(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년07월06일
(11) 등록번호 10-1754850
(24) 등록일자 2017년06월30일

(51) 국제특허분류(Int. Cl.)
G06F 9/52 (2006.01) *G06F 15/167* (2006.01)
G06F 9/48 (2006.01) *G06T 1/20* (2006.01)
(52) CPC특허분류
G06F 9/52 (2013.01)
G06F 15/167 (2013.01)
(21) 출원번호 10-2015-7011883
(22) 출원일자(국제) 2013년10월28일
심사청구일자 2015년05월06일
(85) 번역문제출일자 2015년05월06일
(65) 공개번호 10-2015-0067316
(43) 공개일자 2015년06월17일
(86) 국제출원번호 PCT/US2013/067076
(87) 국제공개번호 WO 2014/088726
국제공개일자 2014년06월12일
(30) 우선권주장
13/707,930 2012년12월07일 미국(US)
(56) 선행기술조사문헌
US5920714 A*
US20030033489 A1*
US20100115249 A1*
US20060136640 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
인텔 코포레이션
미합중국 캘리포니아 95054 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
날루리 헤마 찬드
인도 하이데라바드 500039 업팔 사이나가르 콜로니 8-68
나베일 아디티아
미국 캘리포니아주 95630 폴솜 위드제온 코트 1129
(74) 대리인
제일특허법인

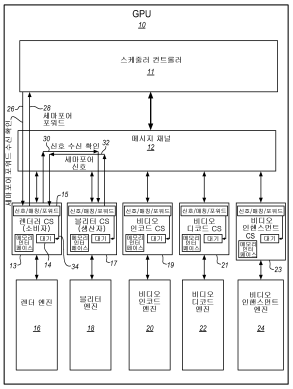
전체 청구항 수 : 총 30 항 심사관 : 김경완

(54) 발명의 명칭 **메모리 기반의 세마포어**

(57) 요 약

상이한 프로세싱 엔진 사이에서 동작을 동기화시키기에 유용한 메모리 기반 세마포어가 서술된다. 한 예에서, 동작은 생산자 엔진에서 콘텍스트를 실행하는 단계-실행하는 단계는 메모리 레지스터를 갱신하는 단계를 포함함-를 포함하고, 생산자 엔진으로부터 소비자 엔진에 메모리 레지스터가 갱신되었다는 신호를 보내는 단계-신호는 레지스터를 갱신하기 위해 소비자 엔진에 의해 실행되는 콘텍스트를 식별하기 위한 콘텍스트 ID를 포함함-를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 9/48 (2013.01)
G06T 1/20 (2013.01)

명 세 서

청구범위

청구항 1

생산자 엔진과 소비자 엔진 사이에서 콘텍스트를 동기화하기 위한 방법으로서,

생산자 엔진에서 콘텍스트를 실행하는 단계-상기 실행하는 단계는 메모리 레지스터를 갱신하는 단계를 포함함-와,

상기 생산자 엔진으로부터 소비자 엔진으로 상기 메모리 레지스터가 갱신되었다는 신호 메시지를 보내는 단계-상기 신호 메시지는 상기 레지스터를 갱신하기 위해 상기 소비자 엔진에 의해 실행되는 콘텍스트를 식별하기 위한 콘텍스트 ID(a Context ID)를 포함함-와,

상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에서 현재 처리되고 있는 콘텍스트의 콘텍스트 ID에 매칭되는지 여부를 상기 소비자 엔진에서 판정하는 단계와,

상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 콘텍스트에 매칭되는 경우에 상기 소비자 엔진에서 대기하는 단계를 포함하는

방법.

청구항 2

제 1 항에 있어서,

상기 신호 메시지는 상기 갱신된 메모리 레지스터 인라인 데이터(memory register inline data)를 식별하기 위한 메모리 레지스터 어드레스를 포함하는

방법.

청구항 3

제 1 항에 있어서,

생산자 엔진으로부터 소비자 엔진에 보내지는 상기 신호 메시지를 생성하는 단계-상기 신호 메시지는 상기 소비자 엔진 및 상기 소비자 엔진에 의해 상기 신호 메시지가 인가되는 콘텍스트 ID를 식별함-를 더 포함하는

방법.

청구항 4

제 3 항에 있어서,

스케줄러에서 상기 콘텍스트 ID를 생성하고 상기 소비자 엔진에 상기 콘텍스트 ID를 배치하는 단계-상기 콘텍스트 ID는 상기 소비자 엔진을 식별하기 위한 엔진 ID를 포함함-를 더 포함하는

방법.

청구항 5

제 1 항에 있어서,

상기 소비자 엔진에서 상기 신호 메시지를 수신하고 상기 생산자 엔진에 대하여 상기 소비자 엔진에 의해 상기

신호 메시지의 수신 확인을 하는 단계를 더 포함하는

방법.

청구항 6

제 1 항에 있어서,

상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되지 않는 경우, 상기 신호 메시지를 스케줄러에 포워딩하는 단계를 더 포함하는

방법.

청구항 7

제 6 항에 있어서,

상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되지 않는 경우, 상기 신호 메시지의 상기 콘텍스트 ID에 의해 식별되는 콘텍스트를 상기 스케줄러에서 다시 스케줄링하는 단계를 더 포함하는

방법.

청구항 8

제 1 항에 있어서,

기다리는 단계는 타이머를 기다리는 단계를 포함하는

방법.

청구항 9

제 1 항에 있어서,

상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 콘텍스트에 매칭되는 경우, 상기 소비자 엔진에서 상기 콘텍스트를 실행함에 있어서 사용하기 위해 상기 메모리 레지스터로부터 데이터를 인출하는 단계를 더 포함하는

방법.

청구항 10

제 9 항에 있어서,

상기 소비자 엔진에서 대기 힌트를 수신하는 단계와,

상기 대기 힌트에 응답하여 세마포어 메모리 어드레스로부터 데이터를 인출하는 단계와,

상기 대기 힌트 내의 인라인 세마포어 데이터와 상기 인출된 데이터를 비교하는 단계를 더 포함하는

방법.

청구항 11

제 9 항에 있어서,

상기 신호 메시지에 응답하여 파이프라인을 플러싱(flushing)하는 단계와,

상기 매칭하는 콘텍스트 ID에 대응하는 콘텍스트를 실행하는 단계를 더 포함하는

방법.

청구항 12

제 1 항에 있어서,

상기 콘텍스트를 실행하기 이전에 스케쥴러로부터 사용가능하게 될 크레딧을 대기하는 단계와, 상기 사용가능한 크레딧을 크레딧 카운트에 추가하는 단계와, 상기 신호 메시지를 보내는 단계 이후에 상기 크레딧 카운트를 감소시키는 단계를 더 포함하는

방법.

청구항 13

인스트럭션을 갖는 비일시적 머신 판독 가능 매체로서,

상기 인스트럭션은 머신에 의해 실행될 때에 상기 머신으로 하여금,

생산자 엔진에서 콘텍스트를 실행하는 것-상기 실행하는 것은 메모리 레지스터를 갱신하는 것을 포함함-과,

상기 생산자 엔진으로부터 상기 메모리 레지스터가 갱신되었다는 신호메시지를 소비자 엔진에 보내는 것-상기 신호 메시지는 상기 레지스터를 갱신하기 위해 상기 소비자 엔진에 의해 실행되는 콘텍스트를 식별하기 위한 콘텍스트 ID를 포함함-과,

상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에서 현재 처리되고 있는 콘텍스트의 콘텍스트 ID에 매칭되는지 여부를 상기 소비자 엔진에서 판정하는 것과,

상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 콘텍스트에 매칭되는 경우에 상기 소비자 엔진에서 대기하는 것을 포함하는 동작을 수행하게 하는

머신 판독 가능 매체.

청구항 14

제 13 항에 있어서,

상기 인스트럭션은 생산자 엔진으로부터 소비자 엔진에 보내지는 상기 신호메시지를 생성하는 것-상기 신호 메시지는 상기 소비자 엔진 및 상기 소비자 엔진에 의해 상기 신호 메시지가 인가되는 콘텍스트 ID를 식별함-을 더 포함하는

머신 판독 가능 매체.

청구항 15

제 14 항에 있어서,

상기 인스트럭션은 스케줄러에서 상기 콘텍스트 ID를 생성하고 상기 소비자 엔진에 상기 콘텍스트 ID를 배치하는 것-상기 콘텍스트 ID는 상기 소비자 엔진을 식별하기 위한 엔진 ID를 포함함-을 더 포함하는

머신 판독 가능 매체.

청구항 16

생산자 엔진과 소비자 엔진 사이에서 콘텍스트를 동기화하기 위한 장치로서,

메모리 레지스터를 갱신하는 것을 포함하는 콘텍스트를 실행하고, 상기 메모리 레지스터가 갱신되었다는 신호 메시지를 소비자 엔진에 보내는 생산자 엔진-상기 신호 메시지는 상기 레지스터를 갱신하기 위해 상기 소비자 엔진에 의해 실행되는 콘텍스트를 식별하기 위한 콘텍스트 ID를 포함함-과,

상기 신호 메시지를 수신하고, 상기 생산자 엔진에 대하여 상기 신호 메시지의 수신 확인을 수행하며,

상기 신호 메시지의 상기 콘텍스트 ID가 현재 처리되고 있는 콘텍스트의 콘텍스트 ID에 매칭되는지 여부를 판정하고, 상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 콘텍스트에 매칭되는 경우에 대기하도록 구성된 소비자 엔진을 포함하는

장치.

청구항 17

제 16 항에 있어서,

상기 소비자 엔진은 상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되는 경우에 상기 소비자 엔진에서 상기 콘텍스트를 실행함에 있어서 사용하기 위해 상기 메모리 레지스터로부터 데이터를 인출하는

장치.

청구항 18

생산자 엔진과 소비자 엔진 사이에서 콘텍스트를 동기화하기 위한 방법으로서,

상기 소비자 엔진에서 상기 생산자 엔진으로부터 콘텍스트 ID를 갖는 신호 커맨드를 수신하는 단계와,

상기 소비자 엔진에서 상기 콘텍스트 ID에 대응하는 콘텍스트의 커맨드 스트림을 실행하는 단계-상기 커맨드 스트림은 대기 커맨드를 포함하며, 상기 대기 커맨드는 세마포어 메모리 어드레스, 인라인 세마포어 데이터 및 비교 연산자를 포함함-와,

상기 생산자 엔진으로 상기 대기 커맨드를 수신 확인하는 단계와,

세마포어 메모리 어드레스로부터 데이터를 인출하고 상기 비교 연산자를 사용하여, 상기 인출된 데이터를 상기 인라인 세마포어 데이터와 비교함으로써 상기 소비자 엔진에서 상기 대기 커맨드를 실행하는 단계와,

상기 비교 연산자와의 비교가 충족되는 경우에 상기 콘텍스트의 상기 커맨드 스트림에 있어서의 다음 인스트럭션을 실행하는 단계를 포함하는

방법.

청구항 19

제 18 항에 있어서,

타이머가 만료되는 것을 기다리고, 기다린 후 상기 비교가 충족되지 않는 경우 데이터 인출 및 비교를 반복하는 단계를 더 포함하는

방법.

청구항 20

제 19 항에 있어서,

기다리는 단계는 생산자 엔진으로부터 세마포어 신호를 수신하기를 기다리는 단계를 포함하는

방법.

청구항 21

제 18 항에 있어서,

상기 비교가 충족되지 않는 경우에 상기 콘텍스트를 전환하는 단계를 더 포함하는

방법.

청구항 22

제 18 항에 있어서,

상기 비교가 충족되지 않는 경우에 스케줄러에 알리는 단계를 더 포함하는

방법.

청구항 23

제 22 항에 있어서,

콘텍스트 스케줄러에서 상기 대기 커맨드를 갱신하는 단계를 더 포함하는

방법.

청구항 24

제 23 항에 있어서,

상기 소비자 엔진의 상기 콘텍스트를 전환하기 전에 스케줄러에서 대기 조건을 재평가하는 단계를 더 포함하는

방법.

