논문 2006-43SD-2-12

VLSI 구현을 위한 CAN 프로토콜 컨트롤러의 설계 및 검증

(Design and Verification of a CAN Protocol Controller for VLSI Implementation)

김 남 섭*, 조 원 경*

(Namsub Kim and Wonkyung Cho)

요 약

본 논문에서는 VLSI구현을 위한 CAN 프로토콜 컨트롤러의 최적화된 구조를 제안하였으며, 제안된 구조를 이용하여 VLSI로 구현하였다. 또한 많은 시간이 소요되는 검증의 문제점을 보완하기 위하여 3단계 검증기법을 제안하였으며 이를 통하여 빠른 속도의 검증이 가능하게 되었다. 제안된 구조는 기존의 CAN 프로토콜 컨트롤러보다 적은 사이즈의 게이트 수를 갖고 있을 뿐만 아니라 호스트 프로세서와의 연결이 용이하게 구성되어 있기 때문에 비용 및 효율성에서 장점을 갖고 있고, 제안된 3단계 검증기법은 반복되는 검증의 수를 줄임으로써 최적화된 검증을 수행하도록 구성되어 있기 때문에 빠른 속도의 검증이 가능하다. 설계된 CAN 프로토콜 컨트롤러는 0.35마이크론 CMOS공정을 이용하여 제작되었다.

Abstract

This paper presents design methodology, efficient verification and implementation of a CAN protocol controller. The design methodology uses a heuristic technique to make the design flexible and cost effective. Using the design methodology, we created architecture for a CAN controller which has flexible and low cost features. For faster time-to-market and reliable operation of the designed CAN protocol controller, we proposed a three-step verification process which uses three different kinds of verification techniques. The goal of this three-step verification is to reduce the number of test sequences in order to rapidly implement the design without loss of reliability for faster time-to-market. The designed CAN protocol controller was fabricated using a 0.35 micrometer CMOS technology.

Keywords : CAN, CANbus, Controller, Verification, Time-to-market, VLSI

I. 서 론

CAN(Controller Area Network)은 시리얼 통신 프로토콜의 일종으로 자동차 및 산업 응용기기의 데이터 통신에 자주 사용되는 프로토콜이다[1]. 특히 자동차의 경우에는 한 개의 자동차당 수십에서 수백 개의 컨트롤러가 내장이 되기 때문에 비용절감을 위하여 가능한 적은 사이즈를 갖는 컨트롤러의 설계가 필수적이다[2]~[8]. 그러나 CAN 프로토콜 컨트롤러(protocol controller)의 경우 복잡한 에러처리 알고리즘을 갖고 있기 때문에 설계가 용이하지 않고 또한 검증 시 많은 시간을 소요하게 된다[9]. 따라서 본 논문에서는 이러한 문제점을 해결하기 위하여 새로운 CAN 프로토콜1 컨트롤러의 구조를 제안하였으며 빠른 검증을 위하여 3단계 검증기법을 제안하였다.

본 논문의 구성은 다음과 같다. II장에서 기존의 설계기법의 문제점 및 이를 해결하기 위한 방안을 살펴보고, 제안된 CAN 프로토콜 컨트롤러의 구조에 대하여 기술한다. III장에서는 제안된 구조의 VLSI구현을 위한 검증기법에 대하여 최적화된 최종구현과정을 기술한다. VI

* 정회원, 경희대학교 전자정보학부
(School of Electronics and Information, Kyung Hee University)
접수일자 : 2005년9월20일 수정완료일 : 2005년2월1일

장에서는 구현된 칩에 대한 결과 및 평가를 기술하고 마지막으로 V장에서 결론을 맺는다.

II. CAN 프로토콜 컨트롤러의 설계

1. CAN 프로토콜

CAN 프로토콜은 1983년 Robert Bosch GmbH에 의하여 개발되었으며 1991년 ISO(International Organization for Standardization)에 의하여 국제표준으로 채택되었다[10]. CAN은 멀티마스터(multi-master)의 구조를 갖고 있으며 복잡한 point-to-point의 연결방식을 하나의 직렬라인(serial line)으로 대체할 수 있기 때문에 많은 주변기기들과의 통신에 자주 사용된다. 또한 EMI 노이즈가 심하게 발생할 수 있는 상황에서도 에러 없는 데이터 전송을 할 수 있는 장점이 있다.

CAN 프로토콜은 기본적으로 NRZ(Non Return to Zero) 비트 코딩기법을 사용하고 있으며 CSMA/CD (Carrier Sense Multiple Access/Collision Detection)와 NDA(Non-Destructive Arbitration), AMP(Arbitration by Message Priority)기능을 포함하고 있다. CAN 프로토콜의 가장 큰 장점은 복잡한 에러처리 기법을 통해 에러에 강건하게 대체할 수 있는 것이며 CAN 프로토콜에서 사용하는 에러처리 기법은 CRC(Cyclic Redundancy Check), 비트 스터핑(bit stuffing), ACK 검사(acknowledgement check), 비트 모니터링(bit-monitoring)과 데이터 프레임(data frame)단위의 검사를 수행한다. CAN 버스 시스템은 40kbit/s의 속도로 최대 1km까지 데이터를 전송할 수 있으며 고속 전송이 필요할 경우 1Mbit/s의 속도로 40m까지 자유로운 데이터 전송이 가능하다.

