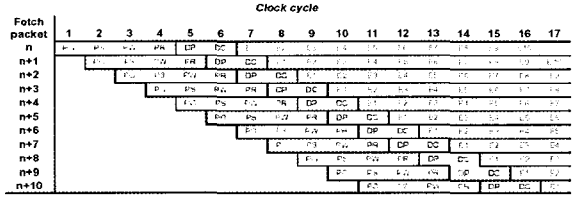
종 류	Part number	읽 기	쓰 기
EEPROM	AM29F800B-55nS	100nS	-
SRAM	EDI816256CA-17nS	60nS	60nS
DPRAM	IDT7025-20nS	70nS	70nS
NVRAM	STK11C68-25nS	100nS	100nS
I/O Device	-	100nS	100nS

C40은 ALU(Arithmetic Logic Unit), Multiplier 2개의 Function Unit으로 구성되어 있으나, C6701은 동시에 최대 8개의 명령어를 처리할 수 있도록 8개의 Function Unit이 구현되어 있는 구조적인 특징을 가지고 있다. 그러므로 8개의 명령어를 동시에 공급할 수 있도록 프로그램 메모리를 한번에 8번을 연속 Fetch한다.

C6701은 비동기 메모리의 읽기 억세스 시에는 추가 클럭 7개, 쓰기 억세스 시에는 추가 클럭 4개가 필요한데 이것은 일종의 Memory Conflict로 Throughput 예측 시 고려해야한다. EEPROM은 8개 Fetch마다 7개의 추가 클럭이 발생하고, 그 이외의 메모리는 한번의 읽기/쓰기마다 각각 7개/4개의 추가 클럭이 발생한다.

3) Pipeline 구조에 따른 Throughput 영향[4]

C40의 Pipeline 구조는 그림 4와 같은 4-level 구조를 가지므로 Branch Conflict의 경우 B_{clk}은 3클럭이 된다. C6701은 그림 9와 같이 16-level Pipeline구조를 가지고 있고, 프로그램 메모리를 한번에 8번 연속 Fetch 해야 하는 구조적인 특징 때문에, Branch 관련 명령의 Fetch 이후, 다음 첫 명령어의 수행까지 프로그램 메모리를 24~32번까지 Fetch해야 하는 문제가 발생한다. 수행시간으로 계산하면 Branch 관련 명령어 수행($B_c \times B_{clk} \times t_{clk}$)에 최고 약 0.003mS가 소요된다. 내부 메모리를 사용하여 빠른 프로그램 Fetch와 동시에 최대 8개의 명령어를 실행할 수 있도록 설계된 C6701에게 외부 메모리 사용은 최악의 성능을 가져오는 제약조건이다. 이러한 Branch Conflict가 프로세서 변경 설계 후의 Throughput에 치명적인 결과를 가져올 것으로 예상된다.



[그림 9] C6701의 16-Level Pipeline Operation

4) CPU Throughput 예측

NI 社의 PXI-6542(Digital Waveform Analyzers)를 이용하여 제어주기 15.625mS 동안 C40의 Local-Bus와 Global-Bus의 어드레스 버스를 측정하였고, 측정한 결과를 이용하여 프로그램이 메모리 및 I/O 소자에 억세스하는 개수를 카운팅 하였다. 결과는 표 5와 같다.

[표 5] 메모리별 억세스 개수

메모리 종류	메모리 억세스 개수	
	읽 기	쓰 기
EEPROM	76759	-
SRAM	24011	9155
DPRAM	1306	502
NVRAM	6	15
I/O	26	6

Branch 관련 명령어 개수는 5231개, 전체 프로그램 수행시간은 13.767mS로 제어주기의 88.1%의 Throughput을 가지는 것으로 측정되었다.