청구항 25

인스트럭션을 갖는 비일시적 머신 판독 가능 매체로서,

상기 인스트럭션은 머신에 의해 실행될 때에 상기 머신으로 하여금,

소비자 엔진에서 생산자 엔진으로부터 콘텍스트 ID를 갖는 신호 커맨드를 수신하는 것과,

상기 소비자 엔진에서 상기 콘텍스트 ID에 대응하는 콘텍스트의 커맨드 스트림을 실행하는 것-상기 커맨드 스트림은 대기 커맨드를 포함하며, 상기 대기 커맨드는 세마포어 메모리 어드레스, 인라인 세마포어 데이터 및 비교 연산자를 포함함-과,

상기 생산자 엔진으로 상기 대기 커맨드를 수신 확인하는 것과,

상기 세마포어 메모리 어드레스로부터 데이터를 인출하고 상기 비교 연산자를 사용하여, 상기 인출된 데이터를 상기 인라인 세마포어 데이터와 비교함으로써 상기 소비자 엔진에서 상기 대기 커맨드를 실행하는 것과,

상기 비교 연산자와의 상기 비교가 충족되는 경우에 상기 콘텍스트의 커맨드 스트림에 있어서의 다음 인스트럭션을 실행하는 것

를 포함하는 동작을 수행하게 하는

머신 판독 가능 매체.

청구항 26

제 25 항에 있어서,

상기 인스트럭션은 상기 비교가 충족되지 않는 경우에 스케줄러에 알리는 것을 더 포함하는

머신 판독 가능 매체.

청구항 27

제 26 항에 있어서,

상기 인스트럭션은 콘텍스트 스케줄러에서 상기 대기 커맨드를 갱신하는 것을 더 포함하는

머신 판독 가능 매체.

청구항 28

생산자 엔진과 소비자 엔진 사이에서 콘텍스트를 동기화하기 위한 장치로서,

어드레스를 갖는 메모리와,

콘텍스트를 위한 커맨드 스트림을 갖는 커맨드 스트리머와,

생산자 엔진으로부터 콘텍스트 ID를 갖는 신호 커맨드를 수신하고, 상기 콘텍스트 ID에 대응하는 커맨드 스트림을 실행하기 위한 소비자 엔진-상기 커맨드 스트림은 대기 커맨드를 포함하며, 상기 대기 커맨드는 상기 메모리로의 세마포어 메모리 어드레스, 인라인 세마포어 데이터 및 비교 연산자를 포함함-을 포함하되,

상기 소비자 엔진은 상기 생산자 엔진으로 대기 커맨드를 수신 확인하고, 상기 세마포어 메모리 어드레스에서 상기 메모리로부터 데이터를 인출하고 상기 인출된 데이터를 상기 인라인 세마포어 데이터와 비교함으로써 상기 대기 커맨드를 실행하고,

상기 소비자 엔진은 또한 상기 비교 연산자의 상기 비교가 충족되는 경우에 상기 커맨드 스트림에 있어서의 다음 인스트럭션을 실행하는

장치.

청구항 29

제 28 항에 있어서,

상기 비교가 충족되지 않는 경우에 상기 대기 커맨드를 갱신하기 위해 상기 소비자 엔진에 접속되는 스케줄러를 더 포함하는

장치.

청구항 30

제 29 항에 있어서,

상기 스케줄러는 또한 상기 소비자 엔진의 상기 콘텍스트를 전환하기 전에 대기 조건을 재평가하는

장치.

발명의 설명

기 술 분 야

[0001] 본 개시는 상이한 프로세싱 엔진 사이의 동작을 동기화하는 분야, 구체적으로는, 세마포어(semaphore) 및 메모리 레지스터를 사용하는 시그널링에 관련된다.

배 경 기 술

[0002] 컴퓨팅 기술은 범용 동작이 GPU(graphic processing unit)에서 행해질 수 있도록 하기 위해 개발되었다. GPU는 그래픽 프로세싱에 최적화되는 다수의 단순한 병렬 프로세싱 파이프라인을 갖는다. 많은 유사한 또는 동일한 병렬 계산을 필요로 하는 범용 동작을 GPU에 이동시킴으로써, 이들 동작은 CPU(Central Processing Unit)에서 행해지는 것보다 신속하게 행해질 수 있는 한편 CPU에서의 프로세싱 요구는 감소된다. 이것은 성능을 향상시키면서 전력 소비를 감소시킬 수 있다.

[0003] GPU는 상이한 기능을 행하기 위해 최적화되는 다수의 상이한 프로세싱 엔진을 갖는다. 이들 엔진은 특히 블리터 엔진(Blitter Engine), 렌더 엔진(Render Engine), 비디오 디코드 엔진(Video Decode Engine), 비디오 인코드 엔진(Video Encode Engine), 및 비디오 인핸스먼트 엔진(Video Enhancement Engine)을 포함할 수 있다. 각 엔진 처리는 독립된 스케줄링 프로세서에 의해 스케줄링되는 콘텍스트 내에서 인스트럭션한다. 스케줄링 프로세서는 콘텍스트를 각 엔진에 배치하고 각 콘텍스트와 관련된 커맨드 스트림의 실행을 관리한다.

[0004] 하지만, GPU의 프로세싱 엔진, 커맨드 버퍼 및 커맨드 스트리머는 상이한 엔진 사이의 중간값 및 커맨드의 이송을 조정하여야 한다. 한 엔진이 다른 엔진에 의해 실행되는 커맨드에서 소비될 값을 생성하고 있을 때, 해당 값이 소비자가 사용할 준비가 되어 있음을 보장하기 위해 몇몇의 메커니즘이 사용되어야 한다. 엔진 사이의 조정은 상당한 리소스를 소비할 수 있고 따라서 해당 리소스는 커맨드를 실행하기 위해 사용될 수 없다.

도면의 간단한 설명

[0005] 본 발명의 실시 형태는, 한정을 위해서가 아닌 예시의 목적으로, 첨부한 그림의 도면에 도시되고, 동일한 참조 번호는 유사한 요소를 가리킨다.

도 1은 본 발명의 실시 형태에 따른 커맨드 스트리머 및 신호를 갖는 그래픽 프로세싱 유닛의 일부분의 블록도이다.

도 2a는 본 발명의 실시 형태에 따른 예측 인에이블 비트를 사용하여 배치 버퍼(batch buffer)를 실행하는 처리 흐름도이다.

도 2b는 본 발명의 실시 형태에 따른 조건 지정 레지스터(predicate register)에 있어서의 값을 리프레시하는 처리 흐름도이다.

도 3은 본 발명의 실시 형태에 따른 예측 인에이블 비트 레지스터를 갖는 산술 논리 유닛의 하드웨어 논리도이다.

도 4는 본 발명의 실시 형태에서의 사용에 적합한 그래픽 프로세싱 유닛의 일부분의 블록도이다.

도 5는 본 발명의 실시 형태에서의 사용에 적합한 컴퓨터 시스템의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0006] 본 발명은 예컨대 그래픽 렌더링 회로에서의 메모리 액세스의 생산자/소비자 모델에 있어서의 엔진 사이에서 동기화하기 위해 메모리 기반 시그널링을 사용하는 것에 관한 것이다. 메모리 기반 세마포어는 다수의 엔진 및 호스트 CPU(Central Processing Unit)에서 실행되는 다수의 콘텍스트 사이의 통신을 위한 소프트웨어 스케줄링 모드에서 레지스터 기반 세마포어 대신에 사용될 수 있다. 다수의 엔진은 특히 블리터 엔진, 렌더 엔진, 비디오 디코드 엔진, 비디오 인코드 엔진, 및 비디오 인핸스먼트 엔진을 포함할 수 있다.

[0007] 세마포어는 데이터 소비자와 데이터 생산자 사이에서 상이한 타입의 정보를 시그널링하기 위해 사용될 수 있다. 세마포어는 어드레스, 파라미터 및 값을 포함할 수 있다. 세마포어 커맨드는 세마포어를 사용하기 위해 사용될 수 있다. 생산자/소비자 모델에서, 세마포어 데이터는 메모리에 저장될 수 있다. 메모리 액세스 대기 시간을 줄이기 위해, 메모리로부터 세마포어 데이터를 언제 또한 어떻게 샘플링하는지 결정하기 위해 현명한 방법이 사용될 수 있다.

[0008] 세마포어의 사용은 세마포어 실패에 대한 콘텍스트 전환 정책을 포함함으로써 향상될 수 있다. 프로그램 가능성 인터페이스가 또한 소프트웨어에 제공될 수 있다. 소프트웨어 지원에 더하여, 세마포어 실패로 인해 전환되는 콘텍스트의 효과적인 리스케줄링을 위한 결정을 행하기 위해 하드웨어 지원이 스케줄러 소프트웨어에 제공될 수 있다.

[0009] 성공하지 못한 세마포어 대기에 있어서 콘텍스트가 전환되는 때, 세마포어 대기 커맨드는 PPHWSP(per process hardware status page) 또는 유사한 메모리 공간에서 갱신될 수 있다. PPHWSP는 특정한 콘텍스트에 배치되는 그래픽 메모리에 있어서의 스크래치 공간이다. 각 콘텍스트가 그 자신의 PPHWSP를 가질 수 있거나 또는 스크래치 공간이 2개 이상의 콘텍스트에 의해 공유될 수 있다. 이 스크래치 공간은 하드웨어와 소프트웨어 사이의 통신의 수단으로서 사용될 수 있고 또한 하드웨어 및 소프트웨어에 의한 일시적 저장을 위해 사용될 수 있다. 세마포어 대기로 인한 전환된 콘텍스트를 리스케줄링하기 전에, 스케줄러는 하드웨어에 대기 조건을 제출하기 전에 대기 조건을 재평가하기 위해 PPHWSP를 판독할 수 있다. 이것은 스케줄러로 하여금 하드웨어 사용을 개선하고 재제출 중에 불필요한 콘텍스트 전환을 방지하게 한다.

[0010] 이하에 보다 상세하게 서술되는 바와 같이, 기억 장소에 있는 데이터는 동기화하기 위해 콘텍스트 사이의 통신을 위한 메일 박스로서 사용될 수 있다. 본 명세서에서 말하는 동기화는 제 2 콘텍스트가 실행을 시작하기 전에 제 1 콘텍스트가 제 1 콘텍스트의 의존(dependency)을 클리어하기를 기다리는 제 2 콘텍스트를 수반할 수 있다. 예컨대 전형적인 생산자/소비자 모델에서, 소비자는 소비자가 진행을 시작하기 전에 생산자가 완료를 시그널링하기를 기다린다. 이 동기화는 예컨대 이하에 서술되는 MI_SEMAPHORE_SIGNAL 커맨드 및 MI_SEMAPHORE_WAIT 커맨드를 사용하여 달성될 수 있다.

[0011] 이하의 예는 하드웨어 리소스에서의 동작을 관리하기 위해 콘텍스트를 사용한다. 각 콘텍스트에는 고유한 ID(Identification)가 배치될 수 있고 그 고유한 ID에 의해 콘텍스트는 그 수명 동안 식별된다. 콘텍스트는 또한 콘텍스트가 사용할 하드웨어 리소스인 엔진에 배치되고 그 엔진에서 콘텍스트는 실행될 것이다. 배치는 콘텍스트의 선두에서의 작업 타입 또는 콘텍스트의 임의의 고유한 요건에 근거할 수 있다. 콘텍스트 ID 및 배치는 일단 배치된 후에는 둘 중 어느 것도 변경되지 않도록 고정될 수 있다.

[0012] 도 1은 그래픽 프로세싱 유닛(GPU)(10)의 일반화된 블록도이다. 이 예에 있어서의 GPU는 렌더 엔진(16), 블리터 엔진(18), 비디오 인코드 엔진(20), 비디오 디코드 엔진(22), 및 비디오 인핸스먼트 엔진(24)의 5개의 엔진을 포함한다. 각 엔진은 그 자신의 커맨드 스트리머(CS)(13, 17, 19, 21, 23)에 접속된다. 엔진은 또한 메모리 및 다른 리소스(도시하지 않음)로의 다른 접속을 가질 수 있다. 각 커맨드 스트리머는 대기 모듈(14), 신호 인터페이스(15), 및 메모리 인터페이스를 포함한다. 메모리 인터페이스는 도 5의 메모리(525)와 같은 내부 또는 외부 공유 메모리에 접속한다. 세마포어 시그널링을 위한 통신은 이들 커맨드 스트리머 사이에서 이루어진다. 커맨드 스트리머는 모두 메시지 채널(12)에 연결되고 메시지 채널은 스케줄러 컨트롤러(SHIM)(11)에 연결된다. 스케줄러 컨트롤러는 본 명세서에서 대안적으로 스케줄러, 컨트롤러, 마이크로프로세서, 및 SHIM으로 불린다. 스케줄러 컨트롤러는 특정한 구현에 의존하여 이들 형태 또는 다른 형태 중 임의의 하나를 취할 수 있다.