2. 기존의 CAN 프로토콜 컨트롤러 설계의 문제점

그림 1은 기존의 CAN 프로토콜 컨트롤러의 설계구조이다[11].

그림 1에 나타난 구조는 전체적인 데이터의 흐름을 잘 표현하고 있으나 실제 VLSI로의 구현을 위해서는 다음과 같은 문제점을 갖고 있다. 먼저 첫 번째로 위와 같은 구조의 설계 시 각 블록의 의존도가 높기 때문에 변화하는 호스트 프로세서에 대하여 유동적으로 대처하기 어렵다. 두 번째 문제점은 BSP(Bit Stream Processor)블록의 복잡도(complexity)가 타 블록에 비하여 높기 때문에 설계 시 블록간의 균형이 맞지 않아 설계 오류를 야기할 수 있으며 이는 곧 검증의 어려움을

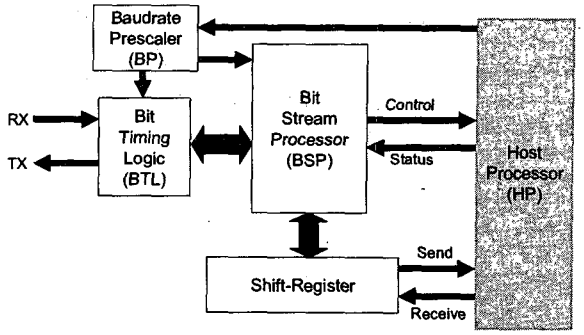

그림 1. 기존의 CAN 프로토콜 컨트롤러의 구조
Fig. 1. Structure of conventional CAN protocol controller.

가져올 수 있다. 세 번째 문제점은 위에서 지적한 블록간의 균형이 맞지 않을 경우 HDL(Hardware Description Language)을 이용한 최종 설계 시 사용되는 합성툴의 종류 및 셀 라이브러리에 따라 설계되는 칩의 크기에 많은 변동이 있다는 것이다.

이와 같은 합성결과의 차이는 셀 라이브러리뿐만 아니라 사용된 제약조건(constraints), 합성 툴의 종류에도 영향을 받는다[9].

3. 디자인 기법

2절에서 설명한 문제점들을 해결하기 위하여 본 논문에서는 그림 2와 같은 설계 최적화 과정을 통하여 최적화된 CAN 프로토콜 컨트롤러의 구조를 생성해 내었다.

위 디자인 기법은 "divide and conquer"의 기법을 채택하여 이를 설계기법에 적용하였다. Divide and conquer는 새로운 알고리즘 개발 및 수학적인 문제풀이를 위하여 자주 사용되는 기법으로 이를 설계 시 적용하였을 경우 IV장의 실험결과에서 나타낸 것과 같이 최적화된 설계구조의 개발에 적합하다는 것을 알 수 있다. 또한 각 블록의 합성평가(synthesis evaluation) 시 게이

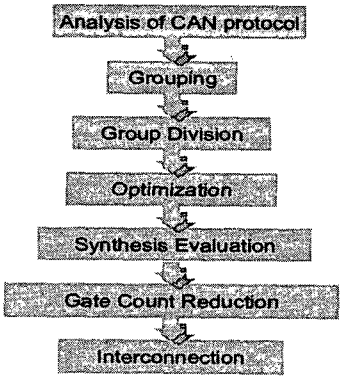

그림 2. 최적화된 구조설계를 위한 디자인 기법
Fig. 2. Design methodology for optimized architecture.

트 수(gate count)가 1000을 넘을 경우 합성 라이브러리의 영향을 많이 받기 때문에 그룹 분할(group division) 과정의 기준 게이트 수를 1000으로 설정하여 설계 하였다.

4. 제안하는 구조 및 설계

그림 3은 위의 기법을 통해 생성해 낸 최적화된 CAN 프로토콜 컨트롤러의 구조를 나타낸다.

제안된 구조는 총 19개의 블록으로 구성되어 있으며 CAN protocol의 분석을 통하여 각각의 모듈들이 독립적으로 작동되도록 되어 있다. 이와 같은 독립성은 설계시 발생하는 설계오류에 대한 디버깅을 용이하게 하여주며 독립적인 블록의 최적화를 통하여 적은 사이즈의 칩 설계가 가능하게 된다.

가. Synchronizer

Synchronizer는 CAN Transceiver를 통하여 들어온 CAN신호를 내부 클럭을 이용하여 동기를 맞추는 기능을 한다. 동기화는 ISO 11898-1[10]에 명시된 방법에 따라 입력된 신호를 Time Quanta를 사용하여 최종 데이터를 직렬로 추출해낸다.

나. Field Manager

Field Manager는 입력된 데이터의 형식에 따라 Data Frame, Remote Frame, Error Frame, Overload Frame, Interframe Space인지를 판단하며 판단된 결과에 따라 데이터를 처리한다. 또한 에러가 발생된 데이터의 처리를 위하여 입력된 데이터를 병렬로 Error Checker블록들에 전송한다.

다. Error Checkers

CAN protocol은 데이터의 에러검출 및 처리를 위하여 발생되는 에러의 형태를 크게 CRC, Form, Stuff, ACK, Bit의 5가지 형태로 구분하고 이에 따른 에러의 발생 유무를 CAN bus에 전송하도록 되어있다. 본 논문에서는 이와 같은 5가지 에러검출 블록을 독립적으로 따로 구성하였으며 각각의 블록으로부터 검출된 결과는 Frame Generator와 Serializer를 통하여 CAN bus에 전송된다.

