euresys
Empowering Computer Vision

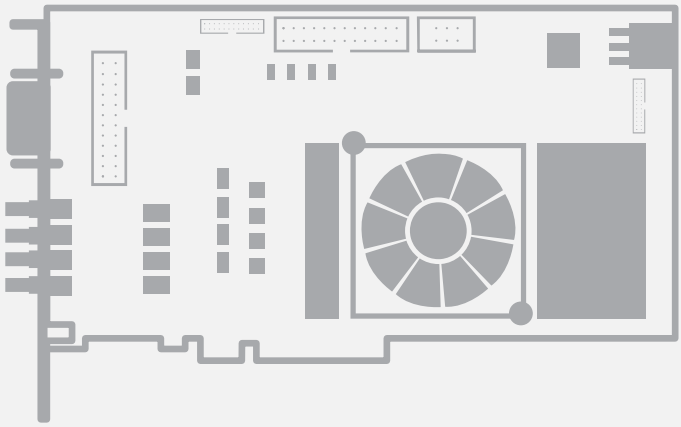
사용자 설명서

CustomLogic

CustomLogic 12.2 User Guide

3602 Coaxlink Octo

3603 Coaxlink Quad CXP-12



사용 약관

이 문서는 CustomLogic 12.2.1 의 부속 자료입니다(문서 빌드 2100).
www.euresys.com

목차

1. CustomLogic 소개

1.1. 원칙

CustomLogic을 통해 사용자는 "CustomLogic"펌웨어 변형이 장착된 Euresys 프레임 그래버의 FPGA (Field Programmable Gate Array)에서 주문형 온보드 이미지 처리를 추가할 수 있습니다.

CustomLogic에는 사용자 지정 이미지 처리 기능을 프레임 그래버와 상호 연결하는 데 사용되는 문서화된 인터페이스를 제공하는 디자인 프레임 워크가 포함되어 있습니다.

1.2. 유효성

참고CustomLogic 설치 패키지는 요청 시 해당 지역 영업 사무소에서만 이용할 수 있습니다.

CustomLogic은 다음 제품 및 펌웨어 변형에 사용할 수 있습니다.

3602 Coaxlink Octo

펌웨어 변형	설명
1- 카메라, 맞춤형 로직	◻ CXP-6 DIN 4 CoaXPress 인터페이스 ◻ 1- 또는 2-또는 4-연결 영역 스캔 카메라 1개 ◻ 2GB RAM DDR4 온보드 메모리 ◻ 8 레인 Gen 3 PCI Express 인터페이스

3603 Coaxlink Quad CXP-12

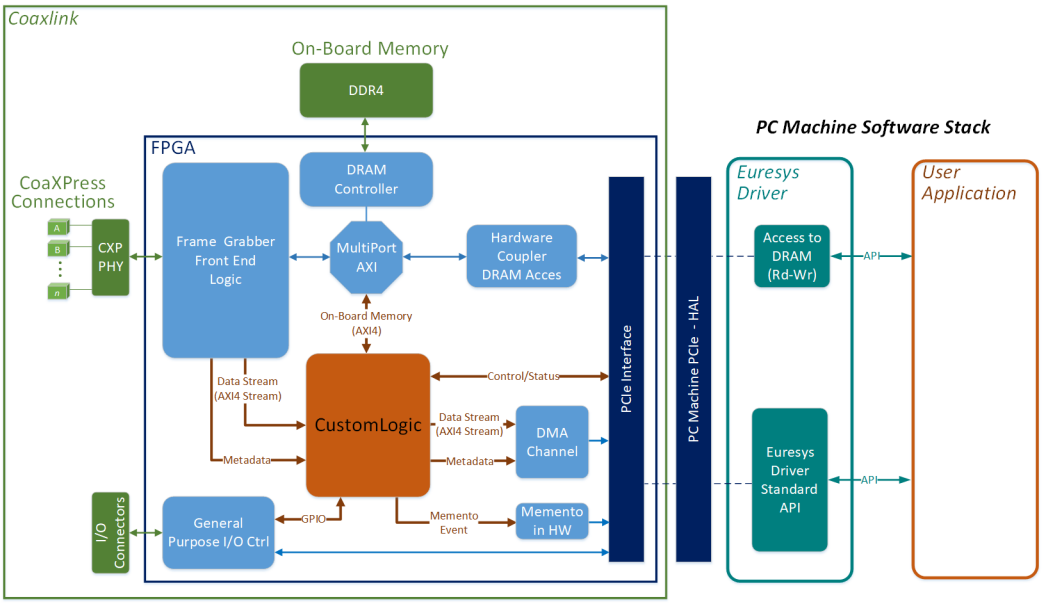
펌웨어 변형	설명
1- 카메라, 맞춤형 로직	◻ CXP-12 HD-BNC 4 CoaXPress 인터페이스 ◻ 1- 또는 2-또는 4-연결 영역 스캔 카메라 1개 ◻ 2GB RAM DDR4 온보드 메모리 ◻ 8 레인 Gen 3 PCI Express 인터페이스
4- 카메라, 맞춤형 로직	◻ CXP-12 HD-BNC 4 CoaXPress 인터페이스 ◻ 1-연결 영역 스캔 카메라 4개 ◻ 2GB RAM DDR4 온보드 메모리 ◻ 8 레인 Gen 3 PCI Express 인터페이스

1.3. 프레임워크

Coaxlink 프레임 워크는 다음과 같은 내장 모듈을 제공합니다:

1. 전체 기능의 CoaXPress 프레임 그래버.
2. 온보드 메모리 인터페이스.
3. DMA 백엔드 채널이 있는 PCI Express 인터페이스.
4. 하드웨어 이벤트 로깅 시스템의 Memento.
5. 사용자는 Eureys Driver API를 통해 액세스를 등록합니다.

샘플 블록 다이어그램



Coaxlink CustomLogic 모델

2. CustomLogic 인터페이스

2.1. 글로벌 신호

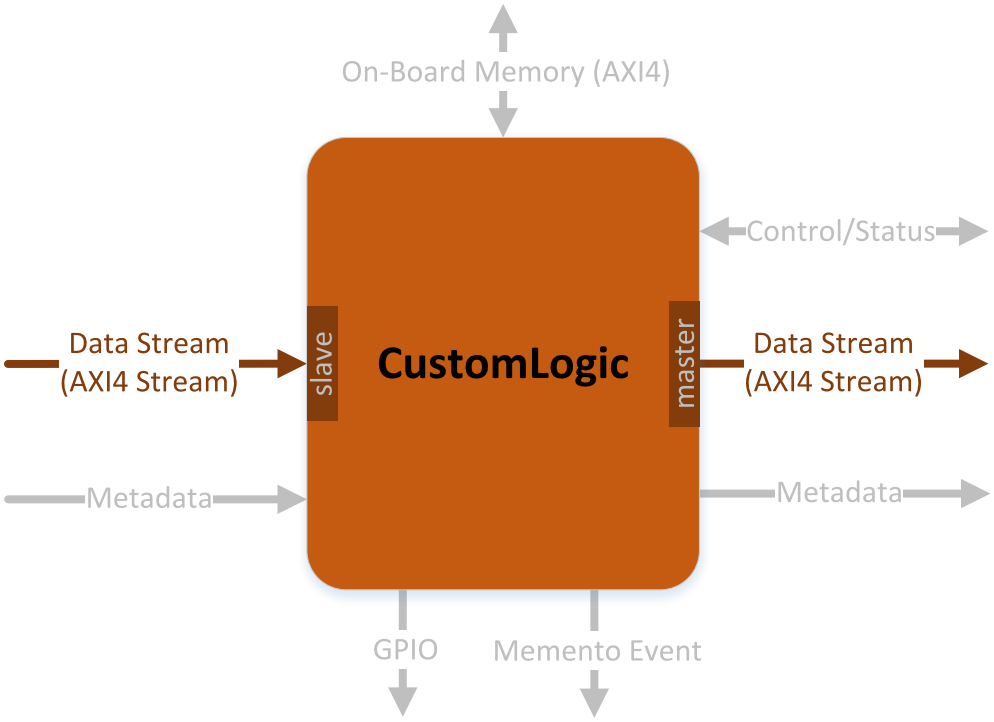
사용 가능한 모든 인터페이스는 250MHz의 동일한 클럭 도메인에 있으며 *CustomLogic* 글로벌 신호는 다음과 같습니다.

신호	지시문	설명
clk250	입력	모든 *CustomLogic* 인터페이스에 공통적인 250 MHz 클럭 소스.
srst250	입력	동기식 리셋(clk250) PCI 익스프레스 리셋 중에 발휘.

2.2. 데이터(픽셀) 스트림 인터페이스

데이터 스트림 인터페이스는 AMBA AXI4-Stream 프로토콜 사양을 기반으로 합니다:

- 슬레이브 측에서 *CustomLogic*은 CoaXPress 장치(예: CoaXPress 카메라)에서 얻은 이미지를 수신합니다.
- 마스터 측에서 데이터 스트림 인터페이스는 *CustomLogic*에 의해 생성된 결과 이미지/데이터를 PCI Express DMA 백엔드 채널로 전송합니다.



