(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0063496
(43) 공개일자 2016년06월07일

(51) 국제특허분류(Int. Cl.)
G11C 29/10 (2015.01)
(21) 출원번호 10-2014-0166549
(22) 출원일자 2014년11월26일
심사청구일자 2014년11월26일

(71) 출원인
인하대학교 산학협력단
인천광역시 남구 인하로 100, 인하대학교 (용현동)
에스케이하이닉스 주식회사
경기도 이천시 부발읍 경충대로 2091
(72) 발명자
강진구
서울특별시 서초구 잠원로12길 5, 335동 1003호(잠원동, 신반포아파트)
장해종
인천 서구 봉오대로253번길 58, 101동 207호 (가정동, 동우아파트)
(74) 대리인
양성보

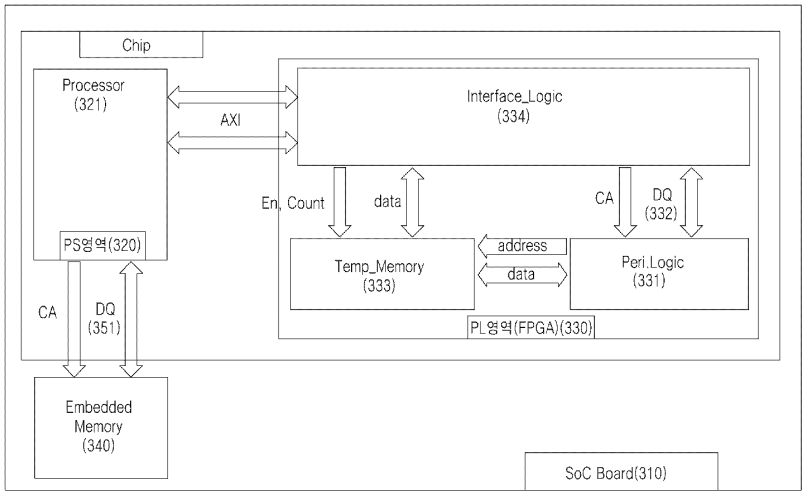
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 **메모리 구동 주변 회로 검증시간을 단축하기 위한 임베디드 DDR 메모리를 이용한 검증 방법 및 시스템**

(57) 요 약

메모리 구동 주변 회로 검증시간을 단축하기 위한 임베디드 DDR 메모리를 이용한 검증 방법 및 시스템이 제시된다. 본 발명에서 제안하는 메모리 구동 주변 회로 검증시간을 단축하기 위한 임베디드 DDR 메모리를 이용한 검증 방법은 입력 벡터를 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환하여 프로세스 영역에 전달하는 단계, 상기 전달된 입력 벡터를 프로그램 로직 영역에서 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환하는 단계, 쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 상기 입력 벡터를 비교하고, 읽기 동작 시 상기 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 상기 메모리 구동 주변 회로를 검증하는 단계를 포함할 수 있다.

대 표 도



명 세 서

청구범위

청구항 1

임베디드 메모리를 이용한 검증 방법에 있어서,

입력 벡터를 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환하여 프로세스 영역에 전달하는 단계;

상기 전달된 입력 벡터를 프로그램 로직 영역에서 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환하는 단계; 및

쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 상기 입력 벡터를 비교하고, 읽기 동작 시 상기 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 상기 메모리 구동 주변 회로를 검증하는 단계

를 포함하는 임베디드 메모리를 이용한 검증 방법.

청구항 2

제1항에 있어서,

상기 쓰기 동작 시 상기 프로세스 영역의 프로세서를 통해 전달된 입력 벡터의 정보들은 인터페이스 로직을 거쳐 메모리 구동 주변회로의 인터페이스의 입력인 CA 및 DQ 신호로 변경되어 전달되고, 출력으로 어드레스 정보와 데이터를 출력하는 것을 특징으로 하는 임베디드 메모리를 이용한 검증 방법.

청구항 3

제2항에 있어서,

상기 인터페이스 로직의 제어를 통해 상기 프로세서와 연동하여 상기 임베디드 메모리의 해당 어드레스에 상기 쓰기 동작을 수행하는 것을 특징으로 하는 임베디드 메모리를 이용한 검증 방법.

청구항 4

제1항에 있어서,

상기 읽기 동작 시 상기 입력 벡터를 입력 받은 인터페이스 로직이 상기 임베디드 메모리의 데이터를 상기 임시 메모리로 읽어 들이는 선행 동작을 수행한 후, 상기 메모리 구동 주변회로의 입력인 CA가 생성되고, 출력으로 어드레스를 상기 임시 메모리로 전달하면 읽어 들인 상기 임베디드 메모리의 데이터를 가져오는 것을 특징으로 하는 임베디드 메모리를 이용한 검증 방법.

청구항 5

제1항에 있어서,

상기 메모리 구동 주변회로와 데이터를 주고 받을 공간으로 상기 임베디드 메모리를 사용하는 것을 특징으로 하는 임베디드 메모리를 이용한 검증 방법.

청구항 6

제1항에 있어서,

상기 주변장치 로직과 상기 임베디드 메모리 사이에는 임지 저장 공간이 존재하는 것을 특징으로 하는 임베디드 메모리를 이용한 검증 방법.