라. Frame Generators

Frame Generator는 Host processor로부터 전송된 데이터를 CAN bus에 전송하는 Data&Remote Frame Generator와 에러발생의 유무를 CAN bus에 알리는 Error&Overload Frame Generator로 구성되어 있다.

마. Interface Logic

Interface Logic은 Host Processor와의 인터페이스를 담당하는 역할을 한다. 참고문헌 [2][3][4][5][16]에 나타난 바와 같이 외부 인터페이스 블록을 데이터 처리블록과 합쳐놓았을 경우 Host Processor의 종류가 바뀔 때

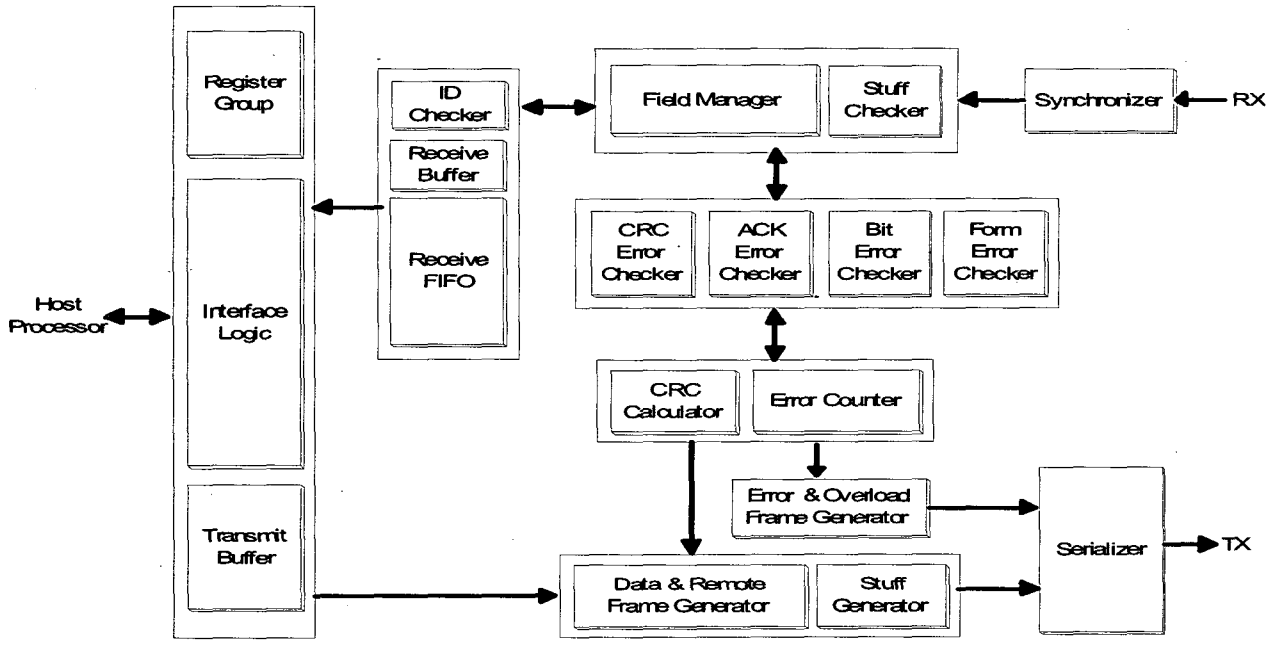

그림 3. 제안된 CAN protocol controller의 구조
Fig. 3. Structure of proposed CAN protocol controller.

마다 전체 블록을 수정하여야 하는 단점이 있다. 이는 설계된 회로가 IP(Intellectual Property)로 사용될 경우 IP의 용이성(Flexibility)이 떨어진다는 것을 의미하며 본 논문에서는 간단한 Interface Logic의 수정을 통하여 다양한 Host Processor와의 연결이 가능하도록 하였다. 제안된 설계는 8051 마이크로 컨트롤러(micro-controller) 계열의 프로세서와 인터페이스가 가능하며 전체 블록의 수정 없이 Interface Logic의 수정만으로 모든 외부 프로세서와의 인터페이스가 가능하다.

바. Serializer 및 기타블록

Serializer는 병렬로 입력된 데이터를 CAN bus에 해당하는 직렬 비트로 변환하여 출력하는 기능을 한다. 그 외 Transmit, Receive Buffer 및 Register Group은 데이터의 임시저장 및 비트레이트(bit rate)를 포함한 회로 초기화 값의 저장용도로 사용되어 진다.

Ⅲ. 검 증

1. 검증 및 문제점

VLSI설계를 위한 가장 중요한 요소는 바로 검증이다. 정확한 검증을 통하지 않은 설계는 신뢰 할 수 없기 때문에 실제 설계 시 검증은 전체 설계의 80%에 해당하는 시간을 요구하게 된다. 따라서 보다 빠른 시장대응(Time-to-Market)을 위해서는 빠르고 정확한 검증기법을 필요로 하며 본 논문에서는 3단계 검증기법(Three Step Verification)을 제안하고 이를 설계된 CAN 프로토콜 컨트롤러에 적용하였다.