[0013] 도시한 바와 같이, 세마포어 신호(32)는 생산자, 도시된 예에서의 블리터 엔진으로부터 그 블리터 CS(17)를 통해서 소비자 엔진, 도시된 예에서의 렌더 엔진에 보내진다. 그러한 신호는 임의의 엔진으로부터 임의의 엔진으

로 올 수 있지만, 이 예에서는 블리터 엔진으로부터 오는 것으로 나타내어진다. 신호는 메시지 채널(12)에 의해 커맨드 스트림 사이에서 반송된다. 소비자 엔진(16)은 그 커맨드 스트리머를 통해서 신호 수신 확인(signal acknowledgement)으로 대답한다.

[0014] 그러면 렌더 엔진은 세마포어 신호에 있어서의 콘텍스트 ID를 그 현재의 콘텍스의 콘텍스트 ID와 비교한다. 매칭되는 경우, 대기 엔진(14)이 기동되고, 대기 엔진이 세마포어 대기를 기다리고 있는 경우, 대기 후에, 대기 엔진은 생산자에 의해 제공된 데이터를 다시 샘플링하고 조건을 재평가할 것이다.

[0015] 통신 채널(34)은 "Context Match"로서 지정되어 Sig/Match/Fwd 블록(15)과 대기 fub(14) 사이에 제공된다. 생산자 CS, 여기에서의 블리터 엔진으로부터 신호가 수신되는 때, 소비자 CS, 여기에서의 렌더 엔진은 현재의 콘텍스트 id로 수신되는 콘텍스트 ID를 매칭하고 신호를 대기 fub(14)에 포워딩한다(15). 대기 fub가 대기 커맨드에 붙는 경우, 다른 평가를 위한 대기 기간 후에 메모리 인터페이스를 통해서 메모리 값을 다시 획득한다.

[0016] 대기 fub(14, 17, 19, 21, 23)는 이하에 보다 상세하게 서술되는 MI_SEAMPHORE_WAIT 커맨드를 실행하는 로직이다. 그것은 메모리 콘텐츠를 판독하고 조건을 평가한다. 이후, 폴 모드 또는 신호 모드에 근거하여, 대기 fub는 조건이 만족될 때까지 커맨드에서 기억 장소를 다시 샘플링한다. 도 5는 기억 장소에 액세스 가능한 메모리 인터페이스(515)를 나타낸다.

[0017] 콘텍스트 ID가 매칭되지 않을 때, 세마포어 포워드 신호(28)는 SHIM에 보내지고, SHIM은 세마포어 포워드 수신 확인으로 대답한다. 이것은 리소스가 렌더 엔진에 할당되게 한다. 이것은 스케줄러로 하여금 커맨드 스케줄러로부터 수신되는 콘텍스트 ID가 세마포어 대기로 인해 중단된 기존 콘텍스트에 매칭되는지 보게 한다. 커맨드 스케줄러는 다음 기회에 다시 제출될 준비가 되도록 중단된 콘텍스트를 이동시킬 것이다.

[0018] 콘텍스트 ID가 매칭되지 않을 때, 세마포어 포워드 신호(28)는 SHIM에 보내진다. SHIM은 세마포어 포워드 수신 확인으로 대답한다. 이것은 리소스가 렌더 엔진에 할당되게 한다. 본 명세서에서, 생산자 및 소스는 교환되어 사용되고 소비자 및 타겟은 교환되어 사용된다. 본 명세서에서 두 용어 사이에 임의의 차이가 있는 한 양쪽에 동일하게 적용된다.

[0019] 도 2a는 소스 엔진(102), 타겟 엔진(104) 및 마이크로컨트롤러(106)에서 행해지는 동작을 나타내는 처리 흐름도이다. 생산자의 예인 소스 엔진에서, 처리는 112에서 세마포어 시그널링을 사용하는 신호 프로세싱으로 시작된다. 114에서, 소스 엔진은 크레딧이 그 콘텍스트에 의해 사용 가능한 것을 기다린다. 크레딧이 사용 가능하게 될 때, 소스 엔진은 콘텍스트 ID 및 엔진 배치를 수신할 것이다. 116에서 이들은 타겟 엔진(104)에 보내진다. 그 후 118에서 소스 엔진은, 116에서의 콘텍스트를 위해 사용된 크레딧을 갖는 그 크레딧 카운트를 감소시키고 120에서 소스 엔진 세마포어 신호 처리는 종료된다.

[0020] 상술한 바와 같이, 세마포어 시그널링은 생산자-콘텍스트로 하여금 메모리에 있어서의 그 세마포어 중 하나가 갱신되었음을 명시함으로써 소비자-콘텍스트에 알리게 한다. 이것은 메모리에서 세마포어를 갱신하는 인스트럭션 이후의 생산자의 콘텍스트 커맨드 시퀀스 내에서, 이하에 보다 상세하게 서술되는 MI_SEMAPHORE_SIGNAL 커맨드를 프로그래밍함으로써 이루어질 수 있다. MI_SEMPHPHORE_SIGNAL 커맨드는 소비자 콘텍스트 ID 및 콘텍스트가 실행될 것인 엔진의 배치의 상세를 반송한다.

[0021] 생산자로서의, 콘텍스트 내에서 동작하고 있는 임의의 엔진은 MI_SEMAPHORE_SIGNAL 커맨드를 실행할 때에 소비자로의 신호 메시지를 생성할 것이다. 생산자는 커맨드에서 언급되는 엔진으로의 신호 메시지를 생성한다. 신호는 소비자의 콘텍스트 ID를 포함할 것이다. 이것은 세마포어 타겟 콘텍스트 ID가 타겟 엔진에 보내지는 116에서의 처리 흐름에 해당한다.

[0022] 소비자로서의, 신호 메시지를 수신하는 임의의 엔진은 메시지를 처리하고 시그널링 엔진에게 수신 확인한다. 세마포어 신호 메시지에서 수신되는 소비자 콘텍스트 ID는 엔진이 현재 실행하고 있는 처리를 위해 콘텍스트 ID와 비교된다. 매칭 및 미스매칭 결정은 이 비교에 근거하여 실시될 것이다. 엔진에서 실행되는 활성 콘텍스트가 없는 경우, 이것은 미스매칭으로서 여겨질 수 있다.

[0023] 도 2a에서, 122에서 타겟 세마포어 신호 프로세싱의 처리가 타겟 엔진(104)에서 시작된다. 124에서, 타겟 엔진은 실행하는 콘텍스트 ID가 소스 엔진으로부터 세마포어 신호로 수신된 콘텍스트 ID에 매칭되는지 여부를 결정하기 위해 비교를 행한다. 매칭이 있는 경우, 136에서 신호는 158에서의 세마포어 대기 로직에 보내진다. 이 신호는 구현에 따라서 타겟 세마포어 어드레스를 포함할 수 있거나 또는 포함하지 않을 수 있다. 어드레스는 예컨대 용장(redundant) 메모리 판독을 최적화하기 위해 사용될 수 있다. 대기 로직이 대기가 완료된 것을 나타낸 후에, 타겟 엔진은 콘텍스트에 있어서의 다음 인스트럭션을 실행한다. 이와 달리, 콘텍스트가 세마포어

대기를 기다리고 있지 않은 경우, 세마포어 신호는 세마포어 대기 fub에 의해 드롭될 수 있다. 세마포어 fub가 세마포어 대기를 기다리고 있는 경우에 착신 신호는 기억 장소를 다시 샘플링하고 대기 조건을 평가하도록 세마포어 대기 fub를 트리거할 것이다.

[0024] 콘텍스트 ID가 매칭되지 않는 경우, 신호는 포워딩된다. 세마포어 신호로 수신되는 소비자 콘텍스트 ID가 실행하는 콘텍스트를 위한 ID에 매칭되지 않을 때, 수신된 콘텍스트 ID는 스케줄러(106)에 메시징된다. 스케줄러는 데이터를 처리하고 그에 따라 보고된 콘텍스트를 다시 스케줄링한다. 이것은 타겟 엔진이 크레딧이 사용 가능하게 되는 것을 기다리는 126에서 나타내어진다. 크레딧은 리소스가 세마포어 시그널링에 이어지는 동작을 처리하기 위해 리소스가 사용 가능함을 보장하기 위해 사용된다. 크레딧이 사용 가능할 때, 128에서 매칭되지 않는 콘텍스트 ID와 함께 메시지가 스케줄러(106)에 보내진다.

[0025] 그 후 130에서 동작을 지원하기 위해 크레딧이 감소되고 132에서 수신 확인이 소스 엔진에 보내진다. 수신 확인을 보내면, 134에서 타겟 엔진에서의 세마포어 신호 프로세싱이 종료된다. 그러면 콘텍스트는 타겟 엔진에서 실행되고 임의의 결과가 메모리에 위치된다. 시스템에는 많은 엔진이 있을 수 있다. 결과적으로, 세마포어 시그널링을 사용하여 서로 메시징하는 다수의 생산자 및 소비자가 있을 수 있다. 도 2a의 처리 흐름에서, 이것은 엔진 사이의 메시징을 위한 크레딧 기반 흐름 메커니즘으로 관리된다.

[0026] 마이크로컨트롤러(106)는 크레딧 시스템을 사용하여 리소스 할당을 관리한다. 한 예에서, 마이크로프로세서는 마이크로컨트롤러와 다른 엔진 사이의 인터페이스이다. 이 하드웨어 요소는 다른 엔진과 마이크로컨트롤러 사이의 통신 및 그 반대의 통신을 가능하게 한다. 몇몇의 구현에서, 마이크로컨트롤러는 SHIM(106)으로 불린다. 마이크로컨트롤러는 콘텍스트 ID를 처리하고 요청에 수신 확인한다. 도 2b에 나타낸 바와 같이, 128에서 마이크로컨트롤러는 크레딧에 대한 요청을 콘텍스트 ID와 함께 수신한다. 마이크로컨트롤러는 크레딧을 제공(126)하기 위해 또는 크레딧을 수신(130)하기 위해 마이크로프로세서의 일부일 수 있거나 또는 마이크로프로세서로부터 분리될 수 있는 레지스터에서 크레딧(138)을 관리한다. 도시된 실시 형태는 크레딧을 사용하여 처리를 관리하는 맥락에서 나타내어지지만, 리소스 할당의 다른 형태가 대안으로서 사용될 수 있다.

[0027] 도 2b의 처리 흐름도에서 나타낸 바와 같은 세마포어 대기는 임의의 지정된 엔진에 의한 콘텍스트의 실행이 커맨드 스트림 실행 중에 정확한 시간에 중단되도록 한다. 엔진은 미리 배치된 기억 장소에서 요구되는 값이 갱신될 때까지 커맨드 스트림에 있어서의 정확한 위치로부터의 포워드 처리를 행할 수 없게 된다. 서술된 예에서, 이것은 실행된 커맨드 스트림에 MI_SEMAPHORE_WAIT 커맨드를 위치시킴으로써 달성된다. MI_SEMAPHORE_WAIT 커맨드는 세마포어 메모리 어드레스, 인라인 세마포어 데이터 및 비교 연산자를 반송한다. 커맨드 스트림에 있어서의 MI_SEMAPHORE_WAIT 커맨드를 실행하면, 엔진은 커맨드에 의해 나타내어지는 세마포어 메모리 어드레스로부터 데이터를 인출하고 해당 데이터를 인라인 세마포어 데이터와 비교한다.

[0028] 도 2b를 참조하면, 142에서 대기 프로세싱이 시작되고, 144에서 엔진은 인라인 데이터와의 비교를 위해 세마포어 어드레스로부터 데이터를 인출한다. 146에서, 데이터가 비교되고, 데이터가 매칭되는 경우, 148에서 대기 처리는 종료된다.

[0029] 146에서 비교가 통과하는 경우, 엔진은 대기 커맨드에 이어지는 다음 인스트럭션으로 이동한다. 한편, 비교가 실패하는 경우, 타겟 엔진은 어떻게 더 진행할지를 결정한다. 도 2b의 예에서, 150에서 대기 처리가 execlist (실행 리스트 : execution list) 모드에 있는지 우선 결정한다. 그러한 경우, 152에서, 콘텍스트 전환이 억제되는지 여부가 결정된다.

[0030] execlist 기반 스케줄링에서, 타겟 콘텍스트 ID 필드는 커맨드가 시그널링하고 있는 타겟 엔진의 콘텍스트에 대응하는 콘텍스트 ID를 포함한다. 신호 모드에 있어서의 MI_SEMPHORE_WAIT 커맨드를 기다리고 있는 타겟 엔진은 메모리로부터 데이터를 다시 인출하거나 또는 그 콘텍스트 ID가 이 시그널링된 콘텍스트 ID와 동일한지 비교할 것이다.