청구항 7

임베디드 메모리를 이용한 검증 시스템에 있어서,

미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환된 입력 벡터를 전달 받는 프로세스 영역;

상기 전달된 입력 벡터를 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환하는 프로그램 로직 영역를 포함하고,

상기 프로세스 영역은,

쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 상기 입력 벡터를 비교하고, 읽기 동작 시 상기 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 상기 메모리 구동 주변회로를 검증하는 것을 특징으로 하는 임베디드 메모리를 이용한 검증 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 메모리 구동 주변 회로 검증시간을 단축하기 위한 임베디드 DDR 메모리를 이용한 검증 방법 및 시스템에 관한 것이다.

배 경 기 술

[0002] 기존의 메모리 구동 주변회로(Memory Peripheral Logic)를 검증하기 위해서는 도 1의 흐름처럼 그 회로(Circuit)에 대한 넷리스트(Netlist)정보를 Verilog HDL(합성 불가능한 문법)로 변환한 후에 시뮬레이션(Simulation)을 진행해야 한다.

[0003] 도 1은 종래 기술에 따른 메모리 구동 주변 회로 검증 프로세스를 나타내는 도면이다.

[0004] 메모리 구동 주변회로 넷리스트(110)를 Verilog 모델링(120)을 통해 변환한 후, 시뮬레이션(Simulation) 검증(130)을 수행할 수 있다. 이때 모델링(Modeling)된 Verilog HDL형태의 메모리 구동 주변회로(Memory Peripheral Logic)은 합성이 불가능하기에 FPGA위에 다운로드 할 수 없다. 따라서 시뮬레이션 소프트웨어(Simulation Software)(ModelSim/Questa)를 통한 검증만이 가능하다. 메모리 설계의 테스트 특성상 다양하고 긴 입력 벡터(Input Vector)를 통해 검증해야 할 동작이 매우 많기 때문에 종래 기술에 따른 시뮬레이션에는 매우 오랜 시간이 걸리게 되는 문제점이 있다.

발명의 내용

해결하려는 과제

[0005] 본 발명이 이루고자 하는 기술적 과제는 메모리 구동 주변회로를 검증할 때 다양하고 긴 벡터를 적용한 시뮬레이션은 많은 시간이 소요된다는 문제점을 개선하기 위해 FPGA와 프로세서를 이용한 새로운 검증 방법 및 시스템을 제안함으로써 검증시간을 단축하고자 한다.

과제의 해결 수단

[0006] 일 측면에 있어서, 본 발명에서 제안하는 메모리 구동 주변 회로 검증시간을 단축하기 위한 임베디드 DDR 메모리를 이용한 검증 방법은 입력 벡터를 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환하여 프로세스 영역에 전달하는 단계, 상기 전달된 입력 벡터를 프로그램 로직 영역에서 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환하는 단계, 쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 상기 입력 벡터를 비교하고, 읽기 동작 시 상기 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 상기 메모리 구동 주변 회로를 검증하는 단계를 포함할 수 있다.

[0007] 상기 쓰기 동작 시 상기 프로세스 영역의 프로세서를 통해 전달된 입력 벡터의 정보들은 인터페이스 로직을 거쳐 메모리 구동 주변회로의 인터페이스의 입력인 CA 및 DQ 신호로 변경되어 전달되고, 출력으로 어드레스 정보와 데이터를 출력할 수 있다.

[0008] 상기 인터페이스 로직의 제어를 통해 상기 프로세서와 연동하여 상기 임베디드 메모리의 해당 어드레스에 상기 쓰기 동작을 수행할 수 있다.

[0009] 상기 읽기 동작 시 상기 입력 벡터를 입력 받은 인터페이스 로직이 상기 임베디드 메모리의 데이터를 상기 임시 메모리로 읽어 들이는 선행 동작을 수행한 후, 상기 메모리 구동 주변회로의 입력인 CA가 생성되고, 출력으로

어드레스를 상기 임시 메모리로 전달하면 읽어 들인 상기 임베디드 메모리의 데이터를 가져올 수 있다.

[0010] 상기 메모리 구동 주변회로와 데이터를 주고 받을 공간으로 상기 임베디드 메모리를 사용할 수 있다.

[0011] 상기 주변장치 로직과 상기 임베디드 메모리 사이에는 임지 저장 공간이 존재할 수 있다.

[0012] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 메모리 구동 주변 회로 검증시간을 단축하기 위한 임베디드 DDR 메모리를 이용한 검증 시스템은 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환된 입력 벡터를 전달받는 프로세스 영역, 상기 전달된 입력 벡터를 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환하는 프로그램 로직 영역를 포함하고, 상기 프로세스 영역은 쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 상기 입력 벡터를 비교하고, 읽기 동작 시 상기 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 상기 메모리 구동 주변회로를 검증할 수 있다.

발명의 효과

[0013] 본 발명의 실시예들에 따르면 기존에 소프트웨어(Software)(컴퓨터)상에서 진행하던 시뮬레이션(Simulation)이 다양하고 긴 벡터를 적용시킬 경우에 매우 긴 시간이 소요된다는 문제점을 개선할 수 있다. 이를 위해 임베디드 메모리(Embedded_Memory)를 메모리 셀(Memory Cell)로 활용한 본 시스템을 구축함으로써 하드웨어(Hardware)상에서의 로직(Logic)검증이 가능하여 시뮬레이션 시간을 단축할 수 있다.

도면의 간단한 설명

[0014] 도 1은 종래 기술에 따른 메모리 구동 주변 회로 검증 프로세스를 나타내는 도면이다.

도 2는 본 발명의 일 실시예에 따른 메모리 구동 주변 회로 검증 프로세스를 나타내는 도면이다.