슬레이브 인터페이스 신호

신호	폭	지시문	설명
s_axis_aresetn	N*1	입력	ARESETn은 AXI4-Stream 인터페이스를 재설정합니다. 이 펄스는 수집 중지 명령 (`DSStopAcquisition`)이 실행될 때 발생합니다. 이 신호는 *CustomLogic* 내부 데이터 스트림 파이프라인을 지우는 데 사용해야 합니다.
s_axis_tvalid	N*1	입력	TVALID는 해당 마스터 인터페이스가 유효한 전송을 수행하고 있음을 나타냅니다. 전송은 TVALID와 TREADY가 모두 발휘될 때 이루어집니다.
s_axis_tready	N*1	출력	TREADY는 *CustomLogic*이 현재 주기에서 전송을 수락할 수 있음을 나타냅니다.
s_axis_tdata	N*W	입력	TDATA는 인터페이스를 통해 전달되는 데이터를 제공하는 데 사용되는 기본 페이로드입니다.
s_axis_tuser	N*4	입력	TUSER는 데이터 스트림과 함께 전송할 수있는 사용자 정의 측 파대 정보입니다. TUSER 콘텐츠는 다음과 같이 인코딩됩니다: □ TUSER [0] => Start-of-Frame (SOF) □ TUSER [1] => Start-of-Line (SOL) □ TUSER [2] => End-of-Line (EOL) □ TUSER [3] => End-of-Frame (EOF)

노트
너비 열에서: “*N*"은 *CustomLogic* 변형에서 지원하는 장치/카메라 수인 총 인터페이스 슬롯 수를 나타내고 “*W*"는 장치/카메라 당 스트림 데이터 너비 인 STREAM_DATA_WIDTH를 나타냅니다.

- **3602 Coaxlink Octo** (1- 카메라, 맞춤형 로직) => *N* = 1; *W* = 128;
- **3603 Coaxlink Quad CXP-12** (1- 카메라, 맞춤형 로직) => *N* = 1; *W* = 256;
- **3603 Coaxlink Quad CXP-12** (4- 카메라, 맞춤형 로직) => *N* = 4; *W* = 64;

마스터 인터페이스 신호

신호	폭	지시문	설명
m_axis_tvalid	N*1	출력	TVALID는 *CustomLogic*이 올바른 전송을 유도하고 있음을 나타냅니다. 전송은 TVALID와 TREADY가 모두 발휘될 때 이루어집니다.
m_axis_tready	N*1	입력	TREADY는 PCI 익스프레스 DMA 백엔드가 현재 주기에서 전송을 수락할 수 있음을 나타냅니다.
m_axis_tdata	N*W	출력	TDATA는 인터페이스를 통해 전달되는 데이터를 제공하는 데 사용되는 기본 페이로드입니다.
m_axis_tuser	N*4	출력	TUSER는 데이터 스트림과 함께 전송할 수있는 사용자 정의 측 파대 정보입니다. TUSER 콘텐츠는 다음과 같이 인코딩됩니다: □ axis_tuser_out[0] => Start-of-Buffer (SOB) □ axis_tuser_out[1] => *예약됨* □ axis_tuser_out[2] => *예약됨* □ axis_tuser_out[3] => End-of-Buffer (EOB)

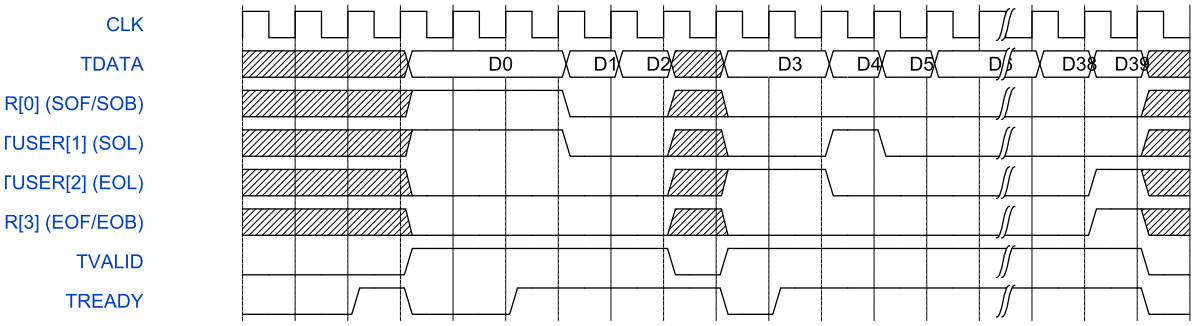
노트
너비 열에서: "*N*"은 *CustomLogic* 변형에서 지원하는 장치/카메라 수인 총 인터페이스 슬롯 수를 나타내고 "*W*"는 장치/카메라 당 스트림 데이터 너비 인 STREAM_DATA_WIDTH를 나타냅니다.

- **3602 Coaxlink Octo** (1- 카메라, 맞춤형 로직) => *N* = 1; *W* = 128;
- **3603 Coaxlink Quad CXP-12** (1- 카메라, 맞춤형 로직) => *N* = 1; *W* = 256;
- **3603 Coaxlink Quad CXP-12** (4- 카메라, 맞춤형 로직) => *N* = 4; *W* = 64;

노트
CustomLogic 마스터 측에서 `m_axis_tuser` 신호에는 PCI Express DMA 백엔드를 제어하는 기능이 있습니다. `m_axis_tuser` 에 의해 전달되는 플래그는 다음과 같이 해석됩니다:

- *Start-of-Buffer*:이 플래그를 포함한 사이클은 새로운 버퍼를 시작합니다.
- *End-of-Buffer*:이 플래그를 포함하는 사이클은 새로운 전송을 수용 할 여유 공간이 남아 있어도 버퍼를 종료합니다.

타이밍 다이어그램



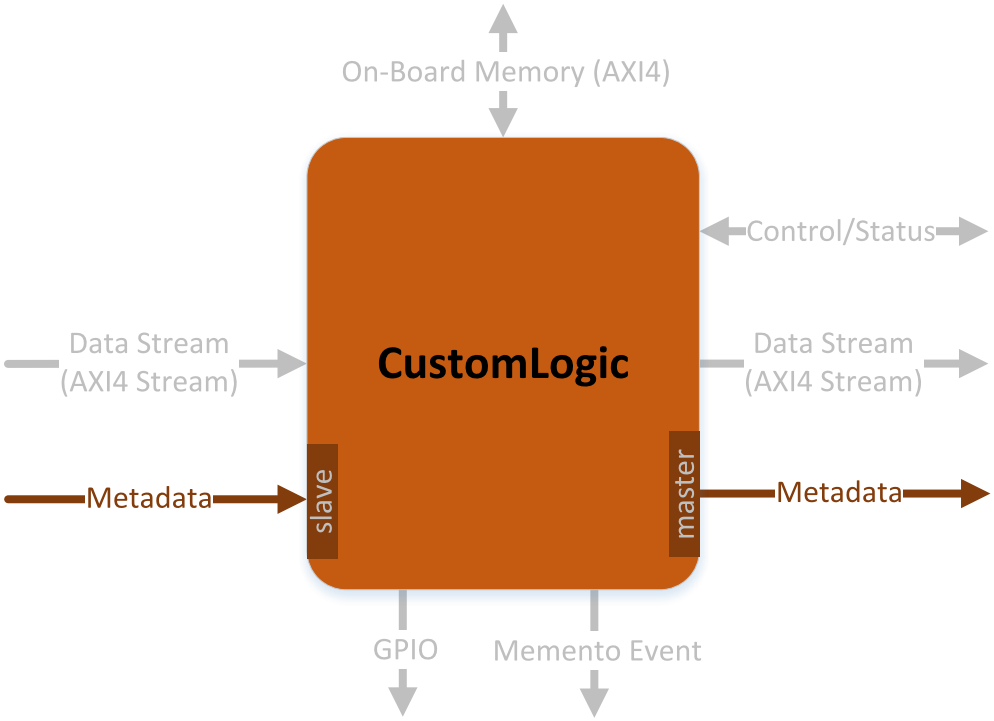
TVALID/TREADY 핸드 셰이크 및 TUSER 플래그 타이밍 다이어그램

이 예제에서 *LinePitch*는 64 바이트(각각 16바이트의 전송 주기 4개)이며 전체 프레임은 10줄/패킷으로 구성됩니다.

AXI4-Stream 프로토콜에 대한 자세한 내용은 www.xilinx.com의 Xilinx "AXI Reference Guide (UG1037)"및 www.amba.com의 "AMBA AXI4-Stream 프로토콜 사양"을 참조하십시오.

2.3. 메타 데이터 인터페이스

CoaXPress 장치에서 생성된 CoaXPress Image Header는 Metadata 인터페이스에 표시됩니다. CoaXPress Image Header 외에도 Metadata 인터페이스는 픽셀 정렬, 타임 스탬프와 관련된 정보를 제공합니다.



인터페이스 신호

슬래이브 측에서 메타 데이터 인터페이스는접두사 *s*를 표시하고 방향 *Input*을 가집니다.

마스터 측에서 *접두사 m을 표시하고 방향 Output을 가집니다.*

메타 데이터 인터페이스에는 두 가지 신호 그룹이 있습니다.

- *CoaXPress Image Header* 그룹 - CoaXPress 장치에서 발행한 CoaXPress Rectangular Image Header의 사본을 포함하는 신호입니다.
- *CustomLogic* 그룹 - 타임 스탬프 및 데이터 스트림 특성을 알리는 신호.

