진보된 캘린더 큐 스케줄러 설계방법론

정회원 김 진 실*, 정 원 영*, 이 정 희**, 이 용 석*

Advanced Calendar Queue Scheduler Design Methodology

Jinsil Kim*, Won-young Chung*, Jung-hee Lee**, Yong-surk Lee* ***Regular Members***

요 약

본 논문에서는 홈 네트워크에서 멀티미디어와 타이밍 트래픽을 처리하기 위해 디자인 된 CQS(Calendar Queue Scheduler)를 제안한다. VoIP, VOD, IPTV, 최선형(Beat-efforts) 트래픽 등 가택으로 유입되는 다양한 속성을 지닌 트래픽의 증가로 가택 내 QoS(Quality of Service) 관리의 필요성이 논의되고 있다. 이러한 제한된 환경에서 성공적으로 QoS를 보장하기 위해서는 각 애플리케이션이나 서비스 단위로 그룹을 형성하여 관리하는 것이 효과적이다. 본 연구에서는 단대단(end-to-end) QoS 측면에서 수신측 말단에 해당하는 홈 게이트웨이를 목표로 제한된 자원내에서 멀티미디어 및 타이밍 트래픽 처리와 큐 사이즈를 최적화시킨 CQS아키텍처를 하드웨어로 제안하였다. 또한, 각각의 모듈과 각각의 메모리에 대한 면적을 시뮬레이션하였다. Synopsys Design Compiler를 사용하여 Magnachip 0.18 CMOS 라이브러리로 합성하였을 때 각 모듈의 면적은 NAND(2x1) 게이트(11.09)를 기준으로 하였다. Memory의 비중이 전체 CQS에서 85.38%를 나타내고 있음을 알 수 있었다. 각 메모리 사이즈의 크기를 CACTI 5.3(단위는 mm^2)을 통하여 추출하였다. 메모리의 entry가 증가함에 따라 메모리 area의 증가 폭은 점점 더 증가하므로, 1 year 에 해당하는 day size의 결정이 전체 CQS 면적에 절대적인 영향을 미치게 된다. 본 논문에서 CQS를 하드웨어로 설계할 때 각 모듈의 설계 방법론과 각 모듈의 동작에 대하여 논하였다.

Key Words : CQS; Enqueue, Dequeue, Scheduler

ABSTRACT

In this paper, we propose a CQS(Calendar Queue Scheduler) architecture which was designed for processing multimedia and timing traffic in home network. With various characteristics of the increased traffic flowed in home such as VoIP, VOD, IPTV, and Best-efforts traffic, the needs of managing QoS(Quality of Service) are being discussed. Making a group regarding application or service is effective to guarantee successful QoS under the restricted circumstances. The proposed design is aimed for home gateway corresponding to the end points of receiver on end-to-end QoS and eligible for supporting multimedia traffic within restricted network sources and optimizing queue sizes. Then, we simulated the area for each module and each memory. The area for each module is referenced by NAND(2x1) Gate(11.09) when synthesizing with Magnachip 0.18 CMOS libraries through the Synopsys Design Compiler. We verified the portion of memory is 85.38% of the entire CQS. And each memory size is extracted through CACTI 5.3(a unit in mm2). According to the increase of the memory's entry, the increment of memory area gradually increases, and defining the day size for 1 year definitely affects the total CQS area. In this paper, we discussed design methodology and operation for each module when designing CQS by hardware.

※ 본 연구는 지식경제부 및 정보통신연구진흥원의 IT산업원천기술개발사업의 일환으로 수행하였습니다. [2009-S-043-01, Scalable 마이크로 플로우 처리 기술 개발]

* 연세대학교 전기전자공학과 프로세서 연구실 (jskim@mpu.yonsei.ac.kr) ** 한국전자통신연구원 차세대 이더넷 팀 (hilee@etri.re.kr)