도 3은 본 발명의 일 실시예에 따른 임베디드 DDR 메모리를 이용한 검증 시스템을 나타내는 도면이다.

도 4는 본 발명의 일 실시예에 따른 임베디드 DDR 메모리를 이용한 검증 방법을 설명하기 위한 흐름도이다.

도 5는 본 발명의 일 실시예에 따른 쓰기 동작을 설명하기 위한 도면이다.

도 6은 본 발명의 일 실시예에 따른 읽기 동작을 설명하기 위한 도면이다.

도 7은 본 발명의 일 실시예에 따른 쓰기 및 읽기 동작 시 전달되는 파형을 나타내는 도면이다.

도 8은 본 발명의 일 실시예에 따른 쓰기 및 읽기 동작의 시뮬레이션 결과이다.

발명을 실시하기 위한 구체적인 내용

[0015] 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0016] 도 2는 본 발명의 일 실시예에 따른 메모리 구동 주변 회로 검증 프로세스를 나타내는 도면이다.

[0017] 종래 기술에 따른 소프트웨어(Software) 상에서의 긴 검증시간을 개선하기 위한 하드웨어(Hardware) 상에서의 검증시스템을 제안한다. 도 1의 프로세스는 도 2의 프로세스로 변경될 수 있다. 메모리 구동 주변회로 넷리스트(210)를 Verilog 모델링(220)을 통해 변환할 수 있고, 이때 Verilog 모델링(220) 과정을 합성이 가능한 모델링(Modeling)과정으로 변경하여 진행하는 것이 우선작업이 되어야 한다. 합성이 가능하도록 모델링(Modeling)된 회로는 FPGA에 로드(Load)(230) 하는 것이 가능해진다. 다시 말해, 하드웨어(Hardware)상에서 구현이 가능해질 수 있다. FPGA에 구현된 회로를 검증하기 위해서는 이를 위한 시스템을 설계해야 한다. 따라서, 제안된 시스템을 통해 검증(240)을 수행할 수 있다.

[0018] 도 3은 본 발명의 일 실시예에 따른 임베디드 DDR 메모리를 이용한 검증 시스템을 나타내는 도면이다.

[0019] 제안하는 임베디드 DDR 메모리를 이용한 검증 시스템은 설계 시 단순 FPGA보드를 사용해서 구성하지 않고, 다양한 입력 벡터(Input vector)를 제어할 수 있는 프로세서(Processor)를 접목시킨 SoC 보드(Board)를 사용하여 구성하였다. 메모리 구동 주변회로(Memory Peripheral Logic)을 검증하기 위한 메모리 셀(Memory Cell)역할로써는 보드내의 임베디드 메모리(Embedded_Memory)를 활용하였다. 프로세서(Processor)와 FPGA를 연동시키는 시

스템을 설계함에 있어서 Xilinx사의 IP 및 프로토콜(Protocol)을 사용하게 되는데 이를 사용할 때 생기는 딜레이(Delay) 지연과 불규칙 인에이블(Enable) 신호에 적응할 수 있는(Adaptive)한 로직 블록(Logic Block)을 구현하였다. 또한 테스트할 로직과 입력 벡터(Input Vector)와의 제어를 위한 코어 로직(Core_Logic)을 설계하고 이러한 로직들을 통하여 전체 시스템을 구성하였다.

[0020] 시스템 구축에 사용되는 보드는 프로세서(Processor)(321) 기반 플랫폼이다. SoC 보드(310)에 탑재된 칩은 FPGA 뿐만 아니라 프로세서(Processor)(321)를 사용할 수 있기에 PS(Processing System)영역(320)과 PL(Programmable Logic)(330)영역을 모두 사용하여 시스템을 구축한다. 또한 보드상의 임베디드 DDR3 메모리(Embedded DDR3 Memory)를 검증할 메모리 구동 주변회로(331)의 메모리 셀(Memory Cell)영역으로써 활용한다.

[0021] 도 4는 본 발명의 일 실시예에 따른 임베디드 DDR 메모리를 이용한 검증 방법을 설명하기 위한 흐름도이다.

[0022] 임베디드 DDR 메모리를 이용한 검증 방법은 입력 벡터를 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환하여 프로세스 영역에 전달하는 단계(410), 상기 전달된 입력 벡터를 프로그램 로직 영역에서 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환하는 단계(420), 쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 상기 입력 벡터를 비교하고, 읽기 동작 시 상기 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 상기 메모리 구동 주변 회로를 검증하는 단계(430)를 포함할 수 있다.

[0023] 단계(410)에서, 입력 벡터를 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환하여 프로세스 영역에 전달할 수 있다.

[0024] 도 3을 참조하여 시스템의 동작을 설명하면, 가장먼저 기존의 입력 벡터(Input Vector)를 시스템의 형식에 맞추어 변환시켜 프로세서(Processor)(321)(다시 말해, PS 영역(320))에 전달할 수 있다.

[0025] 단계(420)에서, 상기 전달된 입력 벡터를 프로그램 로직 영역에서 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환할 수 있다.

[0026] 도 3을 참조하면, 단계(410)에서 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환된 입력 벡터를 다시 PL 영역(330)의 설계된 블록을 거치면서 메모리 구동 주변회로(331)에 맞는 신호로 변환되어 전달될 수 있다. 이와 같은 과정을 거쳐 검증작업이 시작될 수 있다.