CoaXPress Image Header 그룹

신호	폭	설명
(m/s)_mdata_StreamId	N*8	고유한 스트림 ID입니다.
(m/s)_mdata_StreamId	N*16	16비트 소스 이미지 인덱스. 전송된 각 이미지에 대해 증가하며 0xFFFF에서 0으로 감습니다. 동일한 번호는 동일한 이미지와 관련된 데이터를 포함하는 각 스트림에 의해 사용되어야 합니다.
(m/s)_mdata_Xsize	N*24	이미지 폭을 픽셀 단위로 나타내는 24비트 값.
(m/s)_mdata_Xoffs	N*24	전체 장치 이미지의 왼쪽 픽셀에 대한 이미지의 픽셀 단위 가로 오프셋을 나타내는 24비트 값입니다.
(m/s)_mdata_Ysize	N*24	이미지 높이를 픽셀 단위로 나타내는 24비트 값. 이 값은 라인 스캔 이미지의 경우 0으로 설정됩니다.
(m/s)_mdata_Yoff	N*24	전체 장치 이미지의 상단 라인에 대한 이미지의 픽셀 단위 수직 오프셋을 나타내는 24비트 값입니다.
(m/s)_mdata_DsizeL	N*24	이미지 라인 당 데이터 워드의 수를 나타내는 24비트 값.
(m/s)_mdata_PixelF	N*16	픽셀 포맷을 나타내는 16비트 값.
(m/s)_mdata_TapG	N*16	탭 형상을 나타내는 16비트 값.
(m/s)_mdata_Flags	N*8	이미지 플래그.

노트
너비 열에서: "*N*"은 *CustomLogic* 변형에서 지원하는 장치/카메라 수인 총 인터페이스 슬롯 수를 나타냅니다.

- **3602 Coaxlink Octo** (1-카메라, 맞춤형 로직) => *N* = 1;
- **3603 Coaxlink Quad CXP-12** (1-카메라, 맞춤형 로직) => *N* = 1;
- **3603 Coaxlink Quad CXP-12** (4-카메라, 맞춤형 로직) => *N* = 4;

참고설명은 CoaXPress 표준 버전 2.0에서 발췌한 것입니다.

CustomLogic 그룹

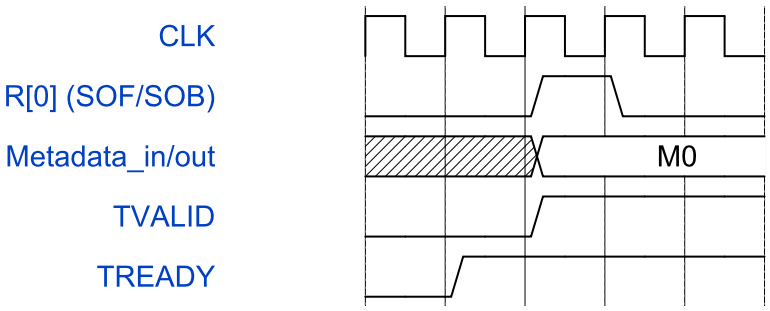
신호	폭	설명
(m/s)_mdata_Timestamp	N*32	장치의 읽기 시작 이벤트의 타임 스탬프입니다.
(m/s)_mdata_PixProcFlgs	N*8	Pixel processing flags: □ PixProcFlgs[0] => RGB에서 BGR 로의 스왑이 활성화되었습니다. □ PixProcFlgs[1] => MSB 픽셀 정렬이 활성화되었습니다. □ PixProcFlgs[2] => 포장된 수집이 가능합니다. □ PixProcFlgs[7:4] => LUT 설정.
(m/s)_mdata_Status	N*32	*CustomLogic*에서 상태를 보고하는 데 사용할 수 있는 32비트 벡터입니다.

참고슬레이브 측의 상태 값은 0x00000000입니다.

노트
너비 열에서: "*N*"은 *CustomLogic* 변형에서 지원하는 장치/카메라 수인 총 인터페이스 슬롯 수를 나타냅니다.

- **3602 Coaxlink Octo** (1-카메라, 맞춤형 로직) => *N* = 1;
- **3603 Coaxlink Quad CXP-12** (1-카메라, 맞춤형 로직) => *N* = 1;
- **3603 Coaxlink Quad CXP-12** (4-카메라, 맞춤형 로직) => *N* = 4;

타이밍 다이어그램

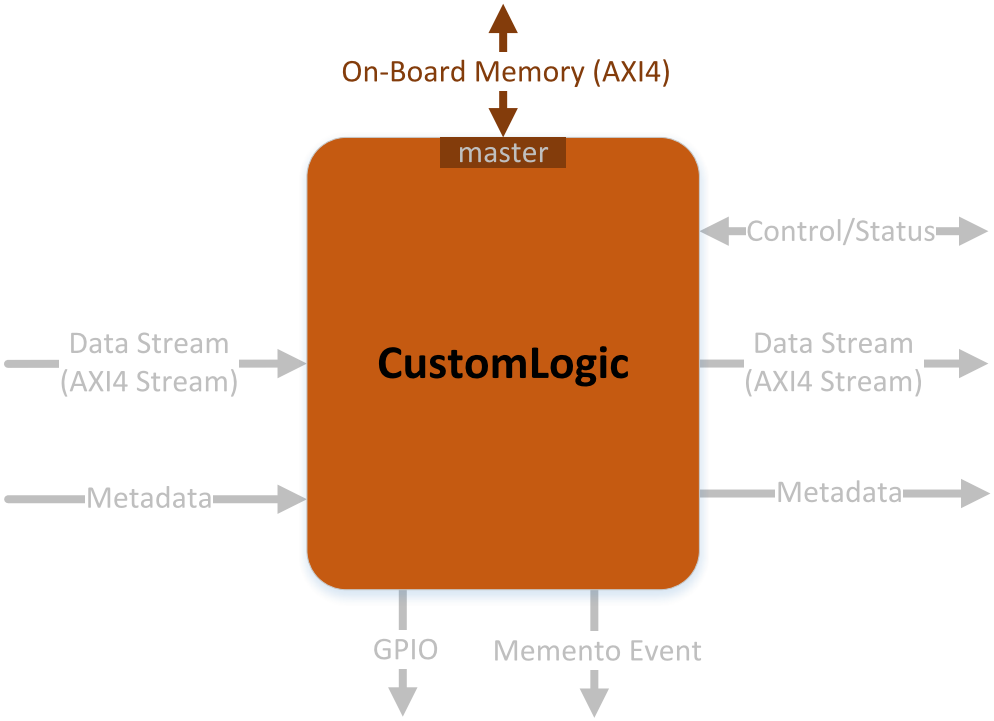


메타 데이터 타이밍 다이어그램

메타 데이터 신호는 *TUSER* 플래그 SOF/SOB가 발휘될 때마다 업데이트됩니다.

2.4. 온보드 메모리 인터페이스.

온보드 메모리 인터페이스는 AMBA AXI4-Stream 프로토콜 사양을 기반으로 합니다.

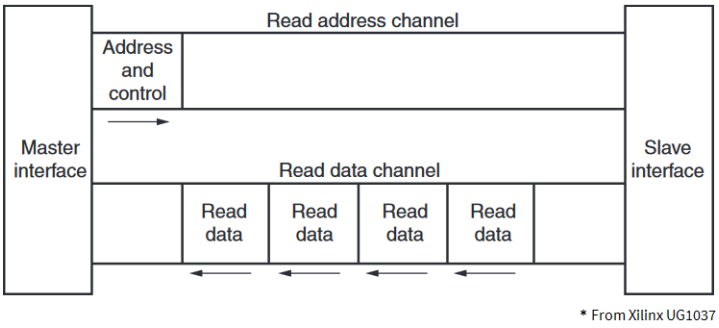


AXI4는 5개의 채널로 구성된 메모리 매핑 인터페이스입니다.

- 주소 채널 쓰기
- 데이터 채널 쓰기
- 응답 채널 쓰기
- 주소 채널 읽기
- 데이터 채널 읽기

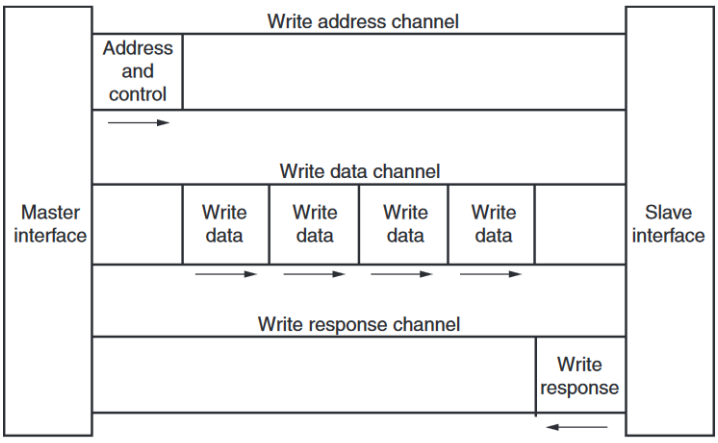
데이터는 마스터와 슬레이브 간에 동시에 양방향으로 이동할 수 있으며 데이터 전송 크기가 다를 수 있습니다. AXI4의 한계는 최대 256 데이터 전송의 버스트 트랜잭션입니다.

AXI4 채널 아키텍처



* From Xilinx UG1037

Channel Architecture of Reads



* From Xilinx UG1037

Channel Architecture of Writes

AXI4 신호 설명

다음 섹션에서는 AXI4 신호에 대해 간략하게 설명합니다.

> **참고**신호, 인터페이스 요구 사항 및 트랜잭션 속성에 대한 전체적인 내용은 www.amba.com에서 AMBA AXI 및 ACE Protocol Specification 문서를 참조하십시오.