3단계 검증기법의 기본 아이디어는 검증을 위하여 수행되는 시뮬레이션(simulation)에서 중첩되는 테스트 벡터를 제거하여 시뮬레이션의 속도를 높이는 것이다. 특히 VLSI설계 시 게이트 수준(gate level)과 포스트 레이아웃 시뮬레이션(post-layout simulation)이 기능적 시뮬레이션(functional simulation)보다 많은 시간을 요구하게 된다는 점에 착안하여 필요하지 않는 테스트 벡터(test vector)를 제거하여 검증을 수행하게 되면 보다 높은 속도의 검증을 할 수 있으며 이는 곧 전체 설계시간을 단축시킬 수 있음을 의미한다.

이를 수학적으로 표현하면 다음과 같다.

전체 시뮬레이션시간은 다음과 같이 정의 된다.

$$t_{total} = \alpha\,(t_{func} + t_{gate} + t_{post}),\ \alpha \geq 1 \qquad (1)$$

수식 (1)에서 α는 테스트 벡터의 가중치(weighting factor)이며 t_{func}, t_{gate}, t_{post}는 각각 기능적, 게이트 수준, 포스트 레이아웃 시뮬레이션 시간을 나타낸다. 즉 테스트 벡터의 수가 커질수록 전체 검증시간은 비례적으로 증가하게 된다. 만약 게이트 수준 과 포스트 레이아웃 시뮬레이션에서 중첩되는 테스트 벡터를 제거하게 되면 식 (2)와같이 다시 쓸 수 있다.

$$t_{total} = \alpha t_{func} + t_{gate} + t_{post},\ \alpha \geq 1 \qquad (2)$$

Ⅳ장의 실험결과에서 나타낸 바와 같이 게이트 수준l과 포스트 레이아웃 시뮬레이션의 수행시간이 기능적 시뮬레이션보다 크기 때문에 식 (2)와같이 중첩된 테스트 벡터를 제거함으로써 전체 검증시간을 크게 단축시킬 수 있다. 그러나 위의 시뮬레이션 시간은 HDL 코딩 시 발생하는 코드오류로 인한 디버깅(debugging) 시간이 포함되지 않았을 뿐만 아니라 이와 같은 테스트 벡터의 감소는 검증의 정확도를 떨어뜨릴 수 있으므로 이를 위하여 본 논문에서는 다음과 같은 3단계 검증기법을 제안한다.

2. 3단계 검증기법(Three Step Verification)

제안하는 3단계 검증기법은 다음과 같은 단계로 구성되어 있다.

- 1단계 : Rough Verification (RV)
- 2단계 : Conformance Test (CT)
- 3단계 : FPGA Emulation

첫 번째 단계에는 그림 4와 같은 검증모델을 사용하여 감소된 테스트 벡터를 이용하여 기능적 시뮬레이션을 수행한다.

이와 같은 감소된 테스트 벡터를 사용하는 이유는 초기 설계 시 발생되는 하드웨어 설계의 디버깅을 용이하게 하기 위함이며 프로토콜의 기본 기능만을 시뮬레이션 한다. 이와 같은 기본기능에 대한 시뮬레이션은 후에 수행되는 게이트 수준 및 포스트 레이아웃 시뮬레이션에서 그대로 사용되며 이는 상대적으로 많은 시간이 소요되는 시뮬레이션에서 적은 테스트벡터 및 테스트 알고리즘을 이용하여 많은 시간단축을 이룰 수 있다는 장점이 있다.

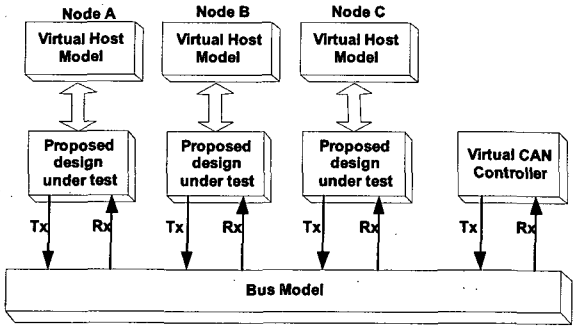

그림 4. CAN protocol controller의 검증모델
Fig. 4. Verification model of CAN protocol controller.

두 번째 단계에서는 ISO-16845에서 제안하는 Conformance Test기법[13]을 사용하여 기능적 시뮬레이션을 수행한다. 본 논문에서는 테스트의 용이성을 위하여 그림 5와 같이 PLI(Programming Language Interface)를 이용하여 테스트를 수행하였다.

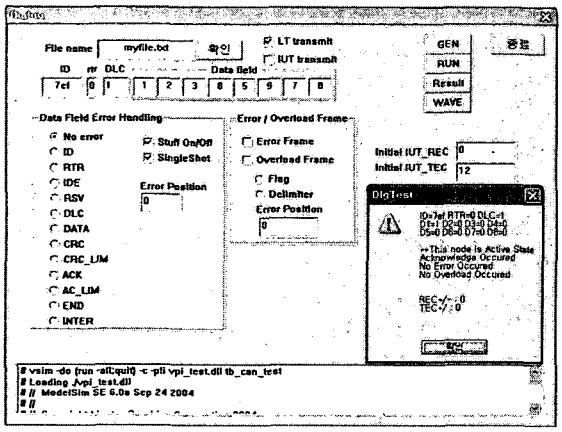

그림 5. PLI를 이용한 Conformance Test
Fig. 5. Conformance Test using PLI.