[0027] 이때 메모리 구동 주변회로(331)와 데이터(Data)를 주고받을 메모리 셀(Memory Cell)로 큰 공간이 필요하기에 임베디드 메모리(Embedded Memory)(340)를 그 영역으로 사용한다(기존의 테스트에서는 바로 메모리 셀(Memory Cell)로 전달되어야 하지만 그 영역이 상당히 크기 때문에 FPGA위에 메모리 셀(Memory Cell)구현이 불가함).

[0028] 단계(430)에서, 쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 상기 입력 벡터를 비교하고, 읽기 동작 시 상기 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 상기 메모리 구동 주변회로를 검증할 수 있다.

[0029] 상기 쓰기 동작 시 프로세스 영역의 프로세서를 통해 전달된 입력 벡터의 정보들은 인터페이스 로직을 거쳐 메모리 구동 주변회로의 인터페이스의 입력인 CA 및 DQ 신호로 변경되어 전달되고, 출력으로 어드레스 정보와 데이터를 출력할 수 있다. 또한, 인터페이스 로직(Interface_Logic)(520)의 제어를 통해 상기 프로세서와 연동하여 상기 임베디드 메모리의 해당 어드레스에 쓰기 동작을 수행할 수 있다.

[0030] 상기 읽기 동작 시 상기 입력 벡터를 입력 받은 인터페이스 로직이 임베디드 메모리의 데이터를 임시 메모리로 읽어 들이는 선행 동작을 수행한 후, 메모리 구동 주변회로의 입력인 CA가 생성되고, 출력으로 어드레스를 상기 임시 메모리로 전달하면 읽어 들인 상기 임베디드 메모리의 데이터를 가져올 수 있다.

[0031] 도 3을 참조하면, 테스트의 패스/페일(Pass/Fail) 결과는 쓰기(Write)시에는 임베디드 메모리(Embedded Memory)(340)에 쓰여진 데이터(Data)와 입력(Input)으로 들어온 데이터 DQ(351)를 비교할 수 있다. 읽기(Read)시에도 같은 방식으로 임베디드 메모리(Embedded Memory)(340)에 쓰여있던 데이터(쓰기 (Write)동작 시 패스/페일(Pass/Fail)가 판정된 데이터)와 메모리 구동 주변회로의(Peripheral Logic)(331)을 거쳐 나온 데이터 DQ(351)를 비교하여 판정할 수 있다. 그리고, 메모리 구동 주변회로의(Peripheral Logic)(331)과 임베디드 메모리(Embedded Memory)(340)사이에는 임시저장공간인 임시 메모리(Temp_Memory)(333)가 존재한다.

[0032] 도 5는 본 발명의 일 실시예에 따른 쓰기 동작을 설명하기 위한 도면이다.

[0033] 도 5를 참조하여 쓰기 동작 흐름(Write Operation Flow)에 대하여 더욱 상세히 설명한다. 프로세서(Processor)(510)을 통해 전달된 입력 벡터(Input Vector)의 정보들은 인터페이스 로직(Interface_Logic)(520)을 거치면서 메모리 구동 주변 회로(Memory Peripheral Circuit)(530)의 인터페이스의 입력(Input)인 CA(541)와 DQ(542)신호로 변경되어 전달되며 출력(Output)으로 어드레스(Address)(543)정보와 데이터(Data)(544)를 출력한다. 출력된 데이터(Data)(544)와 어드레스(Address)(543)정보는 실제 저장될 공간인 임베디드 메모리(Embedded Memory)(560)로 전달되는데 그전에 임시 메모리(Temp_Memory)(550)에 머물러 있게 된다. 그 후 인터페이스 로직(Interface_Logic)(520)의 제어를 통하여 프로세서(Processor)(510)와의 연동을 통해 인터페이스 로직(Interface_Logic)(520)의 해당 어드레스(Address)(543)로 데이터(Data)(544)가 쓰기(Write)될 수 있다.

[0034] 도 6은 본 발명의 일 실시예에 따른 읽기 동작을 설명하기 위한 도면이다.

[0035] 도 6를 참조하여 읽기 동작 흐름(Read Operation Flow)에 대하여 더욱 상세히 설명한다. 읽기(Read)의 경우, 프로세서(Processor)(610)로부터 처음 입력 벡터(Input Vector)를 받아들인 인터페이스 로직(Interface_Logic)(620)이 임시 메모리(Temp_Memory)(650)로 임베디드 메모리(Embedded Memory)(660)의 데이터(Data)를 미리 읽어 들여야 하는 선행 작업이 이루어진다. 이 과정이 끝난 후 메모리 구동 주변 회로(Memory Peripheral Circuit)(630)의 입력(Input)인 CA(641)가 생성되어 출력(Output)으로 어드레스(Address)(643)를 임시 메모리(Temp_Memory)(650)로 전달하면 미리 읽어두었던 데이터(Data)644()를 가져오게 되는 것이다.

[0036] 도 7은 본 발명의 일 실시예에 따른 쓰기 및 읽기 동작 시 전달되는 파형을 나타내는 도면이다.

[0037] 도 7a는 본 발명의 일 실시예에 따른 쓰기(Write) 동작 시 DQ가 임베디드 메모리로 전달되는 파형을 나타내는 도면이고, 도 7b는 본 발명의 일 실시예에 따른 읽기(Read) 동작 시 임베디드 메모리로부터 데이터를 읽어와 DQ로 전달되는 파형을 나타내는 도면이다. 예를 들어, 임베디드 메모리(Embedded Memory)를 셀(cell)로 사용하는 제어 블록 코딩(FPGA 구현)을 이용할 수 있다.