마스터 인터페이스 글로벌 신호

신호	폭	지시문	설명
m_axi_resetn	1	입력	RESETn은 AXI4 인터페이스를 재설정합니다.

마스터 쓰기 주소 채널 인터페이스 신호

신호	폭	지시문	설명
m_axi_awaddr	32	출력	주소 쓰기. 주소 쓰기는 버스트 트랜잭션 쓰기에서 첫 번째 전송 주소를 제공합니다.
m_axi_awlen	8	출력	버스트 길이. 버스트 길이는 정확한 버스트 전송 횟수를 나타냅니다. 이 정보는 주소와 관련된 데이터 전송 수를 결정합니다. 버스트 길이 = AWLEN[7:0] + 1
m_axi_awsize	3	출력	버스트 크기. 이 신호는 버스트의 각 전송 크기를 나타냅니다. 버스트 크기 = 2^AWSIZE[2:0]
m_axi_awburst	2	출력	버스트 유형. 버스트 유형 및 크기 정보는 버스트 내의 각 전송에 대한 주소 계산 방법을 결정합니다. Burst_Type: "00" = FIXED; "01" = INCR; "10" = WRAP
m_axi_awlock	1	출력	잠금 유형. 전송의 원자적 특성에 대한 추가 정보를 제공합니다. Atomic_Access : '0'정상; '1'독점
m_axi_awcache	4	출력	메모리 유형. 이 신호는 트랜잭션이 시스템을 통해 진행되는 데 필요한 방식을 나타냅니다. Memory_Attributes: □ AWCACHE[0] 버퍼 가능 □ AWCACHE[1] 캐시 가능 □ AWCACHE[2] 읽기-할당 □ AWCACHE[3] 쓰기-할당
m_axi_awprot	3	출력	보호 유형. 이 신호는 트랜잭션의 특권 및 보안 레벨과 트랜잭션이 데이터 액세스인지 또는 명령 액세스인지 여부를 나타냅니다. Access_Permissions: □ AWPROT[0] 권한 있음 □ AWPROT[1] 보안되지 않음 □ AWPROT[2] 지침
m_axi_awqos	4	출력	서비스 품질, QoS. 각 쓰기 트랜잭션에 대해 전송된 QoS 식별자입니다. Quality_of_Service: 우선 순위 수준

신호	폭	지시문	설명
m_axi_awvalid	1	출력	올바른 주소 쓰기. 이 신호는 채널이 유효한 쓰기 주소 및 제어 정보를 신호하고 있음을 나타냅니다.
m_axi_awready	1	입력	주소 쓰기 준비. 이 신호는 슬레이브가 주소 및 관련 제어 신호를 받아들일 준비가되었음을 나타냅니다.

참고설명은 AMBA AXI 및 ACE 프로토콜 사양에서 발췌한 것입니다.

마스터 쓰기 데이터 채널 인터페이스 신호

신호	폭	지시문	설명
m_axi_wdata [*]	W	출력	데이터 쓰기.
m_axi_wstrb	W/8	출력	스트로브 쓰기. 이 신호는 유효한 데이터를 보유하고 있는 바이트 레인을 나타냅니다. 데이터 쓰기 버스의 각 8비트에 대해 하나의 스트로브 쓰기 비트가 있습니다.
m_axi_wlast	1	출력	마지막 쓰기. 이 신호는 버스트 쓰기의 마지막 전송을 나타냅니다.
m_axi_wvalid	1	출력	쓰기 준비. 이 신호는 슬레이브가 데이터 쓰기를 받아들일 수 있음을 나타냅니다.
m_axi_wready	1	입력	쓰기 준비. 이 신호는 슬레이브가 데이터 쓰기를 받아들일 수 있음을 나타냅니다.

노트
너비 열에서 : "*W*"는 쓰기 데이터 채널의 데이터 너비 인 MEMORY_DATA_WIDTH를 나타냅니다.

- **3602 Coaxlink Octo** (1-카메라, 맞춤형 로직) => *W* = 128 ;
- **3603 Coaxlink Quad CXP-12** (1-카메라, 맞춤형 로직) => *W* = 256;
- **3603 Coaxlink Quad CXP-12** (4-카메라, 맞춤형 로직) => *W* = 256;

참고설명은 AMBA AXI 및 ACE 프로토콜 사양에서 발췌한 것입니다.

마스터 쓰기 응답 채널 인터페이스 신호

신호	폭	지시문	설명
m_axi_bresp	2	입력	응답 쓰기. 이 신호는 트랜잭션 쓰기 상태를 나타냅니다. 응답: □ "00" = OKAY □ "01" = EXOKAY □ "10" = SLVERR □ "11" = DECERR
m_axi_bvalid	1	입력	올바른 응답 쓰기. 이 신호는 채널이 올바른 응답 쓰기를 신호하고 있음을 나타냅니다.
m_axi_bready	1	출력	응답 준비. 이 신호는 마스터가 쓰기 응답을 받아들일 수 있음을 나타냅니다.

참고설명은 AMBA AXI 및 ACE 프로토콜 사양에서 발췌한 것입니다.

의 경우m_axi_bresp

- *OKAY*: 정상적인 액세스 성공. 정상적인 액세스가 성공했음을 나타냅니다. 독점적인 액세스가 실패했음을 나타낼 수도 있습니다. 정상적인 액세스 성공을 확인합니다.
- *EXOKAY*: 독점 액세스 가능. 독점 액세스의 읽기 또는 쓰기 부분이 성공적으로 완료되었음을 나타냅니다.
- *SLVERR*: 슬레이브 오류. 액세스가 슬레이브에 성공적으로 도달했지만 슬레이브가 원래 마스터에 오류 조건을 반환하고자할 때 사용됩니다.
- *DECERR*: 디코드 오류. 트랜잭션 주소에 슬레이브가 없음을 나타내기 위해 일반적으로 상호 연결 구성 요소로 생성합니다.

마스터 읽기 주소 채널 인터페이스 신호

신호	폭	지시문	설명
m_axi_araddr	32	출력	주소 읽기. 주소 읽기는 버스트 트랜잭션 읽기에서 첫 번째 전송 주소를 제공합니다.
m_axi_arlen	8	출력	버스트 길이. 버스트 길이는 정확한 버스트 전송 횟수를 나타냅니다. 이 정보는 주소와 관련된 데이터 전송 수를 결정합니다. 버스트 길이 = AWLEN[7:0] + 1
m_axi_arsize	3	출력	버스트 크기. 이 신호는 버스트의 각 전송 크기를 나타냅니다. 버스트 크기 = 2^ARSIZE[2:0]
m_axi_arburst	2	출력	버스트 유형. 버스트 유형 및 크기 정보는 버스트 내의 각 전송에 대한 주소 계산 방법을 결정합니다. Burst_Type: "00" = FIXED; "01" = INCR; "10" = WRAP
m_axi_arlock	1	출력	잠금 유형. 전송의 원자적 특성에 대한 추가 정보를 제공합니다. Atomic_Access : '0'정상; '1'독점
m_axi_arcache	4	출력	메모리 유형. 이 신호는 트랜잭션이 시스템을 통해 진행되는 데 필요한 방식을 나타냅니다. Memory_Attributes: □ ARCACHE[0] 버퍼 가능 □ ARCACHE[1] 캐시 가능 □ ARCACHE[2] 읽기 할당 □ ARCACHE[3] 쓰기 할당
m_axi_arprot	3	출력	보호 유형. 이 신호는 트랜잭션의 특권 및 보안 레벨과 트랜잭션이 데이터 액세스인지 또는 명령 액세스인지 여부를 나타냅니다. Access_Permissions: □ ARPROT[0] 권한 있음 □ ARPROT[1] 보안되지 않음 □ ARPROT[2] 지침
m_axi_arqos	4	출력	서비스 품질, QoS. 각 쓰기 트랜잭션에 대해 전송된 QoS 식별자입니다. Quality_of_Service: 우선 순위 수준

신호	폭	지시문	설명
m_axi_arvalid	1	출력	올바른 주소 읽기. 이 신호는 채널이 올바른 읽기 주소 및 제어 정보를 신호하고 있음을 나타냅니다.
m_axi_arready	1	입력	주소 읽기 준비. 이 신호는 슬레이브가 주소 및 관련 제어 신호를 받아들일 준비가되었음을 나타냅니다.

참고설명은 AMBA AXI 및 ACE 프로토콜 사양에서 발췌한 것입니다.

마스터 읽기 데이터 채널 인터페이스 신호

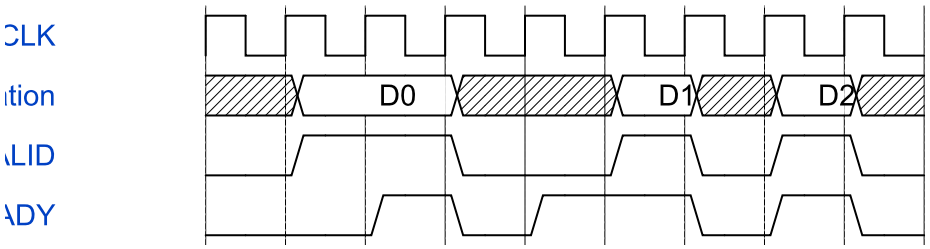
신호	폭	지시문	설명
m_axi_rdata	W	입력	읽기 데이터.
m_axi_rresp	2	입력	응답 읽기. 이 신호는 읽기 전송의 상태를 나타냅니다. 응답: "00" = OKAY; "01" = EXOKAY; "10" = SLVERR; “11” = DECERR
m_axi_rlast	1	입력	마지막 읽기. 이 신호는 버스트 쓰기의 마지막 전송을 나타냅니다.
m_axi_rvalid	1	입력	올바른 읽기. 이 신호는 채널이 필요한 읽기 데이터를 신호하고 있음을 나타냅니다.
m_axi_rready	1	출력	읽기 준비. 이 신호는 마스터가 읽기 데이터 및 응답 정보를 받아 들일 수 있음을 나타냅니다.