두 번째 단계의 검증은 상대적으로 적은 시간이 소요되는 기능적 시뮬레이션에서 가장 많은 테스트 벡터 및 알고리즘을 사용하여 설계의 정확도를 높이기 위함이다. 그러나 이와 같은 많은 수의 테스트 벡터 및 알고리즘을 이용할 경우 시뮬레이션 오류 시 디버깅을 위해 많은 시간이 소비될 수 있으므로 제안된 구조와 같은 설계수정이 용이한 구조로 설계가 되어야만 한다.

마지막 단계에서는 그림 6과 같은 FPGA 프로토타입(prototype)을 이용하여 게이트 수준l 및 포스트 레이아웃 시뮬레이션에 대한 사전 검증을 실시한다. FPGA 프로토타입을 이용 할 경우 최종 VLSI구현 시 필요한 게이트 수준과 포스트 레이아웃 시뮬레이션의 테스트 벡터를 줄일 수 있으며 이를 이용하여 보다 빠른 테스트 뿐만 아니라 정확한 검증이 이루어질 수 있다.

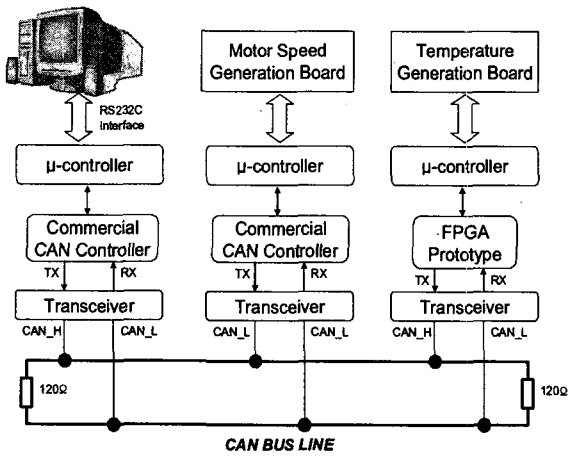

그림 6. FPGA 프로토타이핑 시스템
Fig. 6. FPGA prototyping system.

3. VLSI구현과정 및 검증

제안된 설계구조와 검증기법을 사용한 최종적인 CAN 프로토콜 컨트롤러의 VLSI 구현과정은 그림 7과 같다.

그림 7에 나타낸 VLSI구현 과정은 일반적인 설계과정과 유사하나 전체 설계시간 단축을 위하여 제안된 구조와 검증기법을 적용하였다.

구현을 위한 전체 시뮬레이션 시간은 식 (3)과 같다.

$$t_{total} = t_{func1} + \alpha t_{func2} + t_{gate} + t_{post}, \alpha \geq 1 \quad (3)$$

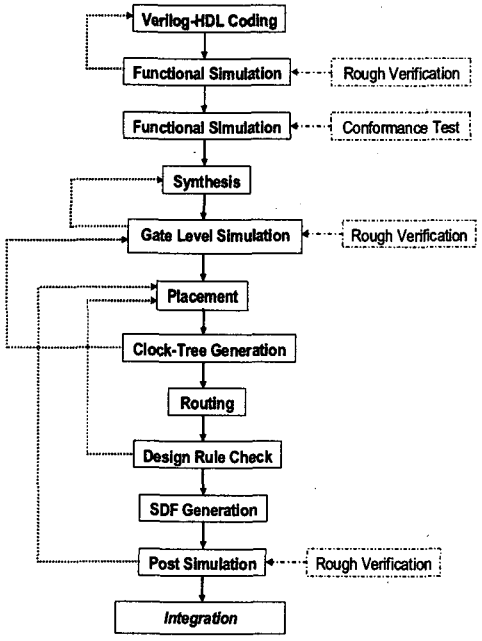

그림 7. VLSI 구현 및 검증과정
Fig. 7. Implementation flow and verification.

여기서 t_{func1}과 t_{func2}는 각각 Rough Verification과 Conformance Test를 나타낸다. 위의 수식은 디버깅을 위한 시간을 고려하지 않은 것이므로 M개의 시뮬레이션 에러가 발생할 때 소요되는 디버깅 시간을 D_i라고 하면 전체 디버깅 시간은 식 (4)와같이 쓸 수 있다.

$$t_{debug} = \sum_{i=1}^{M} D_i \tag{4}$$

여기서 Conformance Test의 시뮬레이션 에러의 수를 N이라고 하고 게이트 수준과 포스트 레이아웃 시뮬레이션에서 발생되는 에러가 없다고 가정하면 최종 소요시간은 식 (3)과 식 (4)에 의해 식 (5)와같이 나타낼 수 있다.

$$t_{total} = \sum_{i=1}^{M} D_i + Mt_{func1} + \sum_{i=1}^{N} D_i + \alpha N t_{func2} + t_{gate} + t_{post} \tag{5}$$

일반적인 설계과정에서의 디버깅을 포함한 전체 시뮬레이션 시간을 나타내면 식 (6)과 같다.

$$t_{total} = \sum_{i=1}^{L} D_i + \alpha (Lt_{func} + t_{gate} + t_{post}) \tag{6}$$

설계 구조의 복잡도가 커짐에 따라 α의 값은 비례적으로 높고 Rough Verification을 통하여 N이 L보다 매우 작기 때문에 식 (5)는 식 (6)에 비해 낮은 시간을 소비하며 이는 곧 전체 검증시간의 단축을 의미한다.