[0038] 쓰기(Write) 동작 시에는 DQ를 통해 임시 메모리(Temp_Memory)로 데이터가 전달되고 읽기(Read) 동작 시에는 임시 메모리(Temp_Memory)로부터 데이터를 읽어 들여 DQ로 전달된다.

[0039] 쓰기(Write) 동작 시 임시 메모리(Temp_Memory)로 DQ가 전달되고 나면, 연속적인 동작으로 그 데이터가 임베디드 메모리(Embedded Memory)로 전달될 수 있다. 이에 대한 시뮬레이션 과정을 도 7a에 나타내었다.

[0040] 읽기(Read) 동작 시에는 임베디드 메모리(Embedded Memory)에 있던 데이터가 임시 메모리(Temp_Memory)로 전달되고, 연속적인 동작으로 그 데이터가 DQ로 전달될 수 있다. 이에 대한 시뮬레이션 과정을 도 7b에 나타내었다.

[0041] 도 8은 본 발명의 일 실시예에 따른 쓰기 및 읽기 동작의 시뮬레이션 결과이다.

[0042] 프로세서(Processor)를 사용하여 전체 검증 시스템 구축하는 과정의 예시에 대하여 설명한다. LPDDR2의 어드레싱(Addressing) 스펙 중에서 512Mb(x32)을 타겟으로 하여 시뮬레이션을 진행하였다. 뱅크(Bank) 수는 4, 로우 어드레스(Row Address)는 R0~R12, 콜 어드레스(Col Address)는 C0~C8이다. 도 8은 BL: 16, RL: 6, WL: 3 의 입력 벡터(Input Vector)를 시뮬레이션 한 결과 중 뱅크2(Bank2)의 동작 부분이다. 뱅크2(Bank2)의 칼럼 어드레스(Column Address) 5E2로 쓰기 입력(Write Input)이 들어왔고, 이때 로우 어드레스(Row Address)는 이전에 뱅크2(Bank2) 활성화(Activate) 명령 시 셋팅된 131B가 된다. 데이터의 값은 Bank2, Row: 131B, Col: 5E2의 어드레스(Address)에 매칭되는 임베디드 메모리(Embedded Memory) 위에 쓰여진 값이며 DQ는 입력으로 들어간 데이터이다. 데이터와 DQ의 값을 비교하여 결과가 같으면 동작이 제대로 이루어졌다는 것이기 때문에 P(Pass) 신호가 출력된다.

[0043] 표 1은 종래 기술에 따른 방법과 제안하는 방법의 검증 시간을 비교한 것이다.

[0044] <표 1>

Input Vector Line수	기존 Simulation 검증시간	제안된 시스템 Simulation 검증시간
100줄	5분	1분10초
1,000줄	6~7분	1분15초
10,000줄	10분	1분30초

[0045]

[0046] 표 1에 이전의 소프트웨어(Software)상에서의 검증시간과 하드웨어(Hardware)상에서의 검증시간을 비교하여 나타내었다. 표 1의 결과시간은 검증을 위한 셋팅 시간이 포함되어있는데 기존의 경우 약 4분 정도가 포함되고 제안된 시스템의 경우 약 1분 정도의 셋팅 시간이 포함되어있다. 결과적으로 기존검증시간은 검증할 입력 벡터 라인(Input Vector Line)의 증가에 따라 분~시간단위로 증가하지만 제안된 방안의 검증시간은 초~분단위로 증가한다. 검증할 입력 벡터 라인(Input Vector Line)의 수가 급격히 증가할수록 검증시간단축의 효과가 급증함을 알 수 있다.

[0047] 따라서, 종래 기술에 따른 소프트웨어(Software)(다시 말해, 컴퓨터)상에서 진행하던 시뮬레이션은 다양하고 긴 벡터(Vector)를 적용시킬 경우 매우 긴 시간이 소요되지만, 제안하는 방법은 임베디드 메모리(Embedded_Memory)를 메모리 셀(Memory Cell)로 이용하여 하드웨어(Hardware)상에서의 로직(Logic)검증이 가능해지기 때문에 시뮬레이션 시간을 단축할 수 있다.

[0048] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0049] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0050] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를

포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

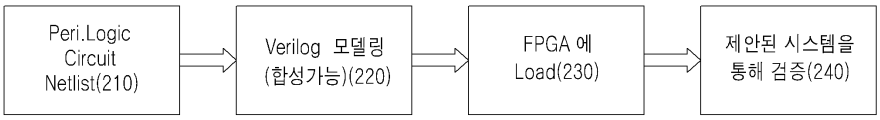
[0051] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0052] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