노트
너비 열에서 : “*W*"는 쓰기 데이터 채널의 데이터 너비 인 MEMORY_DATA_WIDTH를 나타냅니다.

- **3602 Coaxlink Octo** (1-카메라, 맞춤형 로직) => *W* = 128 ;
- **3603 Coaxlink Quad CXP-12** (1-카메라, 맞춤형 로직) => *W* = 256;
- **3603 Coaxlink Quad CXP-12** (4-카메라, 맞춤형 로직) => *W* = 256;

참고설명은 AMBA AXI 및 ACE 프로토콜 사양에서 발췌한 것입니다.

타이밍 다이어그램



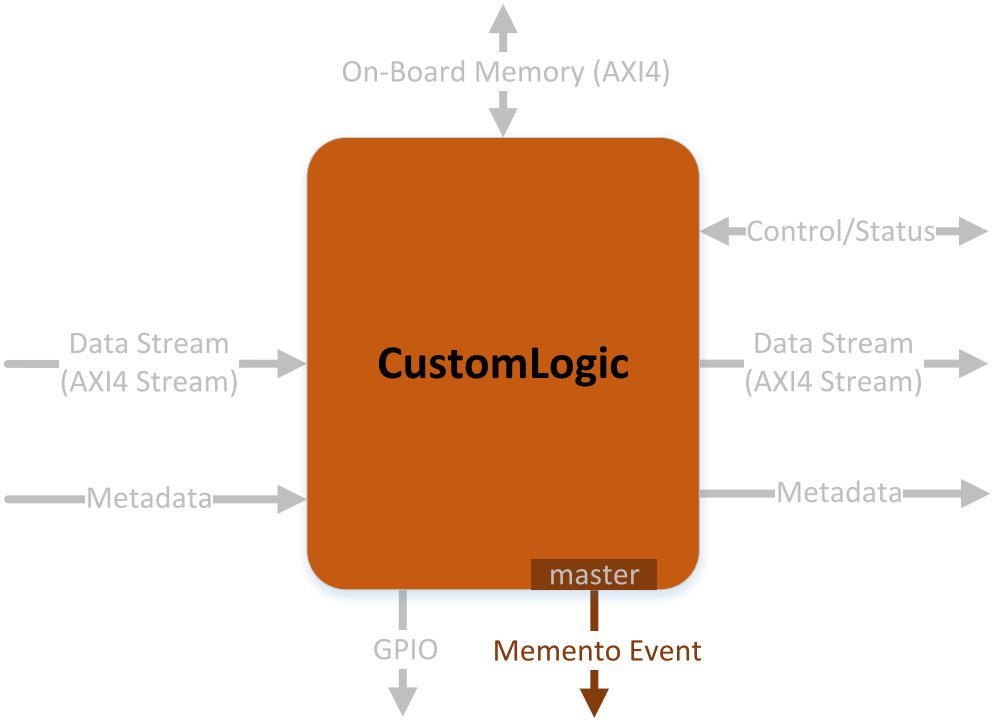
올바른/준비 핸드셰이크 타이밍 다이어그램

2.5. Memento 이벤트 인터페이스

Memento Event 인터페이스를 통해 *CustomLogic*은 1µs의 정밀도로 Memento Logging 도구에 타임 스탬프 처리된 이벤트를 보낼 수 있습니다.

타임스탬프 처리된 이벤트와 함께, 두 개의 32비트 인수가 다음과 같이 Memento에 보고됩니다.

```
ARG_0 ARG_1 [ts : 0195.350699] CustomLogic의 메시지: 0x00000003 0x00000002
```



마스터 인터페이스 신호

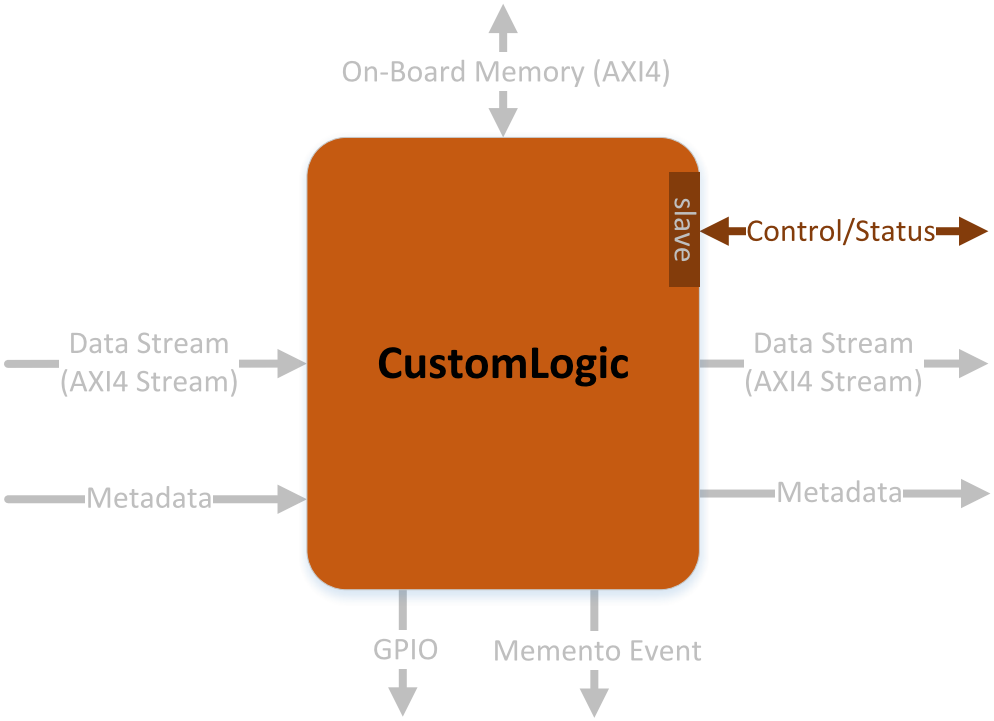
신호	폭	지시문	설명
m_memento_event	N*1	출력	*CustomLogic* 이벤트를 나타내는 한 주기의 펄스입니다.
m_memento_arg0	N*32	출력	Memento Logging 도구에서 해당 *CustomLogic* 이벤트와 함께 보고되는 32비트 인수.
m_memento_arg1	N*32	출력	Memento Logging 도구에서 해당 *CustomLogic* 이벤트와 함께 보고되는 32비트 인수.

노트
너비 열에서: “N"은 *CustomLogic* 변형에서 지원하는 장치/카메라 수인 총 인터페이스 슬롯 수를 나타냅니다.

- **3602 Coaxlink Octo** (1-카메라, 맞춤형 로직) => *N* = 1;
- **3603 Coaxlink Quad CXP-12** (1-카메라, 맞춤형 로직) => *N* = 1;
- **3603 Coaxlink Quad CXP-12** (4-카메라, 맞춤형 로직) => *N* = 4;

2.6. 제어/상태 인터페이스

Control/Status 인터페이스를 사용하면 Coaxlink 드라이버 API를 통해 *CustomLogic* 내부에서 레지스터를 읽거나 쓸 수 있습니다.



이 인터페이스의 사용은 *CustomLogic*이 Control/Status 인터페이스를 정의하는 방법에 따라 크게 달라집니다. 권장되는 정의는 참조 디자인 파일 `control_registers.vhd`에 설명된 대로이 인터페이스를 주소/데이터 제어 레지스터로 사용하는 것입니다.