IV. 실험 결과 및 평가

1. 합성 결과

먼저 게이트 수에 따른 합성의 영향을 조사하기 위하여 그림 8과 같이 두 개의 각기 다른 라이브러리를 사용하여 합성을 한 후 그 결과를 비교하였다.

그림 8에서 사용된 참조 라이브러리(Reference Library)는 자체 제작된 라이브러리를 이용하였으며 비교된 라이브러리(Compared Library)는 Hynix 0.35공정용 합성라이브러리를 사용하였다. 그림 9에서 나타난 바와 같이 합성되는 블록의 게이트 수가 1000을 넘어가면 합성결과에 따른 차이가 크게 됨을 알 수 있다. 따라서

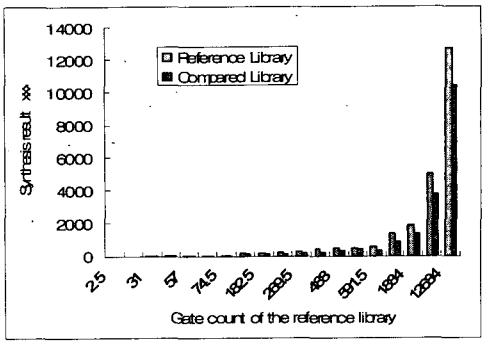

그림 8. 셀 라이브러리에 따른 합성결과의 차이
Fig. 8. Gate count variation from different cell library.

표 1. 각 블록의 게이트 수에 대한 결과
Table 1. Gate counts of each functional block.

Functional Blocks	Gate Count
Interface Logic	100
ID Checker	62
Field Manager	627
Synchronizer	594
Serializer	176
Data & Remote Frame Gen.	364
Error & Overload Frame Gen.	207
CRC Calculator	325
Error Counter	552
Stuff Checker	132
Error Checkers (ACK, BIT, FORM)	50
TOTAL	3189

본 논문에서는 각각의 블록의 합성결과가 1000을 넘지 않도록 하였으며 최종 합성된 결과를 표1에 나타내었다. 합성은 Synopsys사의 Design Compiler를 사용하였으며 Hynix 0.35 마이크론 CMOS공정용 합성 라이브러리를 이용하였다.

표 1에서 나타낸 바와 같이 제안된 구조와 최적화 기법을 통하여 설계를 하였을 경우 각각의 블록이 균일한 분포의 합성 결과를 얻게 되었다. 이와 같은 블록 간 균일성은 II장에서 설명하였던 합성 툴 및 셀 라이브러리의 영향을 최소화 할 수 있다. 또한 제안된 구조는 각각의 블록이 독립적인 기능을 수행하기 때문에 설계과정 중 발생되는 디버깅시간을 단축시킬 수 있는 장점이 있다.

2. 시뮬레이션 결과

먼저 기능적 시뮬레이션과 게이트 수준 시뮬레이션의 차이점을 비교하기 위하여 같은 테스트 벡터와 테스트 환경을 이용하여 실험하였다. 실험은 펜티엄-4 3.2GHz

프로세서를 탑재한 PC를 사용하여 리눅스 환경에서 수행되었으며 시뮬레이션 툴로는 Cadence사의 Verilog-XL을 사용하였고 게이트 변환을 위한 합성툴은 Synopsys사의 Design Compiler를 이용하였다. 표 2의 실험결과가 나타내듯이 같은 조건일 경우 게이트 수준 시뮬레이션 시 많은 시간이 소요됨을 알 수 있다.

이와 같은 차이는 시뮬레이션 시 발생되는 이벤트(event)가 게이트 수준일 경우 기능적 시뮬레이션보다 많은 이벤트를 포함하기 때문이며 포스트 레이아웃 시뮬레이션의 경우 게이트 수준과 같은 방식으로 검증이 진행되기 때문에 비교대상에서 제외하였다.

제안된 구조와 검증기법에 대한 평가를 위하여 표 3에 제안된 구조와 검증기법을 사용한 각 시뮬레이션에 대한 소요된 시간을 나타내었다.

표 3에 나타난 바와 같이 제안된 기법을 사용하여 검증을 할 경우 전체 시뮬레이션 시간이 기존 방식의 게이트 수준 시뮬레이션보다 적게 소요된다는 것을 알 수 있고 또한 초기 설계 시 소요되는 디버깅 시간을 단축

표 2. 시뮬레이션 시간 비교 (단위:초)
Table 2. Simulation time comparisons (units: seconds).

Simulation Type	Compile Time	Link time	Sim. time	Total
Functional simulation (RV)	0.1	0.1	299.5	299.7
Functional simulation (Conformance Test)	0.58	0.58	11321.6	11322.8
Gate level simulation (RV)	0.2	0.9	856.3	857.4
Gate level simulation (Conformance Test)	17.4	23.2	96807.8	96848.4

표 3. VLSI구현 시 소요된 시뮬레이션 시간 (단위:초)
Table 3. Simulation time for VLSI implementation (units: seconds).

Simulation Type	Compile time	Link time	Sim. time	Total
Functional simulation	0.1	0.1	299.5	299.7
Functional simulation (Conformance Test)	0.58	0.58	11321.6	11322.8
Gate level simulation	0.2	0.9	856.3	857.4
Post layout simulation (clock tree included)	0.2	1.0	1007.8	1009.0
Post layout simulation (SDF, without cell check)	0.2	1.0	758.8	760

시킬 수 있음을 알 수 있다.