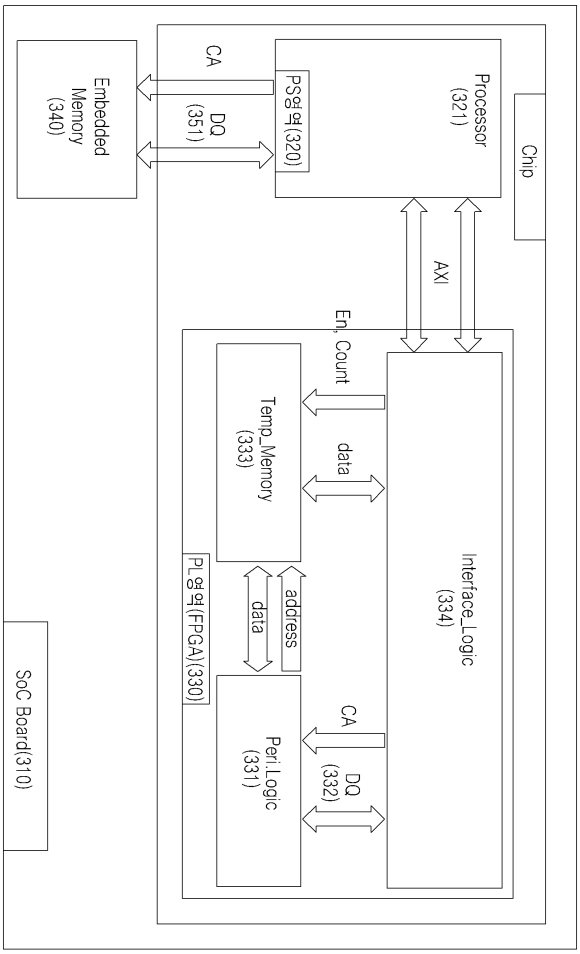
도면

도면1

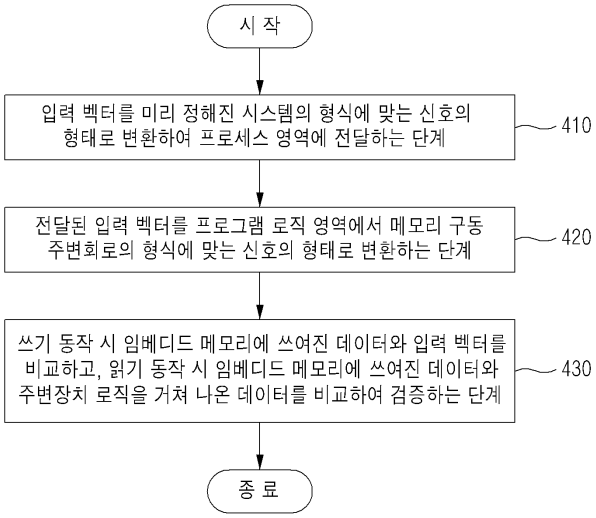
도면2



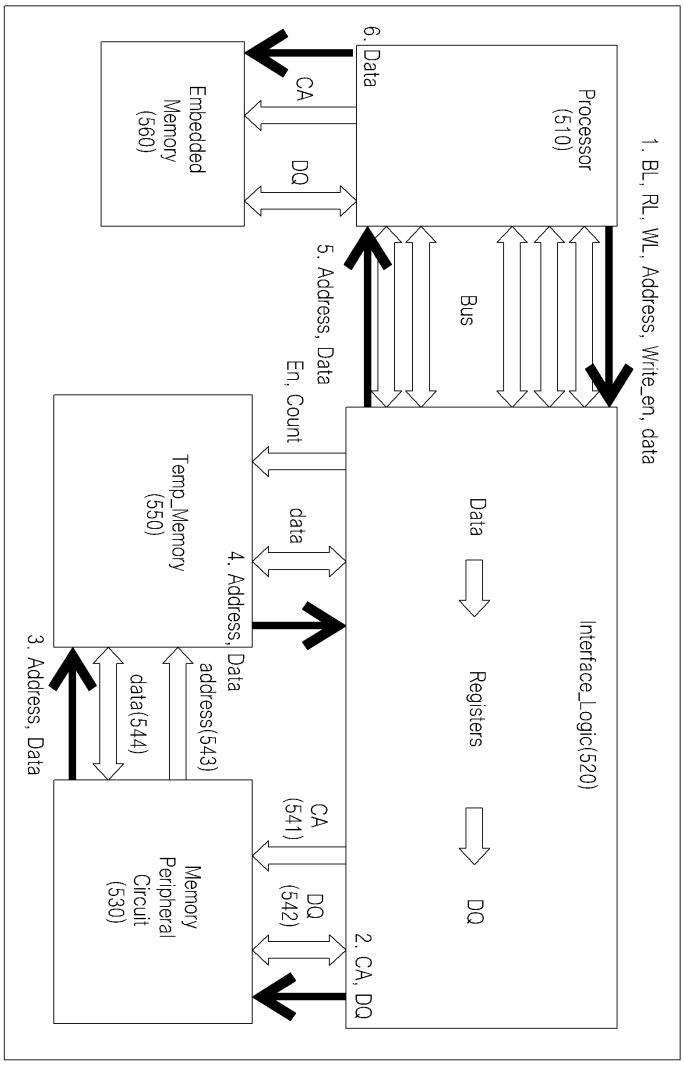
도면3


Embedded Memory (340)
CA
DQ (351)
PS영역(320)
Processor (321)
Chip
AXI
En, Count
Temp_Memory (333)
data
Interface_Logic (334)
PL영역(FPGA)(330)
data
address
CA
Peri.Logic (331)
DQ (332)
SoC Board(310)