[0031] 150에서 Execlist가 인에이블되고 152에서 억제 콘텍스트 전환이 인에이블되지 않을 때, 154에서 타겟 엔진은 콘텍스트를 전환하고 콘텍스트 전환의 이유는 세마포어 대기 실패임을 스케줄러 프로세서 또는 SHIM(106)에 알린다. 콘텍스트 전환으로 이어지는 세마포어 대기에서, CS는 콘텍스트 이미지에 MI_SEMAPHORE_WAIT 커맨드의 상세를 삽입하여, 스케줄러는 콘텍스트를 다시 스케줄링하기 전에 조건을 재평가할 수 있다.

[0032] 동작의 다른 모드에서, 152에서 콘텍스트 전환이 억제되는 경우, 세마포어 실패에 따라 콘텍스트를 전환하는 대신에, 대기 엔진은 세마포어 조건이 만족되기를 기다린다. 진행하기 위해 대기 엔진은 세마포어 커맨드로부터의 조건이 만족될 때까지 메모리에 세마포어 데이터를 다시 획득한다. MI_SEMAPHORE_WAIT 커맨드에 있어서의

모드 비트는 세마포어 데이터가 메모리로부터 언제 다시 획득될 필요가 있는지 명시한다.

[0033] 도 2b는 블록 156에서 시작되는 폴링 기반 메커니즘을 나타낸다. 대기 엔진은 주기적으로 메모리에 세마포어 데이터를 획득하고(166) 비교를 평가한다(168). 이것은 조건이 만족될 때까지 계속된다. 세마포어 데이터의 샘플링의 164에서의 주기성은 설정될 수 있다.

[0034] 도 2b는 또한 158에서 신호 기반 메커니즘을 나타낸다. 대기 엔진은 세마포어 신호를 수신할 때마다 메모리에 세마포어 데이터를 획득한다. 이것은, 예컨대, 실행 중인 콘텍스트 ID가 신호 콘텍스트 ID에 매칭될 때 136에서 타겟 엔진(104)으로부터 유래할 수 있다. 146으로부터의 "아니오"의 경로인 세마포어 실패에 있어서, 엔진이 세마포어 실패에 직면한 것을 나타내는 세마포어 대기 인터럽트가 147에서 생성되고, 필요에 따라, 세마포어 대기는 취소될 수 있고 실행으로부터 콘텍스트를 선점할(preempt) 수 있다. 다시 말해서, 모드에 관계없이, 세마포어가 처음으로 실패할 때, 인터럽트가 스케줄러에 보내진다.

[0035] 또 다른 스케줄링 모드는 링 버퍼 모드이다. 스케줄링의 링 버퍼 모드에서, 타겟 엔진의 콘텍스트 ID 필드는 어떤 관련성도 갖지 않는다. 신호 모드에 있어서의 MI_SEMAPHORE_WAIT 커맨드를 기다리는 타겟 엔진은, 자신이 수신한 콘텍스트 ID에 관계없이, 신호를 수신하면 비교를 위해 메모리로부터 데이터를 인출할 것이다. MI_SEMAPHORE_WAIT 및 MI_SEMPHORE_SIGNAL 커맨드는 모두 링 버퍼 모드에서 execlist 모드와 동일한 방법으로 실행되고 타겟 엔진에 유사하게 영향을 미친다. 하지만 세마포어 대기의 실패는 execlist 모드에 있어서 콘텍스트 전환을 야기할 수 있는 한편 링 버퍼 모드에 있어서 콘텍스트 전환을 야기하지 않을 것이다.

[0036] 폴링 모드로 되돌아가면, MI_SEMAPHORE_SIGNAL 또는 유사한 커맨드에 대한 지원이 없을 수 있다. 도 2b에서, 150에서 대기 엔진이 execlist 모드에 있지 않은 경우, 156에서 대기 엔진이 폴링 모드에 있는지 결정된다. 그러한 경우, 하드웨어는 메모리를 주기적으로 샘플링하고 세마포어 조건을 재평가한다.

[0037] 이것은, 예컨대, 162에서 폴 타이머를 개시하는 것 및 164에서 타이머 상태를 감시하는 것으로서 나타내어진다. 타이머가 만료될 때, 160에서 대기 엔진은 미결정 execlist를 체크한다. 리스트가 발견되는 경우, 148에서 대기가 종료된다. 그렇지 않은 경우, 166에서 폴 타이머가 정지된다. 이 시점에서, 미결정 execlist가 없기 때문에, 대기 엔진은 세마포어 커맨드에 의해 제공되는 어드레스로부터 데이터를 인출하고 168에서 그것을 인라인 데이터와 비교한다. 168에서 매칭이 있는 경우, 170에서 아이들 상태 및 폴 타이머가 리셋되고 148에서 대기가 종료된다. 한편, 매칭이 없는 경우, 폴링 모드는 162에서 폴 타이머를 개시하기 위해 되돌아간다. 또 다른 타이머 사이클이 개시되고 폴링 모드는 다시 반복된다.

[0038] 신호 기반 모드에서, 도 2a의 136에서, 재평가 세마포어 대기 힌트가 대기 로직에 보내진다. 158에서 힌트를 수신하면, 콘텍스트 ID가 힌트 및 미결정 execlist로부터의 매칭으로부터 체크된다. 160에서 대기 처리 미결정이 있는 경우 및 미결정 execlist가 없는 경우, 166에서 세마포어가 메모리로부터 다시 취득되고 재평가된다. 세마포어가 통과하면(168), 148에서 대기 처리가 종료된다. 168에서 세마포어가 실패하면, 처리는 다음 신호 힌트를 기다리기 위해(158) 156으로 돌아간다.

[0039] 위에서 사용된 MI_SEMAPHORE_SIGNAL 커맨드는 타겟 엔진을 시그널링하고 타겟 콘텍스트 ID를 사용하여 타겟 엔진의 콘텍스트 중 하나에 메모리 세마포어 갱신 발생을 언급하기 위한 커맨드로서 정의될 수 있다. 실시 형태에서, MI_SEMPHORE_SIGNAL 커맨드 및 추가적인 MI_SEMAPHORE_WAIT 커맨드는 함께 메일박스 및 레지스터 기반 세마포어 시그널링을 대신할 수 있다. MI_ATOMIC(제시되지 않음) 커맨드는 메모리에 있어서의 세마포어 데이터를 갱신하기 위해 MI_SEMAPHORE_SIGNAL 커맨드보다 먼저 프로그래밍될 수 있다.

[0040] 몇몇의 경우에, 예컨대 커맨드의 비트에 포함되는 정보에 근거하여, MI_SEMAPHORE_SIGNAL 커맨드는 포스트 싱크(post sync) 동작으로서의 세마포어 신호와 함께 파이프라인된(pipelined) PIPE_CONTROL 플러시 커맨드로서 실행될 수 있다. 플러시 완료는 단지 이 커맨드에 앞선 작업은 윈도우어 유닛(windower unit)으로 밀려나가고 이 커맨드에 앞서 발행된 임의의 미처리 플러시가 완료되는 것을 보장한다. 신호는 포스트 싱크 동작으로서 절약될 수 있다. 커맨드 스트리머는 이 경우에 다른 커맨드의 실행을 계속할 수 있다. 포스트 싱크 동작으로서 절약되는 어토믹 동작(atomic operation)은 발행된 플러시 커맨드의 완료 후에 어떤 시점에 실행될 수 있다.

[0041] 이 포스트 싱크 동작 모드에서, 어토믹 세마포어 신호 동작은 링 버퍼 또는 배치 버퍼에 프로그래밍되는 MI 커맨드의 나머지에 대해서는 정상적이지 않을 것이다. 하지만, 어토믹 세마포어 신호 동작은 임의의 PIPE_CONTROL 커맨드로 인해 초래되는 포스트 싱크 동작에 대하여는 정상적일 것이다. MI_SEMAPHORE_SIGNAL 커맨드에 대한 예시적 구조가 표 1에 나타내어진다. 필드의 임의의 하나 이상은 다른 필드와 대체될 수 있다. 필드의 순서는 변경될 수 있고 보다 많은 또는 보다 적은 필드가 보다 많은 또는 보다 적은 비트로 사용될 수

있다.

DWord	비트	설명
0	31:29	커맨드 타입 디폴트 값 : 0h MI_COMMAND 포맷 : OpCode
0	28:23	MI 커맨드 Opcode 디폴트 값 : 1Bh MI_SEMAPHORE_SIGNAL 포맷 : OpCode
0	22	예비됨
0	21	포스트 싱크 동작 0h : 포스트 싱크 동작 없음 1h : 포스트 싱크 동작
0	20:18	예비됨
0	17:15	타겟 엔진 선택 SIGNAL이 보내질 타겟 엔진을 선택한다
0	14:8	예비됨
0	7:0	DWord 길이 디폴트 값 : 1h 포맷 : =n
1	31:0	타겟 콘텍스트 ID execlist 모드에서 이것은 이 커맨드가 시그널링하는 타겟 엔진을 위한 콘텍스트 ID를 포함한다

[0042] 표 1 MI_SEMAPHORE_SIGNAL

[0043] MI_SEMAPHORE_WAIT 커맨드는 사용되는 특정한 시스템 및 표준에 따라 여러 가지의 상이한 형태를 취할 수 있다. 한 예에서, 이 커맨드는 도 2b에서 예를 위해 나타낸 바와 같은 메모리 기반 세마포어 대기를 지원한다. 메모리 기반 세마포어는 생산자 및 소비자 콘텍스트 사이에서의 동기화를 위해 사용될 수 있다. 생산자 및 소비자 콘텍스트는 그래픽 프로세싱 시스템 내부의 상이한 엔진 또는 동일한 엔진에서 실행될 수 있다. 한 실시 형태에서, 두 콘텍스트는 양쪽에서 execlist가 인에이블될 때에 동일한 엔진에서 실행될 수 있다. 여기서 서술되는 커맨드로, 생산자 콘텍스트는 신호를 구현하고 소비자 콘텍스트는 대기를 구현한다.

[0044] 이 커맨드를 분석하면, 커맨드 스트리머는 이 커맨드에 언급되는 세마포어 어드레스로부터 데이터를 인출하고 그것을 인라인 세마포어 데이터 Dword와 비교한다. 비교가 통과하는 경우, 커맨드 스트리머는 다음 커맨드로 이동한다. 커맨드 스트리머는 세마포어 실패에 직면하면 항상 스케줄러로의 인터럽트를 생성한다.

[0045] execlist가 인에이블될 때, 비교가 실패하는 경우, 커맨드 스트리머는 콘텍스트를 전환한다. 콘텍스트 전환은, 예컨대, GFX_MODE 레지스터에 "억제 동기 콘텍스트 전환"을 설정함으로써 억제될 수 있다.

[0046] 스케줄링 또는 "억제 동기 콘텍스트 전환" 세트를 갖는 Execlist의 링 버퍼 모드에서, 비교가 실패하는 경우, 커맨드 스트리머는 비교 동작이 참(true)일 때까지 또는 대기가 소프트웨어에 의해 취소될 때까지 대기 모드에 근거하여 비교 동작을 평가한다.

[0047] MI_SEMAPHORE_WAIT 커맨드에 대한 예시적 구조가 표 2에 나타내어진다. 필드의 임의의 하나 이상은 다른 필드와 대체될 수 있다. 필드의 순서는 변경될 수 있고 보다 많은 또는 보다 적은 필드가 보다 많은 또는 보다 적은 비트로 사용될 수 있다. 비교 동작에서, SAD는 세마포어 어드레스 데이터(Semaphore Adress Data)에 대응하고 SDD는 세마포어 데이터 Dword(Semaphore Data Dword)에 대응한다.