또한 제안된 검증기법의 신뢰성을 조사하기 위하여 Rough Verification을 사용했던 게이트 수준과 포스트 레이아웃단계의 시뮬레이션을 Conformance Test에서 사용하였던 테스트 벡터와 알고리즘을 사용하여 테스트한 결과 동일한 결과를 얻어낼 수 있었다.

3. 비교 및 평가

최종 구현된 칩의 평가를 위하여 표 4와 같이 기존에 발표된 CAN 프로토콜 컨트롤러와 비교를 하였다. 기존 방식의 DBCAN의 경우 적은 수의 복잡도가 높은 블록들로 인하여 전체 칩에 소요되는 게이트의 수가 타 설계보다 큰 단점을 갖고 있으며 참고문헌 [3]에서 제안된 설계의 경우 적은 게이트를 갖는 반면에 외부 메모리 사용으로 인한 설계의 유연성이 떨어짐을 알 수 있다. CLAN의 경우 최종구현을 FPGA로만 국한하여 설계하였기 때문에 정확한 게이트 수에 의한 비교가 어렵지만 ISO의 Conformance Test를 수행하지 않았기 때문에 설계의 신뢰성을 보장할 수 없고 내부 버퍼사용으로 인한 세부 블록들의 상호의존성이 크기 때문에 설계수정이 어렵다는 단점이 있다. 그러나 제안된 구조의 설계의 경우 기존에 발표된 설계보다 적은 게이트 수를 갖고 있을 뿐만 아니라 독립적인 블록으로 구성되어 있기 때문에 설계수정이 용이하다는 장점이 있다.

최종적으로 구현이 완료된 CAN 프로토콜 컨트롤러를 그림 9에 나타내었다. 또한 완성된 칩의 최종 테스트를 위하여 그림 10과 같은 보드를 제작하여 최종 검사를 수행하였고 설계된 칩의 정상적인 동작을 확인하였다.

표 4. 기존의 CAN 프로토콜 컨트롤러와의 비교
Table 4. Comparisons between proposed and previous work.

	The Proposed	DBCAN [2]	The paper [3]	CLAN [16]
Gate count	3,189	11,824	6,840	30,000 equivalent logic gate
Type	BasicCAN	BasicCAN	BasicCAN	BasicCAN
Memory structure	FIFO registers	Dual-port memory	External memory	Internal Buffer
Number of blocks	19	6	12	10
Verification method	Three Step Verification	State chart and emulation	ARDID in RTL simulation	FPGA emulation only

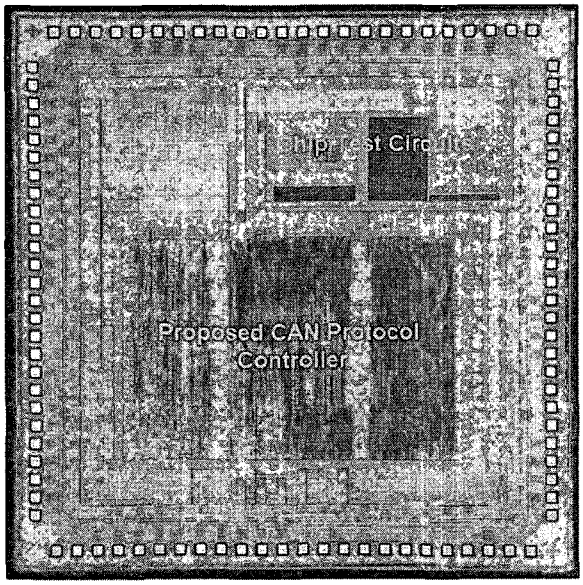

그림 9. 제안된 CAN 프로토콜 컨트롤러의 칩구현
Fig. 9. Photograph of the proposed CAN protocol controller.

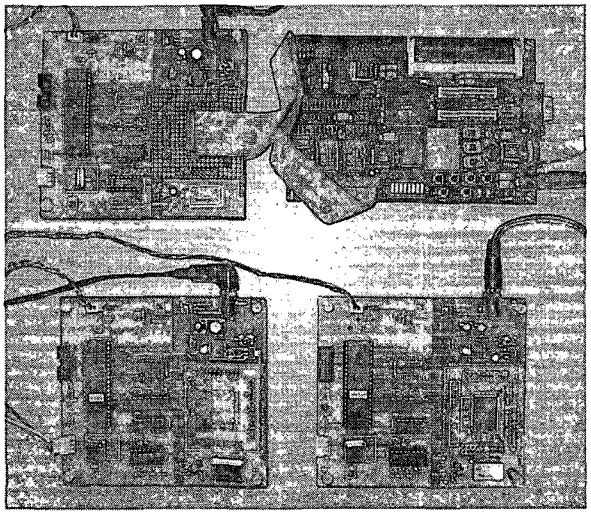

그림 10. 구현된 칩에 대한 최종 테스트 보드
Fig. 10. Photograph of final test boards of the implemented chip.

구현된 CAN 프로토콜 컨트롤러는 Hynix 0.35마이크론 CMOS공정을 이용하여 제작되었으며 38.3712mW의 파워를 소비하고 50MHz의 클록 주파수로 동작한다.