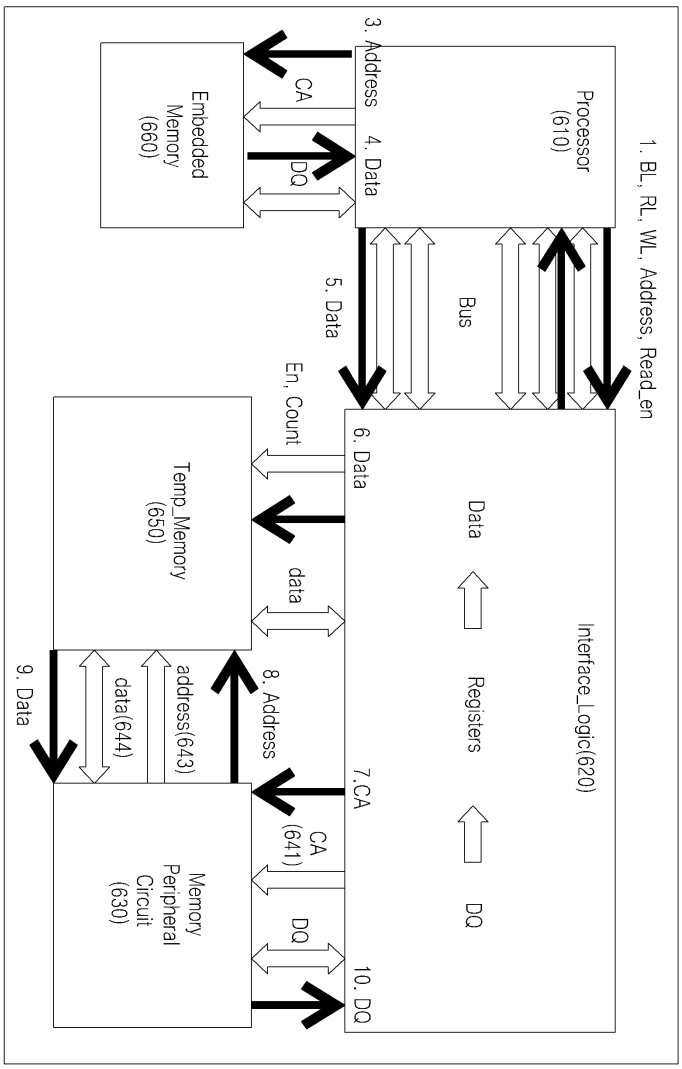
도면4


시 작
입력 벡터를 미리 정해진 시스템의 형식에 맞는 신호의 형태로 변환하여 프로세스 영역에 전달하는 단계
410
전달된 입력 벡터를 프로그램 로직 영역에서 메모리 구동 주변회로의 형식에 맞는 신호의 형태로 변환하는 단계
420
쓰기 동작 시 임베디드 메모리에 쓰여진 데이터와 입력 벡터를 비교하고, 읽기 동작 시 임베디드 메모리에 쓰여진 데이터와 주변장치 로직을 거쳐 나온 데이터를 비교하여 검증하는 단계
430
종 료

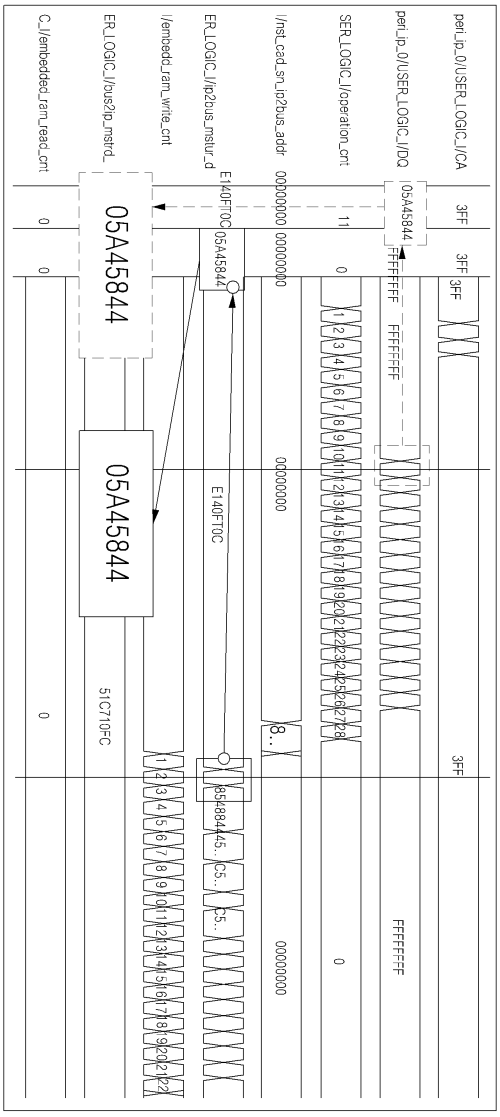
도면5


Processor
(510)
1. BL, RL, WL, Address, Write_en, data
Bus
Interface_Logic(520)
Data
Registers
DQ
2. CA, DQ
CA
(541)
DQ
(542)
Memory
Peripheral
Circuit
(530)
3. Address, Data
data(544)
address(543)
4. Address, Data
Temp_Memory
(550)
data
5. Address, Data
En, Count
6. Data
CA
DQ
Embedded
Memory
(560)