DWord	비트	설명
0	31:29	커맨드 타입 디폴트 값 : 0h MI_COMMAND 포맷 : OpCode
0	28:23	MI 커맨드 Opcode 디폴트 값 : 1Ch MI_SEMAPHORE_WAIT 포맷 : OpCode
0	22	메모리 타입 : (특권을 갖지 않는 배치 버퍼로부터 실행할 때 메모리 타입은 무시될 수 있다.) 퍼 프로세스(Per Process) GTT 인에이블 비트가 클리어되는 경우 이 비트는 1이어야 한다. 0h : 퍼 프로세스 그래픽 어드레스 1h : 글로벌 그래픽 어드레스(글로벌 GTT는 특권을 갖는(안전한) 배치 버퍼로부터 이 커맨드를 실행하는 어드레스를 변환한다)
0	21:16	예비됨
0	15	대기 모드 : (세마포어 비교가 실패할 때 및 콘텍스트가 전환되기 전에 대기 행동을 명시한다.) 1h : 폴링 모드(세마포어 데이터는 그것이 전환된 콘텍스트일 때까지 비교를 위해 메모리로부터 주기적으로 판독된다. 주기성은 SEMA_WAIT_POLL 레지스터와 같은 레지스터에 설정된다. 0H : 신호 모드(세마포어 데이터는 동일한 콘텍스트 ID를 갖는 신호를 수신하면 메모리로부터 다시 취득된다. 스케줄링의 링 버퍼 모드에서 신호와 관련된 콘텍스트 ID는 무시되고 항상 매칭으로서 여겨진다.)
0	14:12	비교 동작 : (콘텍스트가 계속되거나 또는 대기하게 할 것인 결과를 생성하기 위해 실행될 동작을 명시한다.) 0h : SAD > SDD(간접적인 인출된 데이터가 인라인 데이터보다 큰 경우에 계속된다.) 1h : SAD >= SDD(간접적인 인출된 데이터가 인라인 데이터보다 크거나 같은 경우에 계속된다.) 2h : SAD < SDD(간접적인 인출된 데이터가 인라인 데이터보다 작은 경우에 계속된다.) 3h SAD <= SDD(간접적인 인출된 데이터가 인라인 데이터보다 작거나 같은 경우에 계속된다.)
0	11:8	예비됨
0	7:0	DWord 길이 : 디폴트 값 : 1h 포맷 : =n
1	31:0	세마포어 데이터 DWord : (이 데이터 DWord는 커맨드 버퍼의 실행을 제어한다. 세마포어 어드레스에서의 데이터가 이 DWord보다 큰 경우, 커맨드 버퍼의 실행은 계속된다.) 포맷 : U32
2	31:0	세마포어 어드레스 : (이것은 세마포어의 32비트 값의 그래픽 메모리 어드레스이다
3	31:0	세마포어 64비트 어드레스 : (이 필드는 호스트의 64비트 가상 어드레스 공간 내의 그래픽 4GB 가상 어드레스 공간의 4GB 조정된 베이스 어드레스를 명시한다.)

[0048] 표 2 MI_SEMAPHORE_WAIT

[0049] 도 3은 상술한 실시 형태에 따른 생산자 및 소비자 엔진 사이에서 콘텍스트를 동기화하기 위한 방법의 대안적 처리 흐름도이다. 311에서, 생산자 엔진은 콘텍스트를 실행한다. 콘텍스트는 커맨드 스트림을 갖고 생산자는 스트림에 있어서의 커맨드를 실행한다. 세마포어 신호 커맨드에 도달하면, 313에서 생산자는 소비자 엔진에 세마포어 신호를 보낸다. 커맨드 스트림에 있어서의 다수의 세마포어 커맨드를 사용하여, 생산자는 하나 이상의 소비자에게 다수의 세마포어 신호를 보낼 수 있다. 세마포어 신호의 콘텐츠는 커맨드에 의해 제공될 수 있거나 다른 방법으로 얻어질 수 있다. 커맨드는 신호가 영향을 미치는 콘텍스트의 식별, 소비자 또는 타겟 엔진의 식별 및 다양한 다른 프로세싱 상세 및 옵션을 포함할 수 있다.

[0050] 세마포어 신호는 콘텍스트 식별자만을 포함할 수 있거나 또는 다른 정보, 예컨대, 생산자가, 예컨대, 메모리 어드레스를 사용하여 갱신한 영향을 받은(affected) 메모리 레지스터 및 세마포어 신호를 수신하는 것에 응답하여 취할 행동을 포함할 수 있다. 세마포어 신호는 생산자 엔진이 메모리 레지스터를 갱신한 후 또는 소비자 엔진이 필요로 하는 몇몇의 다른 결과를 생성한 후에 보내질 수 있다.

[0051] 한 예에서, 세마포어 신호는 인라인 데이터 및 비교 동작을 포함한다. 비교 동작은 여러 가지의 보다 복잡한 기능보다 크거나, 작거나, 같거나, 또는 그 중 하나일 수 있다. 신호에 있어서의 인라인 데이터는 갱신된 메모리 어드레스에서의 데이터와 비교된다. 이로부터 소비자 엔진은 갱신된 메모리 어드레스에 있어서의 데이터가

갱신되었는지 여부를 결정할 수 있다.

[0052] 315에서 소비자 엔진은 세마포어 신호를 수신하고, 317에서 생산자 엔진에 신호의 수신 확인을 한다. 신호를 수신하면, 소비자 엔진은 신호의 커맨드를 처리할 수 있다. 우선 319에서, 소비자 엔진은 세마포어 신호의 콘텍스트 ID가 소비자 엔진에서 현재 처리되고 있는 콘텍스트의 콘텍스트 ID에 매칭되는지 여부를 결정한다. 매칭되는 경우에 321에서 소비자는 비교 동작을 적용한다. 비교 동작이 확인된 경우에 331에서 소비자 엔진은 현재의 콘텍스트의 실행을 계속한다. 세마포어 신호는, 데이터가 콘텍스트의 커맨드를 실행함에 있어서 소비자 엔진에 의해 사용될 준비가 되었는지 체크된다.

[0053] 비교 동작이 확인되지 않는 경우에, 소비자는 생산자가 레지스터를 갱신하기를 기다릴 수 있다. 생산자는 기다릴지 여부를 세마포어 신호에 나타냄으로써 이것을 제어할 수 있다. 323에서 세마포어 신호에서 또는 몇몇의 다른 방법으로 폴링이 인에이블되는 경우, 325에서 소비자 엔진은 대기할 것이다. 대기는 간단한 타이머이거나 또는 도 2b에 나타낸 바와 같은 보다 복잡한 동작일 수 있다. 또 다른 실시 형태에서 대기는 소비자 엔진이 생산자로부터의 세마포어 신호를 기다릴 것을 요구한다. 대기 후에 321에서 데이터가 비교를 충족시키는 경우, 331에서 소비자는 콘텍스트의 실행을 계속한다.

[0054] 321에서 비교가 부정적이고 323에서 폴링이 인에이블되지 않는 경우 또는 19에서 세마포어 신호의 콘텍스트 ID가 현재의 콘텍스트에 매칭되지 않는 경우, 소비자는 현재의 콘텍스트를 종료하고 소비자가 커맨드 스트리머로부터 수신하는 다음의 콘텍스트로 갈 것이다. 이를 위해, 소비자는 327에서 매칭이 없었음을 스케줄러에 알린다. 스케줄러는 동일한 또는 상이한 소비자 엔진으로 그 콘텍스트를 다시 스케줄링할 것이다.

[0055] 도 4는 본 발명과 함께 사용하기에 적합한 그래픽 프로세싱 유닛의 일반화된 하드웨어 도면이다. GPU(201)는 ALU(101)를 포함하는 커맨드 스트리머(211)를 포함한다. 커맨드 스트리머로부터의 데이터는 미디어 파이프라인(213)에 인가된다. 커맨드 스트리머는 또한 3D 고정 기능 파이프라인(215)에 연결된다. 커맨드 스트리머는 파이프라인 사이의 전환 및 활성 파이프라인에 커맨드 스트림을 포워딩하는 것에 의해 3D 및 미디어 파이프라인의 사용을 관리한다. 미디어 파이프라인이 보다 일반적인 기능을 행하는 반면 3D 파이프라인은 특수화된 프리미티브 프로세싱 기능을 제공한다. 3D 렌더링을 위해, 3D 파이프라인은 버텍스 버퍼(217)에 의해 공급되는 한편 미디어 파이프라인은 메모리 오브젝트(219)의 별도의 그룹에 의해 공급된다. 3D 및 미디어 파이프라인으로부터의 중간 결과와 커맨드 스트리머로부터의 커맨드는 파이프라인 및 커맨드 스트리머에 직접 연결되는 그래픽 서브시스템(221)에 공급된다.

[0056] 그래픽 서브시스템(221)은 그래픽 프로세싱 코어의 어레이(225)에 연결되는 통합 리턴 버퍼(223)를 포함한다. 그래픽 프로세싱 코어의 어레이는 위에서 생산자 엔진 및 소비자 엔진으로 불린다. 통합 리턴 버퍼는 스레드로 하여금 다른 기능 또는 스레드에 의해 나중에 소비될 데이터를 리턴하게 하기 위해 다양한 기능에 의해 공유되는 메모리를 포함한다. 코어의 어레이(225)는 최종적으로 목적지 표면(227)을 생성하기 위해 파이프라인 스트리머로부터의 값을 처리한다. 코어의 어레이는 샘플러 기능부(229), 수학 기능부(231), 인터 스레드 통신(233), 색 계산기(235), 및 최종적으로 렌더링된 표면을 캐시에 저장하기 위한 렌더 캐시(237)에 액세스한다. 소스 표면(239)의 세트는 그래픽 서브시스템(221)에 적용되고 이들 기능부(229, 231, 235, 237, 239)의 전부가 코어의 어레이에 의해 적용된 후, 목적지 표면(227)의 세트가 생성된다. 범용 계산을 목적으로, 특정한 구현에 따라, ALU에 대해서만 또는 코어의 어레이(225)도 거쳐서 동작을 실행하기 위해 커맨드 스트리머(221) 및 ALU가 사용된다.

[0057] 도 5를 참조하면, 그래픽 코어(201)는 보다 큰 컴퓨터 시스템(501)의 일부로서 나타내어진다. 컴퓨터 시스템은 DMI(Direct Media Interface)(507)를 통해서 입력/출력 컨트롤러 허브(ICH)(505)에 연결되는 CPU(503)를 갖는다. CPU는 그래픽 코어(201)에 연결되는 범용 컴퓨팅을 위한, 라스트 레벨 캐시(Last Level Cache)(511)를 공유하는 하나 이상의 코어(509)를 갖는다. CPU는 메모리 인터페이스(515), 디스플레이 인터페이스(571), 및 PCIe 인터페이스(519)와 같은 시스템 에이전트(513)를 포함한다. 도시된 예에서, PCIe 인터페이스는 PCI 익스프레스 그래픽을 위한 것이고 디스플레이(도시하지 않음)에 연결될 수 있는 그래픽 어댑터(521)에 연결될 수 있다. 제 2 또는 대체 디스플레이(523)는 시스템 에이전트의 디스플레이 모듈에 연결될 수 있다. 이 디스플레이는 그래픽 코어(201)에 의해 구동될 것이다. 메모리 인터페이스(515)는 시스템 메모리(525)에 연결된다.

[0058] 입력/출력 컨트롤러 허브(505)는 매스 스토리지(531), 외부 주변 장치(533), 및 키보드 및 마우스와 같은 사용자 입력/출력 장치(535)를 포함한다. 입력/출력 컨트롤러 허브는 또한 디스플레이 인터페이스(537) 및 다른 추가 인터페이스를 포함할 수 있다. 디스플레이 인터페이스(537)는 비디오 프로세싱 서브시스템(539) 내에 있다. 서브시스템은 디스플레이 링크(541)를 통해서 CPU의 그래픽 코어에 선택적으로 연결될 수 있다.

[0059] 폭넓은 추가 및 대체 장치가 도 5에 나타낸 컴퓨터 시스템(501)에 연결될 수 있다. 혹은, 본 발명의 실시 형태는 나타낸 것들과는 상이한 아키텍처 및 시스템에 적합할 수 있다. 추가 요소가 나타낸 기존 유닛에 통합될 수 있고 보다 많은 또는 보다 적은 하드웨어 요소가 서술된 기능을 제공하기 위해 사용될 수 있다. 서술된 기능 중 하나 이상은 완전한 시스템으로부터 삭제될 수 있다.

[0060] 그래픽 코어(201)는 일반 프로세싱 코어(509) 및 다른 요소도 포함하는 CPU와 통합되는 것으로 나타내어지지만, 그래픽 코어는 LLC 및 범용 코어로의 통신 인터페이스를 갖는 별도의 요소로서 구성될 수 있다. 이와 달리, 예컨대 도 4에 나타낸 바와 같은 그래픽 코어 및 그 관련된 요소는 동일한 또는 상이한 패키지에 둘러싸이는 독립된 다이에 구성될 수 있다. 메모리(525), ICH(505) 및 시스템 에이전트(513)와 같은 다른 요소도 동일한 또는 상이한 패키지에서 동일한 또는 상이한 다이에 구성될 수 있다.