표 2의 값과 측정결과를 (1), (2)식에 적용하면 다음과 같은 결과를 얻을 수 있다. C40은 이중버스 구조이므로 프로그램 메모리의 억세스 시간이 곧 t_{mem} 이다. 그러므로 (1)식에는 EEPROM의 읽기 억세스만 대입하여 계산하면 된다. 메모리 읽기/쓰기 억세스 시간에 충분한 Timing Margin을 고려하였음으로 t_{rmci}/t_{wmci}는 t_{racci}/t_{wacci}에 포함되었다.

t_{mem} =7.676mS, t_{inst} =6.091mS,
$R_c \times t_{clk} + N_{inst} \times t_{clk}$ = 4.522mS

$R_c \times t_{clk} + N_{inst} \times t_{clk}$ 항목은 CPU의 내부 동작 수행 부분이고, 메인 클럭에 관계된 요소이므로, 프로세서 변경 설계 조건하에 C6701에도 비례적으로 적용할 수 있다. C6701의 메인 클럭을 100MHz(10nS)로 선정 시 클럭 속도가 10배 향상되었으므로, 수행시간은 10배 감소한다. C6701의 Throughput 예측 시 0.452mS를 적용할 수 있다.

C6701은 비동기 메모리의 읽기/쓰기 억세스 시에 추가 클럭이 필요함으로, 추가적인 t_{mc}의 계산식 (10)이 필요하다.

$$t_{mc} = \left\{ \frac{R_{cp}}{8} \times 7 + \sum_{j=0}^{n-1} (R_{cj} \times 7 + W_{cj} \times 4) \right\} \times 10nS \quad (10)$$

여기서, t_{mc}는 Memory Conflict 시간, R_{cp}는 프로그램 메모리 읽기 개수, R_{cj}/W_{cj}는 프로그램 메모리를 제외한 메모리의 읽기/쓰기 개수이다.

표 4, 5의 값과 t_{mc}값을 식 (1)에 적용하면 t_{mem} = 12.629mS가 된다. 식 (2)에 B_c = 5231, Branch 관련 명령어 수행시간 0.003mS, $R_c \times t_{clk} + N_{inst} \times t_{clk}$ = 0.452mS를 대입하면 t_{inst} = 16.145mS가 된다.

따라서 C6701의 메인 클럭을 100MHz(10nS)로 선정 시 전체 수행시간은 28.477mS가 되고 제어 주기 15.625mS를 훨씬 초과하는 184%의 Throughput을 가지는 것으로 분석되었다. 추가적으로 C6701에서 지원하는 최고 클럭 167MHz(5.988nS)로 선정하여 수행시간을 계산한 결과 전체 수행시간은 27.083mS가 되고 173%의 Throughput을 가지는 것으로 분석되었다.

바. 분석 결과

프로세서 변경 설계에 따른 영향 분석 수행 결과 C6701은 대상 임베디드 컴퓨터의 아키텍처 및 인터럽트 소스의 조건을 모두 만족함으로서 하드웨어/소프트웨어적인 변경 타당성을 만족하였으나, CPU Throughput은 만족하지 못했다.

C40은 외부 메모리를 사용하는 아키텍처에 최적화된 이중버스를 가지는 프로세서이고, C6701은 내부 메모리를 사용하는 아키텍처에 적합하게 설계되어 단일버스를 가지고 있다. 또한, C6701은 연산속도를 높이기 위해 동시에 최대 8개의 명령어를 처리할 수 있도록 8개의 Function Unit이 구현되어 있고, 16-Level Pipeline 구조를 가지는 장점이 있다. 하지만 이러한 장점은 외부의 실시간 모니터링 시스템으로 동작 상태를 검증하기위해 외부 메모리를 사용해야하는 대상 임베디드 컴퓨터에는 단점으로 작용하였다. 표 6의 프로세서별 프로그램 수행시간 분석표와 같이 메모리 억세스와 관련된 항목들은 모두 수행시간이 증가했으며, CPU의 연산과 관련된 항목의 수행시간은 감소했음을 알 수 있다.

하지만 다른 제조사의 프로세서에 비해 같은 제조사의 프로세서를 선택함으로서 얻을 수 있는 장점이 많다.