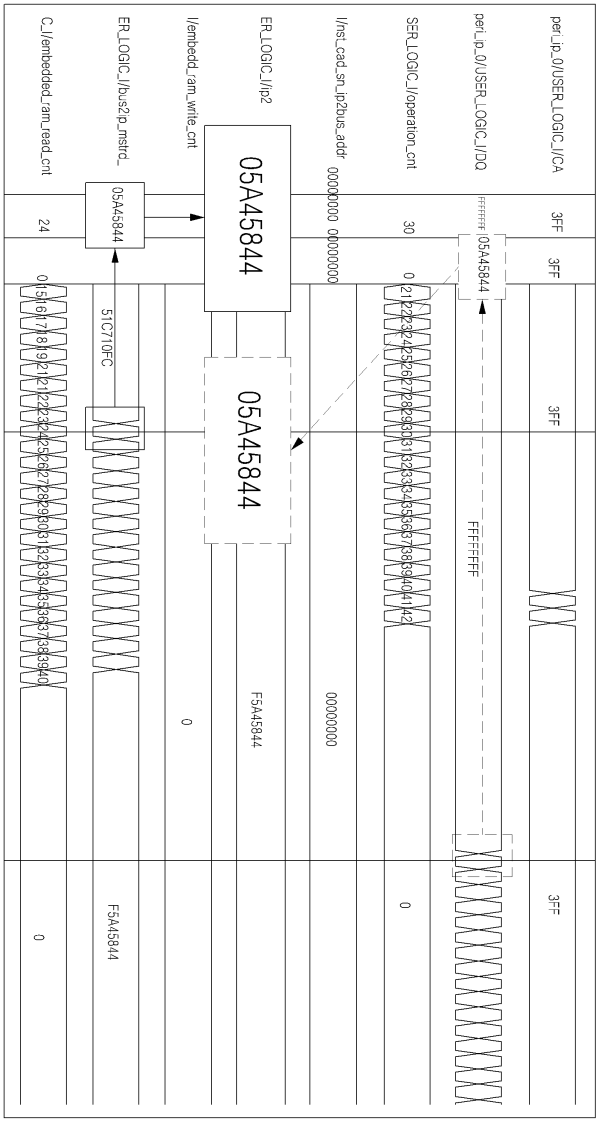
도면6


Processor
(610)
1. BL, RL, WL, Address, Read_en
Bus
Interface_Logic(620)
Data
Registers
DQ
3. Address
4. Data
CA
DQ
Embedded
Memory
(660)
5. Data
En, Count
6. Data
Temp_Memory
(650)
data
8. Address
address(643)
data(644)
9. Data
7.CA
CA
(641)
DQ
10. DQ
Memory
Peripheral
Circuit
(630)

도면7a


peri_ip_0/USER_LOGIC_I/CA
peri_ip_0/USER_LOGIC_I/DQ
SER_LOGIC_I/operation_cnt
//nst_card_sm_ip2bus_addr
ER_LOGIC_I/ip2bus_mstwr_d
//embedd_ram_write_cnt
ER_LOGIC_I/bus2ip_mstrd_
C_I/embedded_ram_read_cnt
3FF
05A45844
FFFFFFFF
11
00000000
E140FT0C
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
8..
85484445..
C5..
51C710FC

도면7b


peri_ip_0/USER_LOGIC_I/CA
peri_ip_0/USER_LOGIC_I/DQ
SER_LOGIC_I/operation_cnt
I/rst_cad_sn_ip2bus_addr
ER_LOGIC_I/ip2
I/embeddd_ram_write_cnt
ER_LOGIC_I/bus2ip_mstrd_
C_I/embedded_ram_read_cnt
3FF
FFFFFFFF
05A45844
30
00000000
24
51C710FC
F5A45844
0

도면8

Compare DQ with DDR start(Write OP)
Embedded Memory Bank=2, Rom=131B, Co1=5E2, Data=[47AD8C84] : DQ0=[47AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E3 Data=[C7AD8C84] : DQ1=[C7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E4, Data=[27AD8C84] : DQ2=[27AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E5, Data=[A7AD8C84] : DQ3=[A7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E6, Data=[67AD8C84] : DQ4=[67AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E7, Data=[E7AD8C84] : DQ5=[E7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E8, Data=[17AD8C84] : DQ6=[17AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E9, Data=[97AD8C84] : DQ7=[97AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EA, Data=[57AD8C84] : DQ8=[57AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EB, Data=[D7AD8C84] : DQ9=[D7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EC, Data=[37AD8C84] : DQ10=[37AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5ED, Data=[B7AD8C84] : DQ11=[B7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EE, Data=[77AD8C84] : DQ12=[77AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EF, Data=[F7AD8C84] : DQ13=[F7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5F0, Data=[7AD8C84] : DQ14=[7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5F1, Data=[87AD8C84] : DQ15=[87AD8C84], Compare Result = P
Compare DQ with DDR start(Write OP)
Embedded Memory Bank=2, Rom=131B, Co1=5E0, Data=[2BDDD4FF] : DQ0=[2BDDD4FF], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E1, Data=[BFEBF79C] : DQ0=[BFEBF79C], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E2, Data=[47AD8C84] : DQ0=[47AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E3 Data=[C7AD8C84] : DQ1=[C7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E4, Data=[27AD8C84] : DQ2=[27AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E5, Data=[A7AD8C84] : DQ3=[A7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E6, Data=[67AD8C84] : DQ4=[67AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E7, Data=[E7AD8C84] : DQ5=[E7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E8, Data=[17AD8C84] : DQ6=[17AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5E9, Data=[97AD8C84] : DQ7=[97AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EA, Data=[57AD8C84] : DQ8=[57AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EB, Data=[D7AD8C84] : DQ9=[D7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EC, Data=[37AD8C84] : DQ10=[37AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5ED, Data=[B7AD8C84] : DQ11=[B7AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EE, Data=[77AD8C84] : DQ12=[77AD8C84], Compare Result = P
Embedded Memory Bank=2, Rom=131B, Co1=5EF, Data=[F7AD8C84] : DQ13=[F7AD8C84], Compare Result = P
Compare DQ with DDR start(Write OP)