[0061] 본 발명의 실시 형태는 시그널링을 사용하여 공유된 메모리에서 데이터를 동기화하기 위해 생산자 엔진 및 소비자 엔진에서의 메커니즘을 제공한다. 서술된 예에서, 이것은 MI_SEMAPHORE_SIGNAL 커맨드 및 MI_SEMAPHORE_WAIT 커맨드를 사용하여 행해지지만, 본 발명은 이것으로 한정되지 않는다. 이것은 코어의 어레이에 있어서 코어에 의해 실행되는 커맨드 스트림에 프로그래밍되는, 커맨드를 사용한 산술적 및 논리적 동작을 행하기 위한 하드웨어 구조인 커맨드 스트리머에 있어서 신호 및 커맨드 메커니즘을 제공한다.

[0062] 폭넓은 추가 및 대체 장치가 도 5에 나타낸 컴퓨터 시스템(501)에 연결될 수 있다. 혹은, 본 발명의 실시 형태는 나타낸 것들과는 상이한 아키텍처 및 시스템으로 조정될 수 있다. 추가 요소가 나타낸 기존 유닛에 통합될 수 있고 보다 많은 또는 보다 적은 하드웨어 요소가 서술된 기능을 제공하기 위해 사용될 수 있다. 서술된 기능 중 하나 이상은 완전한 시스템으로부터 삭제될 수 있다.

[0063] 상술한 예보다 적은 또는 많은 장비를 갖춘 시스템은 특정한 구현에 대하여 바람직할 수 있다. 따라서, 예시적인 시스템 및 회로의 구성은 가격 제약, 성능 요건, 기술적 진보, 또는 다른 환경과 같은 많은 요인에 의존하여 구현마다 달라질 수 있다.

[0064] 실시 형태는 마더보드를 사용하여 서로 연결된 하나 이상의 마이크로칩 또는 집적 회로, 하드웨어에 내장된 로직, 메모리 장치에 의해 저장되고 마이크로프로세서에 의해 실행되는 소프트웨어, 펌웨어, 특정 용도용 집적 회로(ASIC), 및/또는 필드 프로그래머블 게이트 어레이(FPGA) 중의 하나 또는 그 조합으로서 구현될 수 있다. 용어 "로직"은 소프트웨어 또는 하드웨어 및/또는 소프트웨어 및 하드웨어의 조합을 예로서 포함할 수 있다.

[0065] "한 실시 형태", "실시 형태", "예시적 실시 형태", "다양한 실시 형태" 등에 대한 언급은 그렇게 서술된 본 발명의 실시 형태(들)가 특정한 특징, 구조, 또는 특성을 포함할 수 있음을 나타내지만, 모든 실시 형태가 반드시 특정한 특징, 구조, 또는 특성을 포함하지는 않는다. 또한, 몇몇의 실시 형태는 다른 실시 형태를 위해 서술된 특징의 일부 또는 전부를 가질 수 있고 또는 갖지 않을 수도 있다.

[0066] 상세한 설명 및 청구 범위에서, 용어 "연결된다"는 그 파생어와 함께 사용될 수 있다. "연결된다"는 2개 이상의 요소가 서로 협력하거나 또는 상호 작용하지만, 그들 사이에 중간의 물리적 또는 전기적 요소를 가질 수 있거나 또는 갖지 않을 수 있음을 나타내기 위해 사용된다.

[0067] 청구 범위에서 사용되는 바와 같이, 특별히 명시되지 않는 한, 공통 요소를 서술하기 위한 서수의 형용사 "제 1", "제 2", "제 3" 등의 사용은, 단지 동일한 요소의 상이한 예가 그렇게 서술된 요소는 순위를 매김에 있어서 일시적으로 또는 공간적으로 정해진 순서를 따라야 하거나 또는 임의의 다른 방식을 따라야 함을 의미하도록 언급되고 있지만 그것으로 의도되지 않음을 나타낸다.

[0068] 도면 및 상술한 서술은 실시 형태의 예를 제공한다. 당업자는 서술된 요소 중 하나 이상이 단일 기능 요소로 잘 조합될 수 있음을 이해할 것이다. 혹은, 특정한 요소는 다수의 기능 요소로 분할될 수 있다. 한 실시 형태로부터의 요소는 다른 실시 형태에 추가될 수 있다. 예컨대, 본 명세서에 서술된 처리의 순서는 변경될 수 있고 본 명세서에 서술된 방식으로 한정되지 않는다. 또한, 임의의 흐름도의 동작은 나타낸 순서로 구현될 필요도 없고, 행해질 필요가 있는 동작의 전부를 반드시 행하지도 않는다. 또한, 다른 동작에 의존하지 않는 그러한 동작은 다른 동작과 병행하여 행해질 수 있다. 실시 형태의 범위는 이들 특정한 예에 의해 결코 한정되지 않는다. 명세서에 명확하게 주어지는지 여부에 관계없이, 구조, 치수, 및 재료의 사용에 있어서의 차이와 같은 많은 변형이 가능하다. 실시 형태의 범위는 적어도 이하의 청구 범위에 의해 주어지는 것만큼 넓다.

[0069] 이하의 예는 추가 실시 형태에 관련된다. 예에 있어서의 사양은 하나 이상의 실시 형태의 어디에서도 사용될 수 있다. 한 실시 형태에서, 생산자 엔진과 소비자 엔진 사이에서 콘텍스트를 동기화하기 위한 방법은 생산자

엔진에서 콘텍스트를 실행하는 단계-상기 실행하는 단계는 메모리 레지스터를 갱신하는 단계를 포함함-와, 상기 생산자 엔진으로부터 소비자 엔진에 상기 메모리 레지스터가 갱신되었다는 신호를 보내는 단계-상기 신호는 상기 레지스터를 갱신하기 위해 상기 소비자 엔진에 의해 실행되는 콘텍스트를 식별하기 위한 콘텍스트 ID를 포함함-를 포함한다.

[0070] 추가 실시 형태는 상기 방법을 포함하고 상기 신호는 상기 갱신된 메모리 레지스터 인라인 데이터를 식별하기 위한 메모리 레지스터 어드레스를 포함한다.

[0071] 추가 실시 형태는 생산자 엔진으로부터 소비자 엔진에 보내지는 상기 신호를 생성하는 단계-상기 메시지는 상기 소비자 엔진 및 상기 소비자 엔진에 의해 상기 신호가 인가되는 콘텍스트 ID를 식별함-를 또한 포함하는 상기 방법을 포함한다.

[0072] 추가 실시 형태는 스케줄러에서 상기 콘텍스트 ID를 생성하고 상기 소비자 엔진에 상기 콘텍스트 ID를 배치하는 단계-상기 콘텍스트 ID는 상기 소비자 엔진을 식별하기 위한 엔진 ID를 포함함-를 또한 포함하는 상기 방법을 포함한다.

[0073] 추가 실시 형태는 상기 소비자 엔진에서 상기 신호를 수신하고 상기 생산자 엔진에 대하여 상기 소비자 엔진에 의해 상기 신호의 수신 확인을 하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0074] 추가 실시 형태는 상기 신호의 상기 콘텍스트 ID가 상기 소비자 엔진에서 현재 처리되고 있는 콘텍스트의 콘텍스트 ID에 매칭되는지 결정하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0075] 추가 실시 형태는 상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되지 않는 경우에 스케줄러에 상기 신호를 포워딩하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0076] 추가 실시 형태는 상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되지 않는 경우에 상기 스케줄러에서 상기 신호 메시지의 상기 콘텍스트 ID에 의해 식별되는 상기 콘텍스트를 다시 스케줄링하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0077] 추가 실시 형태는 상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되는 경우에 상기 소비자 엔진에서 기다리는 단계를 또한 포함하는 상기 방법을 포함한다. 추가 실시 형태는 상기 방법 중 어느 하나를 포함하고 기다리는 단계는 타이머를 기다리는 단계를 포함한다.

[0078] 추가 실시 형태는 상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되는 경우에 상기 소비자 엔진에서 상기 콘텍스트를 실행함에 있어서 사용하기 위해 상기 메모리 레지스터로부터 데이터를 인출하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0079] 한 실시 형태에서, 머신에 의해 실행될 때에 상기 머신이 동작을 행하게 하는 인스트럭션을 갖는 머신 판독 가능 매체는 생산자 엔진에서 콘텍스트를 실행하는 단계-상기 실행하는 단계는 메모리 레지스터를 갱신하는 단계를 포함함-와, 상기 생산자 엔진으로부터 소비자 엔진에 상기 메모리 레지스터가 갱신되었다는 신호를 보내는 단계-상기 신호는 상기 레지스터를 갱신하기 위해 상기 소비자 엔진에 의해 실행되는 콘텍스트를 식별하기 위한 콘텍스트 ID를 포함함-를 포함한다.

[0080] 추가 실시 형태는 생산자 엔진으로부터 소비자 엔진에 보내지는 상기 신호를 또한 포함하는 상기 인스트럭션을 포함하고, 상기 메시지는 상기 소비자 엔진 및 상기 소비자 엔진에 의해 상기 신호가 인가되는 콘텍스트 ID를 식별한다.

[0081] 추가 실시 형태는 스케줄러에서 상기 콘텍스트 ID를 생성하고 상기 소비자 엔진에 상기 콘텍스트 ID를 배치하는 단계를 또한 포함하는 상기 인스트럭션을 포함하고, 상기 콘텍스트 ID는 상기 소비자 엔진을 식별하기 위한 엔진 ID를 포함한다.

[0082] 한 실시 형태에서 장치는 메모리 레지스터를 갱신하는 것을 포함하는 콘텍스트를 실행하고 상기 메모리 레지스터가 갱신되었다는 신호를 소비자 엔진에 보내기 위한 생산자 엔진-상기 신호는 상기 레지스터를 갱신하기 위해 상기 소비자 엔진에 의해 실행되는 콘텍스트를 식별하기 위한 콘텍스트 ID를 포함함-과, 상기 신호를 수신하고 상기 생산자 엔진에 대하여 상기 신호의 수신 확인을 하기 위한 소비자 엔진을 포함한다.

[0083] 추가 실시 형태는 상기 장치를 포함하고 상기 소비자 엔진은 상기 신호의 상기 콘텍스트 ID가 상기 소비자 엔진에서 현재 처리되고 있는 콘텍스트의 콘텍스트 ID에 매칭되는지 결정한다.

[0084] 추가 실시 형태는 상기 장치를 포함하고 상기 소비자 엔진은 상기 신호 메시지의 상기 콘텍스트 ID가 상기 소비자 엔진에 의해 현재 처리되고 있는 상기 콘텍스트에 매칭되는 경우에 상기 소비자 엔진에서 상기 콘텍스트를 실행함에 있어서 사용하기 위해 메모리 레지스터로부터 데이터를 인출한다.

[0085] 한 실시 형태에서 생산자 엔진과 소비자 엔진 사이에서 콘텍스트를 동기화하기 위한 방법은, 콘텍스트의 커맨드 스트림을 실행하는 단계-상기 커맨드 스트림은 대기 커맨드를 갖고, 상기 대기 커맨드는 세마포어 메모리 어드레스, 인라인 세마포어 데이터 및 비교 연산자를 가짐-와, 상기 세마포어 메모리 어드레스로부터 데이터를 인출하고 그것을 상기 인라인 세마포어 데이터와 비교함으로써 상기 대기 커맨드를 실행하는 단계와, 상기 비교 연산자의 상기 비교가 충족되는 경우에 상기 커맨드 스트림에 있어서의 다음 인스트럭션을 실행하는 단계를 포함한다.

[0086] 추가 실시 형태는 상기 비교가 충족되지 않는 경우에 기다리고 데이터를 인출하고 비교하는 것을 반복하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0087] 추가 실시 형태는 상기 방법 중 어느 하나를 포함하고 기다리는 단계는 타이머를 기다리는 단계를 포함한다.

[0088] 추가 실시 형태는 상기 방법 중 어느 하나를 포함하고 기다리는 단계는 생산자 엔진으로부터 세마포어 신호를 수신하기를 기다리는 단계를 포함한다.

[0089] 추가 실시 형태는 상기 비교가 충족되지 않는 경우에 상기 콘텍스트를 전환하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0090] 추가 실시 형태는 상기 비교가 충족되지 않는 경우에 스케줄러에 알리는 단계를 또한 포함하는 상기 방법을 포함한다.

[0091] 추가 실시 형태는 콘텍스트 스케줄러에서 상기 대기 커맨드를 갱신하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0092] 추가 실시 형태는 상기 소비자 엔진의 상기 콘텍스트를 전환하기 전에 스케줄러에서 상기 대기 조건을 재평가하는 단계를 또한 포함하는 상기 방법을 포함한다.