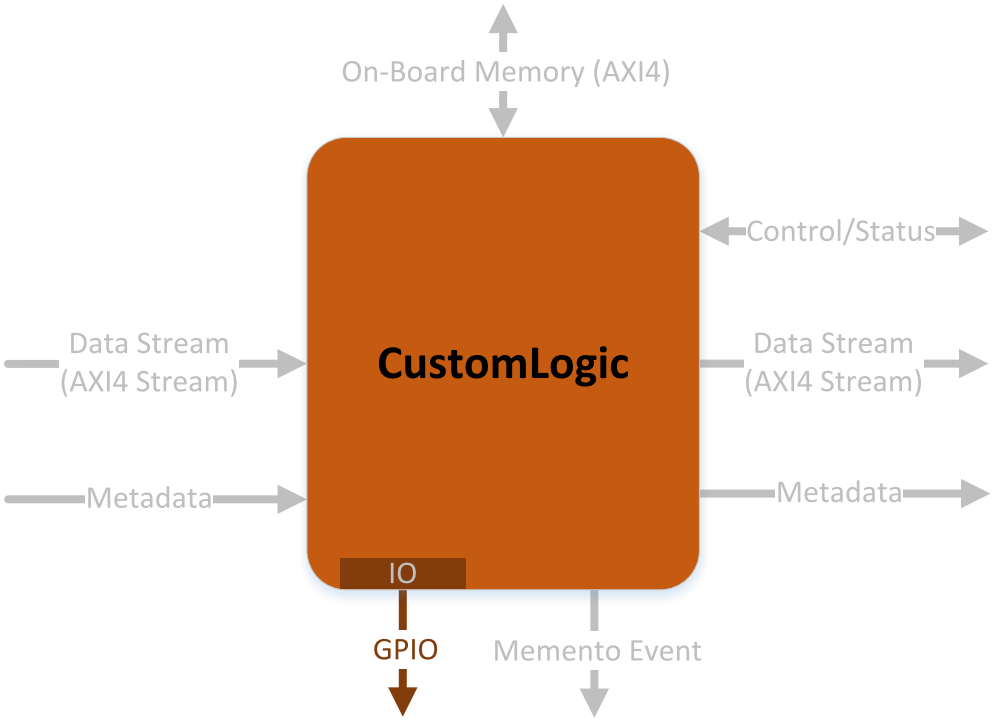
인터페이스 신호

신호	폭	지시문	설명
s_ctrl_addr	16	입력	16비트 WR/RD 주소. WR/RD 주소는 읽기/쓰기될 레지스터를 선택합니다.
s_ctrl_data_wr_en	1	입력	s_ctrl_data_wr에서 업데이트를 나타내는 한 주기의 펄스.
s_ctrl_data_wr	32	입력	32비트 쓰기 데이터. 선택한 레지스터에 32비트 벡터를 씁니다.
s_ctrl_data_rd	32	출력	32비트 읽기 데이터. 선택한 레지스터에서 32 비트 벡터를 복사합니다.

2.7. 범용 I/O 인터페이스

범용 I/O 인터페이스(GPIO) 인터페이스는 Coaxlink 보드에서 사용 가능한 모든 I/O의 상태에 대한 액세스를 제공합니다. 또한 사용자가 '사용자 출력 레지스터를 제어할 수 있습니다.

일반 목적 I/O 및 사용자 출력 레지스터에 대한 자세한 내용은 Coaxlink 기능 가이드 및 매뉴얼 Coaxlink 하드웨어를 참조하십시오.



인터페이스 신호

신호	폭	지시문	설명
user_output_ctrl	16	출력	사용자 출력 레지스터 제어: □ Ctrl[1:0] => UserOutput0 □ Ctrl[3:2] => UserOutput1 □ Ctrl[5:4] => UserOutput2 □ Ctrl[7:6] => UserOutput3 □ Ctrl[9:8] => UserOutput4 □ Ctrl[11:10] => UserOutput5 □ Ctrl[13:12] => UserOutput6 □ Ctrl[오후 3:14] => UserOutput7 각 UserOutput 레지스터는 독립적으로 제어할 수 있습니다. 다음과 같이 'Ctrl' 필드는인코딩됩니다: □ “01” => UserOutputx <= ‘1’ □ “10” => UserOutputx <= ‘0’ □ 기타 => 변경 없음 **참고**각 'Ctrl' 필드는 현장에서 변화가 감지될 때마다 한 번 평가됩니다. 그것은 해당 'Ctrl' 필드가 지속적으로 "01"로 강제하는 경우에도 UserOutput 상태가 Coaxlink 드라이버 API를 통해 변경될 수 있음을 의미합니다.
user_output_status	8	입력	사용자 출력 레지스터 상태: □ Status[0] => UserOutput0 □ Status[1] => UserOutput1 □ Status[2] => UserOutput2 □ Status[3] => UserOutput3 □ Status[4] => UserOutput4 □ Status[5] => UserOutput5 □ Status[6] => UserOutput6 □ Status[7] => UserOutput7
standard_io_set1_	10	입	표준 I/O 세트 #1 상태:

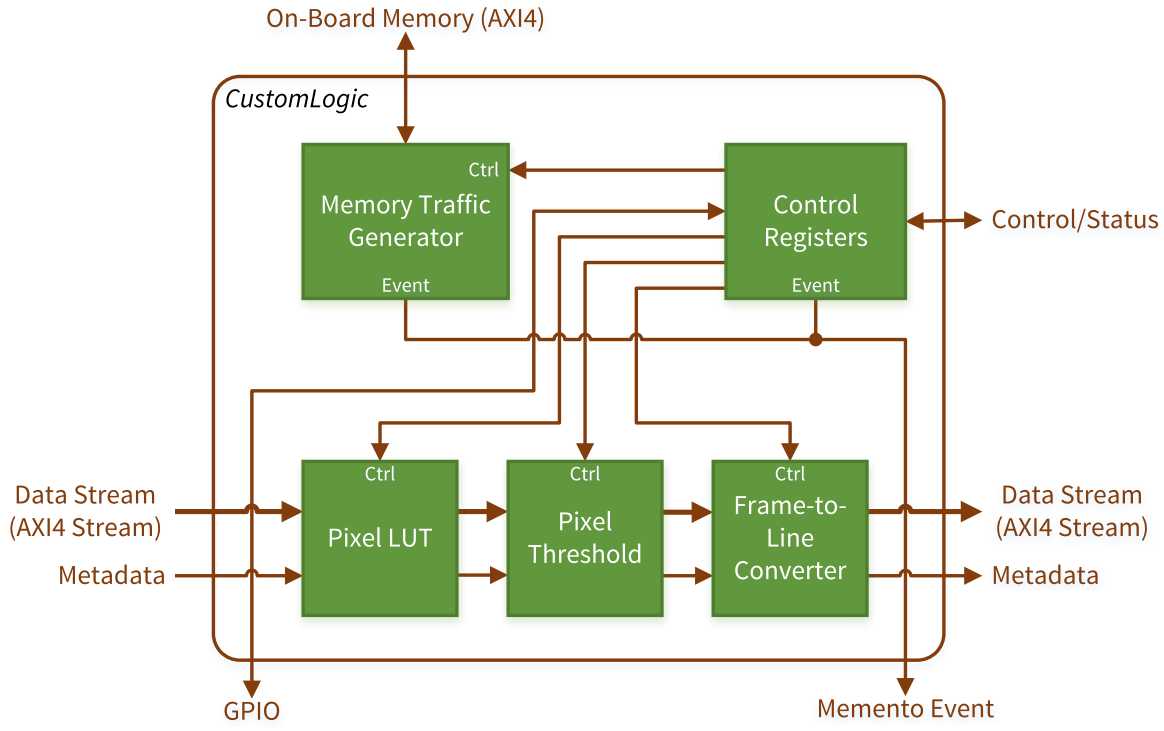
신호	폭	지시문	설명
status		력	□ Status[0] => DIN11 □ Status[1] => DIN12 □ Status[2] => IIN11 □ Status[3] => IIN12 □ Status[4] => IIN13 □ Status[5] => IIN14 □ Status[6] => IOUT11 □ Status[7] => IOUT12 □ Status[8] => TTLIO11 □ Status[9] => TTLIO12
standard_io_set2_status	10	입력	표준 I/O 세트 #2 상태: □ Status[0] => DIN21 □ Status[1] => DIN22 □ Status[2] => IIN21 □ Status[3] => IIN22 □ Status[4] => IIN23 □ Status[5] => IIN24 □ Status[6] => IOUT21 □ Status[7] => IOUT22 □ Status[8] => TTLIO21 □ Status[9] => TTLIO22
module_io_set_status	40	입력	I/O 확장 모듈 상태: □ Status[0] => MIO1 □ Status[1] => MIO2 □ … □ Status[39] => MIO40

3. CustomLogic 참조 디자인

3.1. 소개

CustomLogic 패키지는 *CustomLogic*의 템플릿으로 사용하기위한 참조 디자인과 함께 제공됩니다. 레퍼런스 디자인은 *CustomLogic*에서 사용할 수 있는 모든 인터페이스를 제공합니다.

기준 설계는 다음 블록 다이어그램과 함께 VHDL 파일 세트로 제공됩니다.

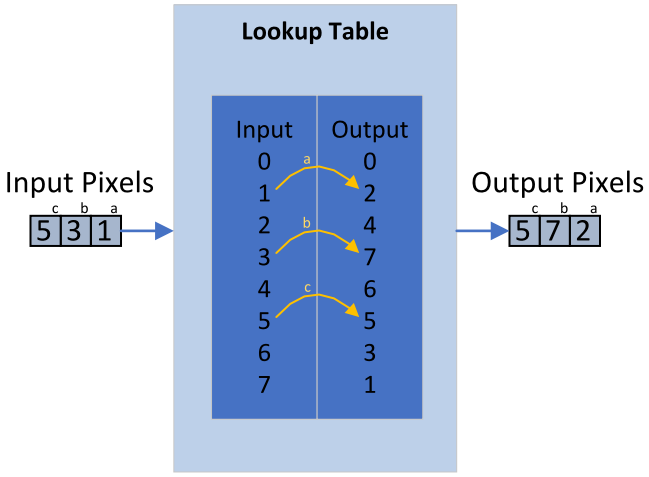


기준 설계 블록 다이어그램

3.2. 사용 가능한 참조 모듈

픽셀 LUT 8비트.

픽셀 LUT 8 비트는 *CustomLogic* 참조 디자인 파이프 라인에서 찾아보기 테이블 연산자를 제공합니다. 찾아보기 테이블 연산자는 테이블의 미리 정의된 값으로 입력 픽셀 값을 변경할 수 있습니다. 찾아보기 테이블(예: 감마 보정 및 대비 향상)에 대한 많은 응용 프로그램이 있습니다. 다음 그림은 찾아보기 테이블 연산자를 보여줍니다.