V. 결 론

사용자의 요구가 많아짐에 따라 다양한 주변기기간의 데이터 전송에 효율적으로 이용될 수 있는 컨트롤 프로토콜로써 CAN의 역할이 증대되고 있다. 특히 자동차나 산업현장 같은 잡음이 심한 상황에서 에러 없는 데이터 전송에 적합하기 때문에 그 사용빈도가 늘어나고 있으나 복잡한 에러처리 알고리즘으로 인해 구현 시 비효율적인 면이 있다. 본 논문에서는 이러한 비효율적인 면의 원인을 규명하여 효율적이고 빠른 구현을 목표로 하였으며 정확하고 빠른 검증을 통한 CAN 프로토콜 컨트롤러를 구현하였다. 또한 구현된 CAN 프로토콜 컨트롤러는 기존의 같은 방식의 컨트롤러보다 적은 게이트 수를 갖고 있을 뿐만 아니라 독립적인 블록으로 유연한 구조를 갖고 있어서 자체적인 VLSI구현 및 IP로의 활용에 있어서 유리함이 있다.

설계된 CAN 프로토콜 컨트롤러는 3단계 검증방법을 통하여 효율적으로 검증되었으며 FPGA 및 실리콘 칩(silicon chip)으로 구현되었다.

차후 연구될 과제로는 특정한 설계에 제한이 되어있는 제안된 검증기법을 일반화하여 모든 설계에 제안된 기법이 사용될 수 있도록 하는 것이며 이를 위해서는 보다 많은 설계 사례에 대한 양적인 비교 검토가 필요할 것이라 판단된다.

참 고 문 헌

[1] Wolfhard Lawrenz, *CAN System Engineering From Theory to Practical Applications*, Springer, 1997.
[2] Kirschbaum A., Renner F. M., Wilmes A., and Glesner M., "Rapid-Prototyping of a CAN-Bus Controller: A Case Study," in Proc. Seventh IEEE International Workshop on Rapid System Prototyping, pp. 19-21, 1996.
[3] J. de Lucas, M. Quintana, T. Riesgo, Y. Torroja, and J. Uceda, "Design of a CAN interface for custom circuits," in Proc. The 25th Annual Conference of the IEEE on Industrial Electronics Society (IECON '99), vol. 2, pp. 662-667, 1999.
[4] Donchev B. and Hristov M., "Implementation of CAN controller with FPGA structures," in Proc. CAD Systems in Microelectronics(CADSM 2003), pp. 577-580, 2003.
[5] Winter A., Bittruf D., Tanurhan Y., and Muller-Glaser K. D., "Rapid prototyping of a communication controller for the CAN bus," in Proc. Seventh IEEE International Workshop on Rapid System Prototyping, pp. 152-157, 1996.
[6] Guerrero C., Rodriguez-Navas G. and Proenza J., "Design and implementation of a redundancy manager for triple redundant CAN controllers," in Proc. The 28th Annual Conference of the IEEE on Industrial Electronics Society (IECON

'02), pp. 2294-2299, 2002.
[7] Ashenden P.J., "Modeling digital systems using VHDL," *IEEE Potentials*, vol. 17, pp. 27-30, 1998.
[8] Hoi Jun Yoo, *IP Authoring and SoC Design Methodology*, Technical Document at SIPAC, 2003.
[9] Byoung-Woon Kim and Chong-Min Kyung, "Exploiting intellectual properties with imprecise design costs for system-on-chip synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, pp. 240-252, 2002.
[10] International Standard ISO 11898-1, *Road vehicles Controller area network (CAN), Part1: Data link layer and physical signaling*, 2003.
[11] Florian Hartwich and Armin Bassemir, "The Configuration of the CAN Bit Timing," in Proc. 6th International CAN Conference (iCC99), 1999.
[12] Van Osch M. and Smolka S. A., "Finite-State Analysis of the CAN Bus Protocol", in Proc. Sixth IEEE International Symposium on High Assurance Systems Engineering, pp. 42-52, 2001.
[13] International Standard ISO 16845, *Road vehicles Controller area network (CAN) Conformance test plan*, 2004.
[14] Philips Semiconductors, *SJA1000 Stand-alone CAN controller DATA SHEET*, 2000.
[15] Namsub Kim, Kyoohyung Cho, Dawi Kim, Jinsang Kim, and Wonkyung Cho, "Design and verification of a CAN controller for custom ASIC," in Proc. 10th International CAN Conference on Semiconductor solutions (iCC2005), pp. 13-18, 2005.
[16] Arnaldo S. R. Oliveira, Nelson L. Arqueiro, Pedro N. Fonseca, "CLAN - A technology-independent synthesizable CAN controller," in Proc. 10th International CAN Conference on Semiconductor solutions (iCC2005), pp. 108-115, 2005.

저 자 소 개

김 남 섭(정회원)
1990년 경희대학교 전자공학과 학사
1992년 경희대학교 전자공학과 석사
2000년 ~ 현재 경희대학교 전자공학과 박사 과정.
<주관심분야 : 반도체 설계, 스테레오 비젼>

조 원 경(정회원)
1971년 경희대학교 전자공학과 학사
1973년 한양대학교 전자공학과 석사
1986년 한양대학교 전자공학과 박사
1980년 ~ 현재 경희대학교 전자공학과 정교수
<주관심분야 : VLSI설계, 마이크로프로세서>