[0093] 한 실시 형태에서, 머신에 의해 실행될 때에 상기 머신이 동작을 행하게 하는 인스트럭션을 갖는 머신 판독 가능 매체는, 콘텍스트의 커맨드 스트림을 실행하는 단계-상기 커맨드 스트림은 대기 커맨드를 갖고, 상기 대기 커맨드는 세마포어 메모리 어드레스, 인라인 세마포어 데이터 및 비교 연산자를 가짐-와, 상기 세마포어 메모리 어드레스로부터 데이터를 인출하고 그것을 상기 인라인 세마포어 데이터와 비교함으로써 상기 대기 커맨드를 실행하는 단계와, 상기 비교 연산자의 상기 비교가 충족되는 경우에 상기 커맨드 스트림에 있어서의 다음 인스트럭션을 실행하는 단계를 포함한다.

[0094] 추가 실시 형태는 상기 비교가 충족되지 않는 경우에 스케줄러에 알리는 단계를 또한 포함하는 상기 인스트럭션을 포함한다.

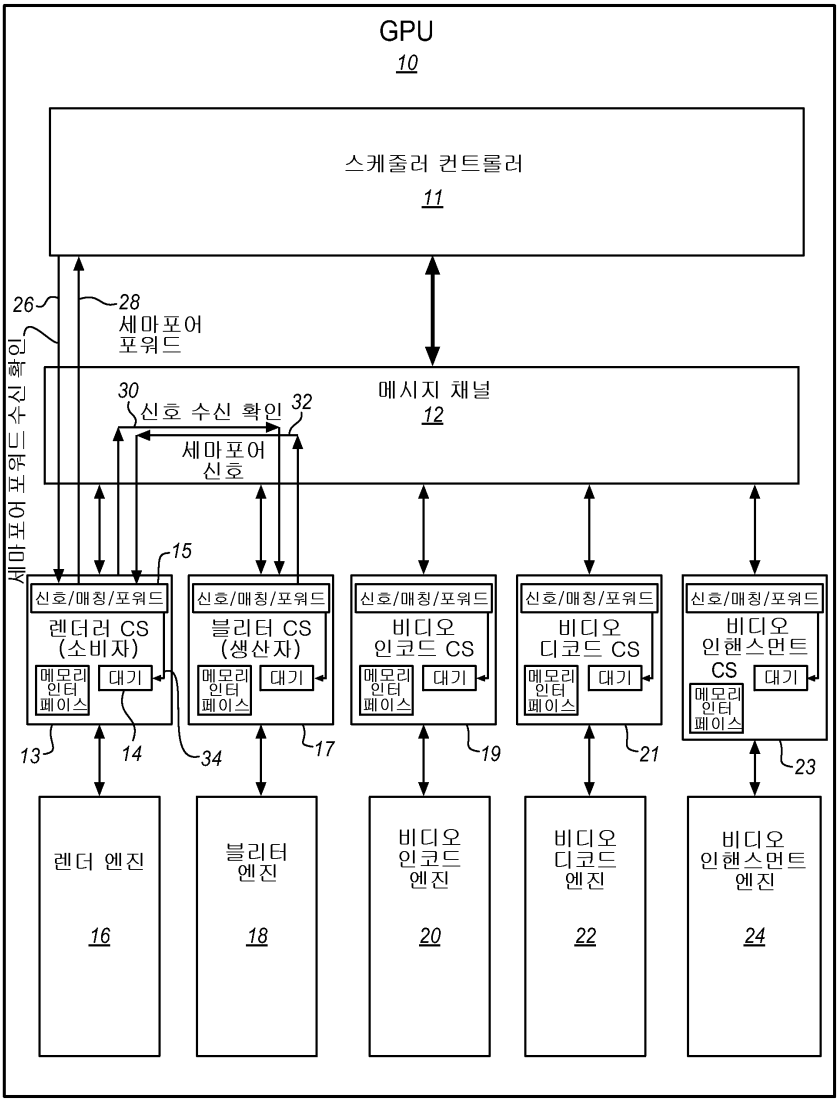
[0095] 추가 실시 형태는 콘텍스트 스케줄러에서 상기 대기 커맨드를 갱신하는 단계를 또한 포함하는 상기 인스트럭션을 포함한다.

[0096] 한 실시 형태에서, 장치는 어드레스를 갖는 메모리와, 콘텍스트를 위한 커맨드 스트림을 갖는 커맨드 스트리머와, 상기 콘텍스트의 상기 커맨드 스트림을 실행하기 위한 소비자 엔진-상기 커맨드 스트림은 대기 커맨드를 갖고, 상기 대기 커맨드는 상기 메모리로의 세마포어 메모리 어드레스, 인라인 세마포어 데이터 및 비교 연산자를 가짐-을 포함하고, 상기 소비자 엔진은 상기 세마포어 메모리 어드레스에서 상기 메모리로부터 데이터를 인출함으로써 상기 대기 커맨드를 실행하고 상기 인출된 데이터를 상기 인라인 세마포어 데이터와 비교하고, 상기 소비자 엔진은 상기 비교 연산자의 상기 비교가 충족되는 경우에 상기 커맨드 스트림에 있어서의 다음 인스트럭션을 실행한다.

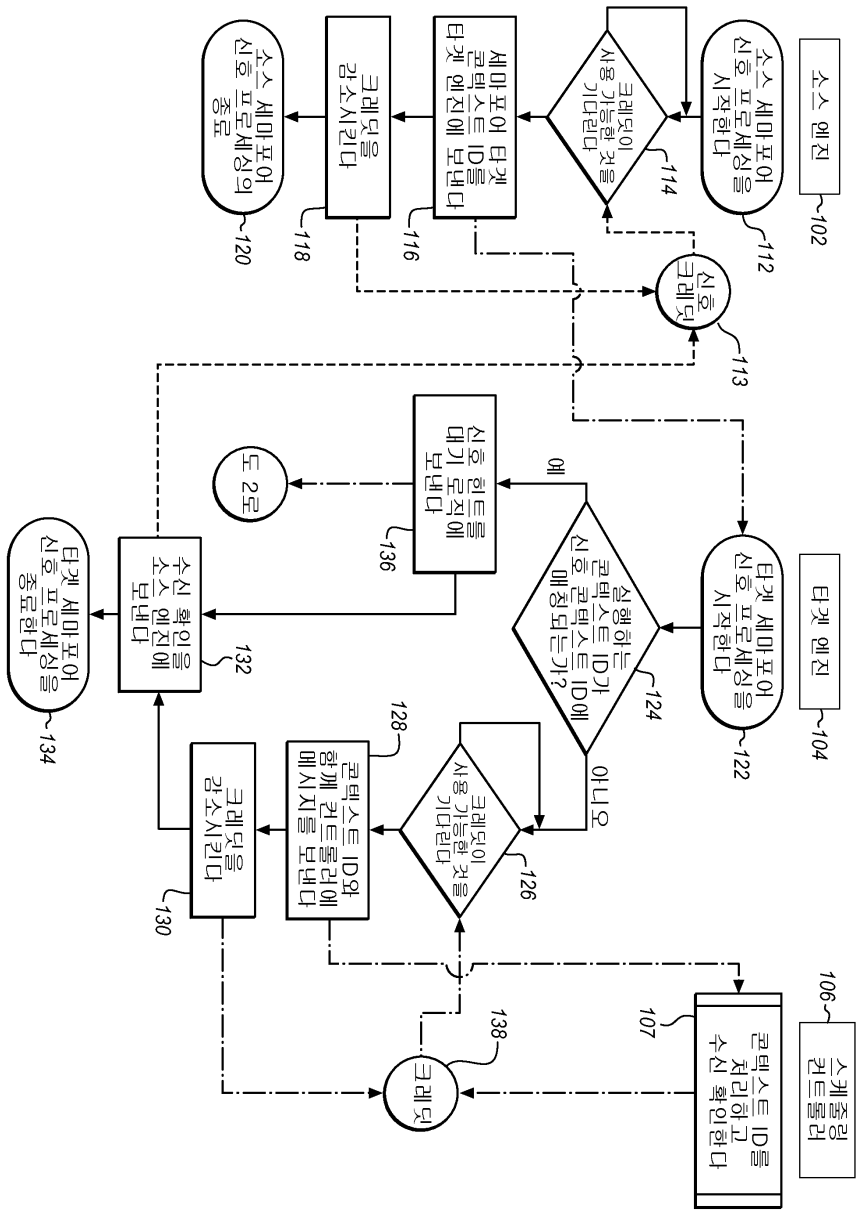
[0097] 추가 실시 형태는 상기 비교가 충족되지 않는 경우에 상기 대기 커맨드를 갱신하기 위해 상기 소비자 엔진에 접속되는 스케줄러를 또한 포함하는 상기 장치 중 어느 하나를 포함한다. 추가 실시 형태에서 상기 스케줄러는 상기 소비자 엔진의 상기 콘텍스트를 전환하기 전에 상기 대기 조건을 재평가한다.

도면

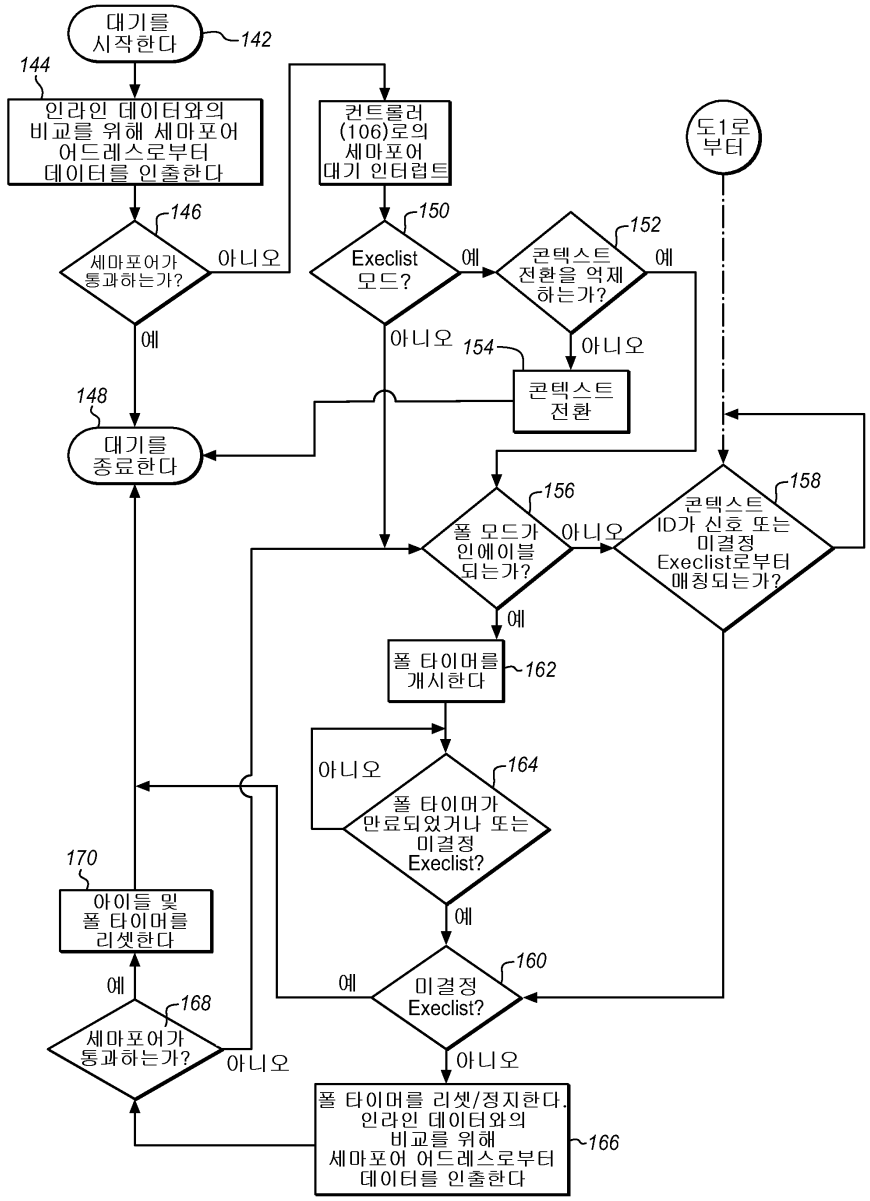
도면1


GPU
10
스케줄러 컨트롤러
11
26
28
세마포어
포워드
세마포어 포워드 수신 확인
메시지 채널
12
30
신호 수신 확인
32
세마포어
신호
15
신호/매칭/포워드
렌더러 CS
(소비자)
메모리
인터
페이스
대기
블리터 CS
(생산자)
비디오
인코드 CS
비디오
디코드 CS
비디오
인핸스먼트
CS
13
14
34
17
19
21
23
렌더 엔진
16
블리터
엔진
18
비디오
인코드
엔진
20
비디오
디코드
엔진
22
비디오
인핸스먼트
엔진
24