[표 6] CPU별 프로그램 수행시간 분석

항 목	C40	C6701	비 고
메모리 억세스 시간	7.676mS	9.796mS	2.120mS 증가
Memory Conflict	-	2.833mS	2.833mS 증가
Branch Conflict	1.569mS	15.693mS	14.124mS 증가
Register Conflict +명령어 실행시간	4.522mS	0.452mS	4.070mS 감소

최근 개발된 프로세서는 모두 내부 메모리 사용에 최적화되어있고, 연산속도를 높이기 위해 높은 레벨의 Pipeline 구조를 채택하고 있음으로 다른 제조사 프로세서를 사용해도 원하는 Throughput을 획득하기는 어렵다.

요구하는 제어주기 내에 CPU Throughput을 만족하기 위해서는 메모리 억세스 관련 수행시간을 줄여야함으로 비동기 메모리에 비해 억세스 속도가 빠른 동기 메모리로 교체 설계가 필요할 것이다. 동기 메모리의 억세스 속도는 메인 클럭에 비례함으로 메인 클럭 속도를 높이면 충분히 Throughput을 만족시킬 수 있을 것으로 판단된다. 하지만 동기 메모리 교체로 인해 전체 시스템에 설계 변경의 영향을 미쳤다. 외부 실시간 모니터링 시스템의 메모리 인터페이스 부분도 동기 메모리에 적합하게 재설계되어야 할 것이다.

5. 결 론

본 논문에서는 무기체계의 수명주기가 끝나기 전에 필수 구성장치인 임베디드 컴퓨터의 프로세서가 단종되는 경우에 대해, 프로세서 변경 설계에 따른 임베디드 시스템에 미치는 영향을 분석하기 위한 방안을 제시하였다.

C40 프로세서를 채택하고 있는 대상 임베디드 컴퓨터를 C6701로 변경 설계하는 경우에 대해 제시된 영향 분석 방안을 적용하였고, 그 결과 프로세서 변경 설계만으로는 시스템 요구사항을 만족할 수 없으며, 비동기 메모리를 동기 메모리로 변경해야 하고 외부 모니터링 시스템도 동기 메모리 사용에 맞는 설계 변경이 필요한 것으로 분석되었다. 이것으로 제작사에서 제시하는 프로세서의 성능지표만 참고하여 변경 설계하는 것은 위험하며, 실제 CPU Throughput은 시스템에서 필요로 하는 주변회로와의 인터페이스에 따라 결정됨이 입증되었다.

본 논문에서 제시된 영향 분석 방안을 적용함으로서 설계 실패에 대한 위험을 줄이고, 비용 및 시간을 절감할 수 있을 것으로 판단된다.

본 논문의 경우와 달리 제어주기와 CPU Throughput 예측 결과가 근소한 차이를 보일 때는 프로세서와 메모리만으로 구성된 Prototype Board를 제작하여 실험을 통한 추가검증이 필요할 것으로 생각된다.

참 고 문 헌

[1] William R. Dunn, "Designing safety-critical computer systems", IEEE Computer Society Press, Vol. 36, No. 11, pp.40~46, November 2003.

[2] John Catsoulis, "Designing Embedded Hardware", O'Reilly, November 2002.

[3] TMS320C4x user's Guide, TI.

[4] TMS320C6000 CPU and Instruction Set. Reference Guide, TI.

[5] SMJ320C40 Data sheet, TI.

[6] SMJ320C6701 Data sheet, TI.

[7] TMS320C6201, TMS320C6416, TMS320C4x Self-test Program, TI.

[8] Syed A. Maroof, "TMS320C62x™ Self-check Program Application Brief : ver 1.0", TI, March 2000.

[9] David Abensur, Sebastien Tomas, "TMS320C6416 Power-On Self Test", TI, February 2004.

[10] Kyle Castille, "TMS320C6000 EMIF to External Asynchronous SRAM Interface", TI, August 2001.

[11] TMS320C6000 DSP External Memory Interface (EMIF) Reference Guide, TI.
[12] AM29F800B Data sheet, AMD.
[13] EDI816256CA Data sheet, White Electronic Designs Corporation.
[14] IDT7025S/L Data sheet, Integrated Device Technology, Inc.
[15] STK11C68 Data sheet, SIMTEK.