픽셀 LUT 8 비트는 클록주기 당 16 비트 (**3602 Coaxlink Octo** 의 경우) 또는 32비트 (**3603 Coaxlink Quad CXP-12** 의 경우) 8비트 픽셀을 계산할 수 있습니다. 제어 레지스터 모듈은 찾아보기 테이블 값을 제어하고 업로드하는 데 사용됩니다.

참고이 모듈은 Mono8 픽셀 포맷을 지원합니다.

픽셀 임계값

픽셀 임계값은 *CustomLogic* 참조 디자인 파이프 라인에서 임계값 연산자를 제공합니다. 각 입력 픽셀에 대해 임계값 연산자는 다음 공식에 따라 0 또는 255를 출력합니다.

$$OutputPixel = 255 \text{ when } InputPixel \geq Th;$$
$$OutputPixel = 0 \text{ when } InputPixel < Th;$$

여기서 는 Th 레벨입니다.

픽셀 임계값 계산은 클럭 주기 당 16비트 (**3602 Coaxlink Octo** 의 경우) 또는 32비트 (**3603 Coaxlink Quad CXP-12** 의 경우)의 8비트 픽셀을 계산합니다. 제어 레지스터 모듈은 픽셀 임계값 모듈을 제어하는 데 사용됩니다.

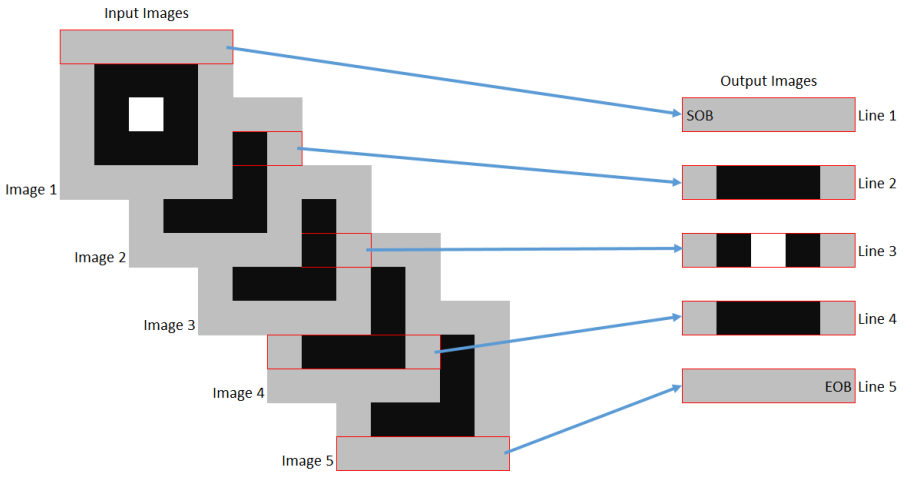
참고이 모듈은 C++ 코드를 입력으로 사용하여 Vivado HLS에서 생성하였습니다. 이 모듈을 다시 생성하려면 파일에 설명된 절차를 따르십시오./05_ref_design_hls/HLS_README.txt

참고이 모듈은 Mono8 픽셀 포맷을 지원합니다.

프레임-라인 변환기

프레임-라인 변환기는 각 입력 이미지에 대해 하나의 라인을 출력합니다. 출력한 라인은 다음과 같은 방식으로 입력 이미지에서 추출됩니다.

- 첫 번째 입력 이미지에서 첫 번째 라인을 추출합니다.
- 두 번째 입력 이미지에서 두 번째 라인을 추출하는 등의 작업을 수행합니다.
- 프레임-라인 변환기가 입력 이미지의 마지막 라인을 추출할 때 (입력 이미지의 수가 이미지 *Ysize*와 같음) 이 라인의 마지막 전송시에 *End-of-Buffer* 플래그를 활성화하고 새로운 획득 주기를 시작합니다.



프레임-라인 변환기는 첫 번째 이미지의 입력 메타 데이터(소스 측)를 시퀀스로 래치하고 이를 출력 메타 데이터(대상 측)로 전송합니다.

이 모듈은 제어 레지스터 참조 설계를 통해 제어할 수 있습니다.

메모리 트래픽 생성기

메모리 트래픽 생성기는 1024바이트의 데이터 버스트를 0x0000000에서 주소를 증가시키고 0x40000000(1GB)에 래핑합니다. 기록된 데이터는 8비트 카운터로 구성됩니다.

1024 바이트의 각 버스트 후에 메모리 트래픽 생성기는 동일한 주소의 데이터를 다시 읽습니다. 또한 발생한 주소 순환 횟수를 보고합니다.

이 모듈은 제어 레지스터 참조 설계를 통해 제어할 수 있습니다.

Memento 이벤트

참고 디자인에는 Memento 이벤트 두 가지 소스가 있습니다:

- 하나는 벡터를 정의할 수 있는 제어 레지스터를 통하는 것입니다.`m_memento_arg0`
- 다른 이벤트는 메모리 트래픽 생성기에서 주소 랩 어라운드가 발생할 때 생성됩니다. 이 경우 `m_memento_arg1`는 주소 랩 어라운드 카운터의 값을받습니다.

범용 I/O

범용 I/O 인터페이스에서 사용 가능한 모든 I/O의 상태는 제어 레지스터를 통해 읽을 수 있습니다. UserOutput 레지스터를 제어하고 제어 레지스터를 통해 상태를 다시 읽을 수도 있습니다.

제어 레지스터

제어 레지스터 모듈은 제어/상태 인터페이스를 통해 *CustomLogic*에 구현된 모듈을 제어/구성하는 메커니즘을 제공합니다. 참조 레지스터 맵은 다음과 같습니다.

레지스터	주소	설명
스크래치 패드	0x0000	비트 31:0(R/W) □ 32비트 스크래치 패드(재설정 값 => 0x00000000)
MemTrafficGen	0x0001	비트 0(R/W) □ '0'=> 메모리 트래픽 생성기가 비활성화된 경우(재설정 값) □ '1'=> 메모리 트래픽 생성기가 활성화된 경우
UserOutCtrl	0x0002	비트 1:0 (R/W) => UserOutput0 비트 3:2 (R/W) => UserOutput1 비트 5:4 (R/W) => UserOutput2 비트 7:6 (R/W) => UserOutput3 비트 9:8 (R/W) => UserOutput4 비트 11:10 (R/W) => UserOutput5 비트 13:12 (R/W) => UserOutput6 비트 15:14 (R/W) => UserOutput7 비트 필드 인코딩: □ “01” => UserOutputx <= ‘1’일 때 □ “10” => UserOutputx <= ‘0’일 때 □ 기타 => 변경 없음
UserOutStatus	0x0003	비트 0 (R) => UserOutput0 상태 비트 1 (R) => UserOutput1 상태 비트 2 (R) => UserOutput2 상태 비트 3 (R) => UserOutput3 상태 비트 4 (R) => UserOutput4 상태 비트 5 (R) => UserOutput5 상태 비트 6 (R) => UserOutput6 상태 비트 7 (R) => UserOutput7 상태
IoSet1Status	0x0004	비트 0 (R) => DIN11 상태 비트 1 (R) => DIN12 상태 비트 2 (R) => IIN11 상태 비트 3 (R) => IIN12 상태 비트 4 (R) => IIN13 상태 비트 5 (R) => IIN14 상태 비트 6 (R) => IOUT11 상태 비트 7 (R) => IOUT12 상태 비트 8 (R) => TTLIO11 상태 비트 9 (R) => TTLIO12 상태
IoSet2Status	0x0005	비트 0 (R) => DIN21 상태 비트 1 (R) => DIN22 상태 비트 2 (R) => IIN21 상태 비트 3 (R) => IIN22 상태 비트 4 (R) => IIN23 상태 비트 5 (R) => IIN24 상태 비트 6 (R) => IOUT21 상태 비트 7 (R) => IOUT22 상태 비트 8 (R) => TTLIO21 상태 비트 9 (R) => TTLIO22 상태

레지스터	주소	설명
MioSetAStatus	0x0006	비트 0 (R) => MIO1 상태 비트 1 (R) => MIO2 상태 … 비트 31 (R) => MIO32 상태
MioSetBStatus	0x0007	비트 0 (R) => MIO33 상태 비트 1 (R) => MIO34 상태 … 비트 7 (R) => MIO40 상태
Frame2Line	0x1n00	비트 0(R/W) □ “01” => 프레임-라인 변환기 바이 패스가 활성화된 경우(재설정 값) □ “10” => 프레임-라인 변환기 바이 패스가 비활성화된 경우
MementoEvent	0x1n01	비트 31:0(R/W) □ 이 레지스터에 쓰기를 수행하면 Memento 이벤트가 생성되고 여기에 정의된 32비트 벡터가 CustomLogic_event_arg0에 복사됩니다
PixelLut	0x1n02	비트 0(W, 자동 클리어) □ '1'=> 계수의 새로운 쓰기 시퀀스를 시작합니다 비트 4(R) □ '1'=> 계수의 쓰기 시퀀스의 끝을 나타냅니다 (재설정 값 => ‘0’) 비트 9:8(R/W) □ "01"=> 픽셀 LUT 바이 패스가 활성화된 경우(재설정 값) □ "10"=> 픽셀 LUT 바이 패스가 비활성화된 경우 □ 다른 경우 => 변경 없음
PixelLutCoef	0x1n03	Bits 7:0(W) □ 픽셀 LUT에 계수를 씁니다. 이 레지스터에 쓸 때마다 계수 인덱스가 0에서 255로 증가합니다.
PixelThreshold	0x1n04	비트 7:0(R/W) □ 0x00 => 변경 없음 □ 기타 => 픽셀 한계값 설정 (재설정 값 => 0x01) 비트 9:8(R/W) □ "01"=> 픽셀 임계값 바이 패스가 활성화된 경우(재설정 값) □ "10"=> 픽셀 임계값 바이 패스가 비활성화된 경우 □ 다른 경우 => 변경 없음

참고주소 열 "n"은 장치/카메라 채널의 선택기에 해당하는 4비트 필드입니다.