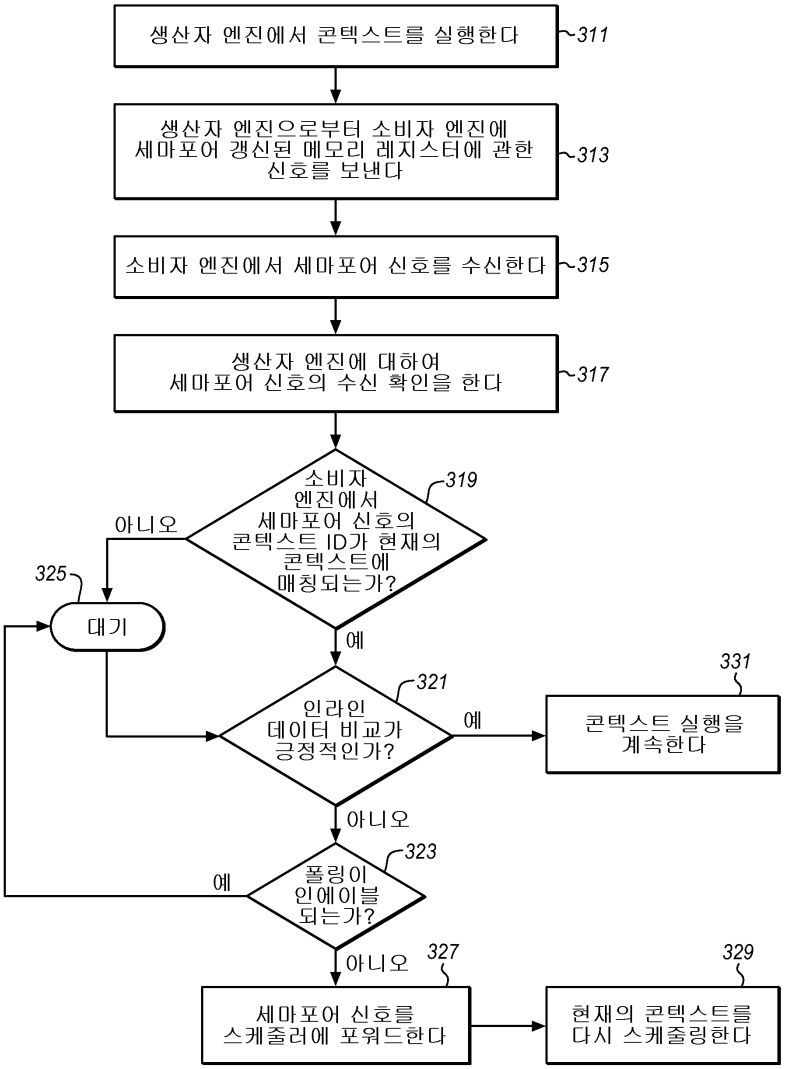
도면2a


소스 엔진
102
소스 세마포어 신호 프로세싱을 시작한다
112
크레딧이 사용가능할 것을 기다린다
114
세마포어 타겟 콘텍스트 ID를 타겟 엔진에 보낸다
116
크레딧을 감소시킨다
118
소스 세마포어 신호 프로세싱의 종료
120
신호 크레딧
113
타겟 엔진
104
타겟 세마포어 신호 프로세싱을 시작한다
122
실행하는 콘텍스트 ID가 신호 콘텍스트 ID에 매칭되는가?
124
예
아니오
신호 힌트를 대기 모션에 보낸다
136
도 2로
수신 확인을 소스 엔진에 보낸다
132
타겟 세마포어 신호 프로세싱을 종료한다
134
크레딧이 사용가능할 것을 기다린다
126
128
콘텍스트 ID와 함께 컨트롤러에 메시지를 보낸다
크레딧을 감소시킨다
130
스케줄링 컨트롤러
106
콘텍스트 ID를 처리하고 수신 확인한다
107
138
크레딧

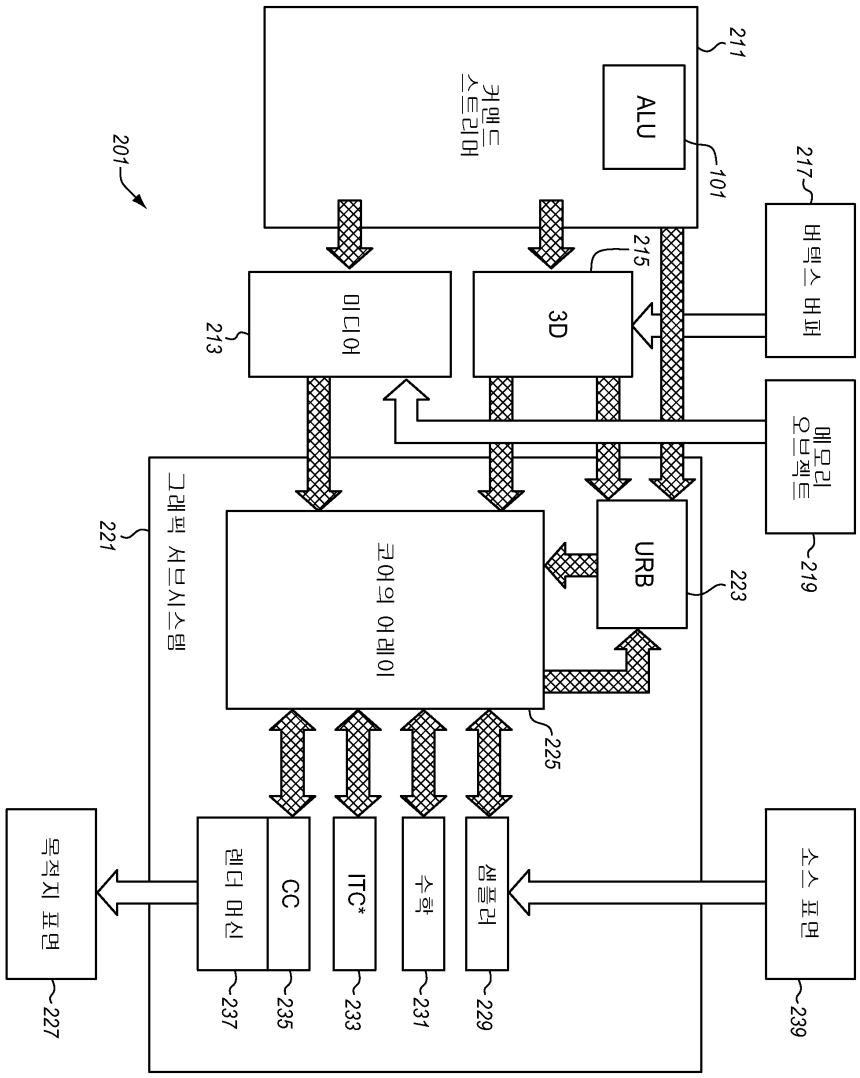
도면2b


대기를 시작한다
142
144
인라인 데이터와의 비교를 위해 세마포어 어드레스로부터 데이터를 인출한다
146
세마포어가 통과하는가?
아니오
예
148
대기를 종료한다
컨트롤러 (106)로의 세마포어 대기 인터럽트
150
Execlist 모드?
예
아니오
152
콘텍스트 전환을 억제하는가?
예
아니오
154
콘텍스트 전환
도1로부터
156
폴 모드가 인에이블 되는가?
아니오
158
콘텍스트 ID가 신호 또는 미결정 Execlist로부터 매칭되는가?
예
폴 타이머를 개시한다
162
아니오
164
폴 타이머가 만료되었거나 또는 미결정 Execlist?
예
160
미결정 Execlist?
예
아니오
170
아이들 및 폴 타이머를 리셋한다
예
168
세마포어가 통과하는가?
아니오
폴 타이머를 리셋/정지한다. 인라인 데이터와의 비교를 위해 세마포어 어드레스로부터 데이터를 인출한다
166

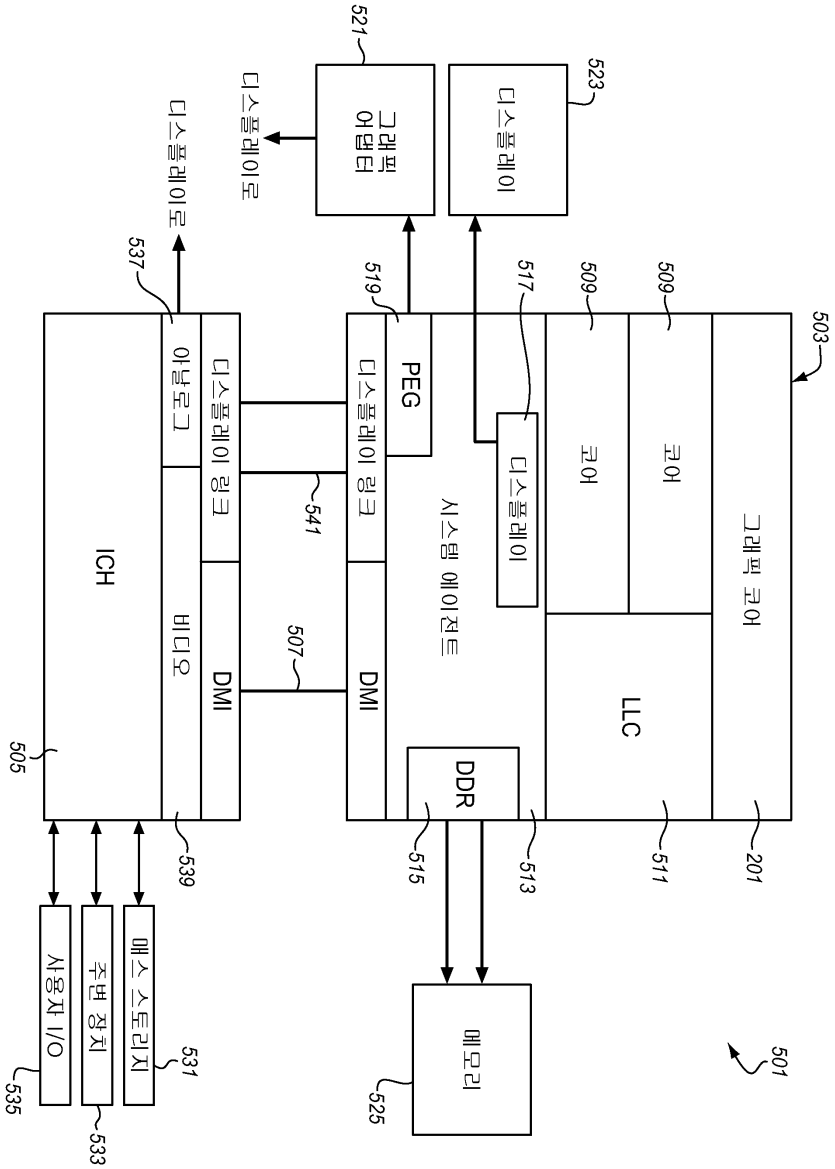
도면3


생산자 엔진에서 콘텍스트를 실행한다
311
생산자 엔진으로부터 소비자 엔진에 세마포어 갱신된 메모리 레지스터에 관한 신호를 보낸다
313
소비자 엔진에서 세마포어 신호를 수신한다
315
생산자 엔진에 대하여 세마포어 신호의 수신 확인을 한다
317
소비자 엔진에서 세마포어 신호의 콘텍스트 ID가 현재의 콘텍스트에 매칭되는가?
319
아니오
예
325
대기
인라인 데이터 비교가 긍정적인가?
321
예
331
콘텍스트 실행을 계속한다
아니오
폴링이 인에이블 되는가?
323
예
아니오
327
세마포어 신호를 스케줄러에 포워드한다
329
현재의 콘텍스트를 다시 스케줄링한다

도면4


201
211
ALU
101
커맨드 스트리머
217
버텍스 버퍼
215
3D
213
미디어
메모리 오브젝트
219
그래픽 서브시스템
221
코어의 어레이
URB
223
225
목적지 표면
227
렌더 머신
237
CC
235
ITC*
233
수학
231
샘플러
229
소스 표면
239

도면5


521
그래픽 어댑터
디스플레이로
523
디스플레이
디스플레이로
537
519
517
509
509
503
아날로그
디스플레이 링크
PEG
디스플레이
코어
코어
그래픽 코어
541
ICH
비디오
DMI
DMI
시스템 에이전트
LLC
507
505
539
515
513
511
201
DDR
메모리
501
사용자 I/O
535
주변 장치
533
매스 스토리지
531
525