3.3. CustomLogic 배송

Coaxlink 패키지는 Vivado 2018.3을 대상으로 하며 다음 파일을 포함합니다:

- `<variant short-name>/01_readme`
  Coaxlink *CustomLogic* 패키지에서 Vivado 프로젝트를 생성하는 방법에 대한 간단한 설명.
- `<variant short-name>/02_coaxlink`
  CustomLogic 프레임워크를 빌드하는 데 필요한 독점적인 파일(암호화된 HDL, 넷리스트 및 TCL 스크립트) 모음입니다.
  참고: 이 파일은 수정해서는 안됩니다.
- `<variant short-name>/03_scripts`
  스크립트의 컬렉션 CustomLogic에서 개발 도움이 됩니다.
- `<variant short-name>/04_ref_design`
  레퍼런스 디자인 소스 파일입니다.
- `<variant short-name>/05_ref_design_hls`
  HLS 레퍼런스 디자인 소스 파일입니다.
- `<variant short-name>/06_release`
  레퍼런스 디자인 비트 스트림 파일을 사전 구축.

제품	변형 이름	변형 짧은 이름
3602 Coaxlink Octo	1- 카메라, 맞춤형 로직	CoaxlinkOcto_1-cam
3603 Coaxlink Quad CXP-12	1- 카메라, 맞춤형 로직	CoaxlinkQuadCxp12_1-cam
3603 Coaxlink Quad CXP-12	4- 카메라, 맞춤형 로직	CoaxlinkQuadCxp12_1-cam

3.4. 참조 디자인 빌드 절차

참조 디자인을 작성하려면:

1. Vivado 요구 사항을 준수하는 폴더에서 패키지의 압축을 푸십시오(경로에 특수 문자가 없음). 예: `c:/workspace/CustomLogic`
2. Vivado 시작
3. Tcl 콘솔에서 "create_vivado_project.tcl" 스크립트를 실행하십시오.
   TCL 명령: `source c:/workspace/CustomLogic/03_scripts/create_vivado_project.tcl`
   As result, a Vivado project is created at the folder `07_vivado_project`. For example: `c:/workspace/CustomLogic/07_vivado_project`.
4. 구현 실행.
   TCL 명령: `launch_runs impl_1`
5. Tcl Console에서 "customlogic_functions.tcl"스크립트를 실행하십시오.
   TCL 명령: `source c:/workspace/CustomLogic/03_scripts/customlogic_functions.tcl`
   이 스크립트는 다음 두 가지 기능을 사용할 수 있게 합니다:
   - *customlogic_bitgen*: .bit 파일을 생성합니다.
   - *customlogic_prog_fpga*: JTAG(volatile:휘발성)을 통해 FPGA를 프로그램합니다.

> **참고**이 기능을 사용하려면 Xilinx JTAG 프로그래머가 필요합니다.

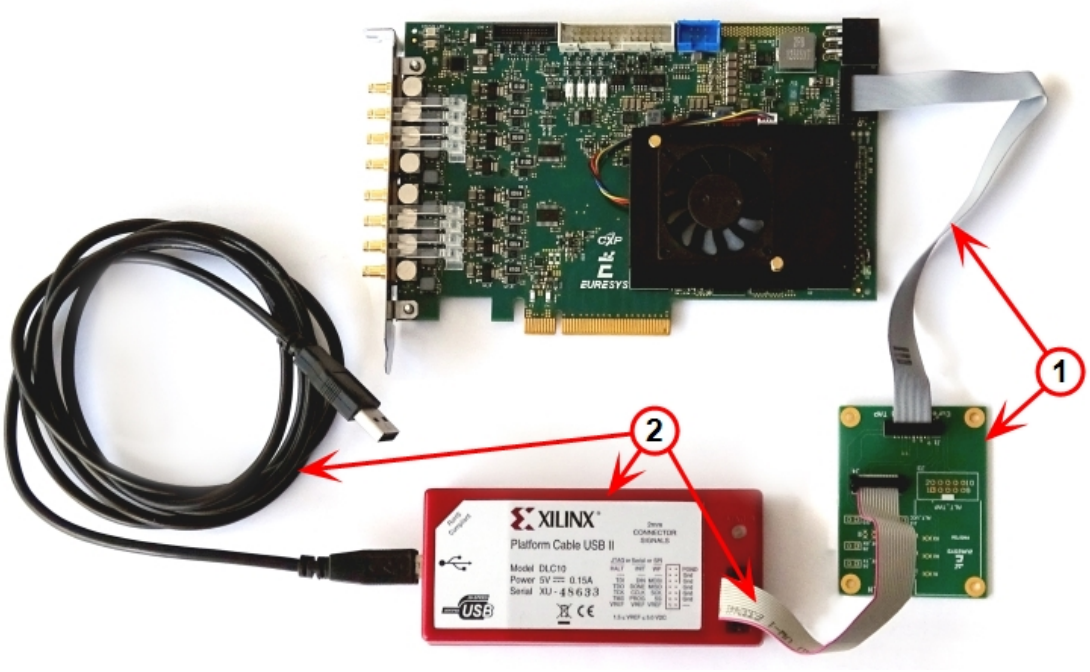
6. 구현이 완료되면 TCL 콘솔에서 "customlogic_bitgen"함수를 실행합니다.
   TCL 명령: `customlogic_bitgen`
   This function updates the bitstream file in the folder `06_release`
7. 비트 스트림이 생성된 후, `customlogic_prog_fpga` in the TCL console.
   TCL command: `customlogic_prog_fpga` 함수를 실행하여 FPGA를 업데이트합니다.

> **참고**이 단계는 선택 사항입니다.

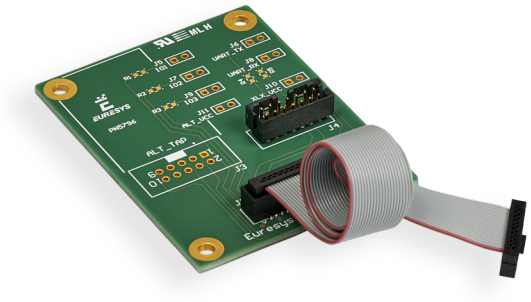
4. Debugging

*CustomLogic*은 FPGA를 프로그래밍하기 위해 추가 하드웨어가 필요하지 않습니다.

그러나 Vivado (ChipScope)의 디버깅 기능을 사용하려면 *3613 JTAG Adapter Xilinx for Coaxlink* (1)를 구입하여 *Xilinx 플랫폼 케이블 USB II 프로그래머* (2)를 Coaxlink FPGA에 연결하십시오.



Xilinx 플랫폼 케이블 USB II 프로그래머 (2)는 3613 JTAG Adapter Xilinx for Coaxlink (1)를 통해 3602 Coaxlink Octo 에 연결됩니다.



3613 JTAG Adapter Xilinx for Coaxlink

5. 시뮬레이션 테스트벤치

CustomLogic은 모든 CustomLogic 인터페이스를 자극할 수 있는 시뮬레이션 테스트벤치와 함께 제공됩니다. 또한, 데이터 스트림, 메타 데이터 및 Memento 이벤트 인터페이스로부터 백엔드 측에서 데이터를 캡처합니다. 결과(캡쳐된 데이터)는 폴더에 확장자가 '.dat'인 파일에 저장됩니다: `<variant short-name>/07_vivado_project/CustomLogic.sim/sim_1/behave/xsim`

테스트벤치는 스크립트 'create_vivado_project.tcl'로 만든 Vivado 프로젝트에 통합되었습니다. 시뮬레이션을 시작하려면 Vivado의 Tcl 콘솔에 `create_vivado_project.tcl` 명령을 입력하십시오.

'SimulationCtrl_tb.vhd'파일을 사용하면 테스트벤치를 제어할 수 있습니다. 이 파일에는 다음 예제와 같이 일련의 명령을 통해 테스트벤치에 대한 일련의 조치를 작성할 수있는 'SimulationCtrl_tb.vhd'라는 프로세스가 있습니다.

```
Simulation : process
Begin
	-- Enable Data Stream at channel 0
	EnableDataStream		(clk,status,ctrl, 0);

	-- Request 5 frames (256x10 Mono8) at channel 0.
	FrameRequest			(clk,status,ctrl, 0, 5, 256, 10, Mono8);

	-- Disable Data Stream at channel 0
	DisableDataStream		(clk,status,ctrl, 0);

	-- End simulation
	std.env.finish;
end process;
```

사용 가능한 모든 명령에 대한 설명은 폴더에 있는'SimulationCtrl_tb.vhd'파일에서 찾을 수 있습니다: SimulationCtrl_tb.vhd

참고'온 - 보드 메모리 인터페이스', 테스트벤치 모델의 저장 크기에 관한 것은 2MB로 제한됩니다.