논문번호 : KICS2009-08-366, 접수일자 : 2009년 8월 19일, 최종논문접수일자 : 2009년 12월 15일

Ⅰ. 서 론

최근 통신 산업은 다양한 응용 기술의 개발과 고객 수요 증가에 따라 급속도로 발전하고 있다. 통신 네트워크와 시스템 사업자들은 한정된 대역폭 내에서 다양한 서비스를 제공하면서 이를 정해진 시간 내에 처리할 수 있는 기반을 형성하고 있다. 특히 실시간 응용 기술의 경우 높은 대역폭 (throughput), 실시간(real-time) 처리, 지연 (delay)과 지연 지터 (delay jitter) 등에 민감하여 엄격한 성능 조건이 요구되고 있다. 오늘날 VoIP, VOD, IPTV, 최선형 (Beat-efforts) 트래픽 등의 증가로 가택 내 QoS(Quality of Service) 관리의 필요성이 논의되고 있다. 따라서 각 서비스 별로 성능을 보장할 수 있는 기법들의 연구가 활발히 진행 중에 있으며, 기존의 대다수 통신 사업자들은 현재 높은 성능 조건을 만족시키기 위하여 패킷 스위칭 기술(packet switching technology)을 많이 사용하고 있다[1],[2].

전형적으로 스위치 또는 라우터와 같은 네트워크 디바이스는 패킷을 받고(receive), 처리(process) 하고, 보내고(forword), 경우에 따라 폐기(discard) 한다. 예를 들어, 입력 버퍼(input buffer)에 저장되어 있는 다양한 사이즈의 패킷 스트림(packet stream)들은 인큐잉 컴포넌트 (enqueing component)에서 각각의 패킷이 분석되고 이를 저장하기 위한 메모리 공간을 할당한다. 일반적으로 패킷 헤더 또는 다른 소스로부터 얻은 목적지 정보(destination information)등을 이용하여 패킷을 분류하고 분류된 패킷을 각기 다른 메모리 영역에 저장한다. 패킷 전체가 메모리에 쓰여지고 나면 이 패킷은 프로세싱하기에 적합해지고, 패킷 인디케이터 (packet indicator)가 스케줄링 기법에 따라 적합한 목적지 큐에 배치된다. 지터와 대역폭 등에 민감한 멀티미디어 트래픽의 양적 증대로 인하여 이를 효과적으로 관리할 수 있는 적합한 스케줄러의 설계가 필요하다.

본 논문에서는 단대단(end-to-end) QoS 측면에서 수신측 말단에 해당하는 홈 게이트웨이에 내장되어 효과적으로 트래픽을 관리해 줄 CQS(Calendar Queue Scheduler) 아키텍처를 제안하고 설계하였다. 제한된 자원을 가지고 애플리케이션 또는 서비스가 속도 제어(rate control)와 지연 제어(delay control)를 할 수 있는 장점들을 가지고 있는 CQS는 성공적인 QoS를 보장하기 위해 효과적이다[3], [4].

본 논문에서는 새로운 네트워크 관리 아키텍처를 제안하고자 한다. 2장에서는 현재 연구 추이를 살펴보고, 이에 대한 문제점에 대하여 논의한다. 3장에서는 제안하고자 하는 전체적인 아키텍처를 설명한다. 4 장에는 시뮬레이션 결과를 정리한다.

Ⅱ. 기존 연구

현재 차세대 통신망으로 가는 과도기적 현상으로 네트워크 전반에 걸친 ALL-IP conversions 가 두드러지게 나타나고 있으며, 서킷 망이 패킷 망으로 통합되는 추이를 보이면서 IP 패킷을 기반으로 하는 트래픽의 양적 증가 현상이 나타나고 있다. 홈 네트워크에서 멀티미디어, 타이밍 트래픽을 처리하는 것이 필수 요소가 되면서 새로운 애플리케이션과 기존 애플리케이션의 성능 향상에 대한 수요는 통신 네트워크와 시스템이 더 빠른 속도와 넓은 대역폭을 가지도록 유도 해 왔다. 높은 우선순위, 보장된 대역폭, 베스트 에포트 트래픽(best effort traffic), 그리고 트래픽 분류들이 공통 자원에 대한 경쟁을 불러 왔다. 트래픽 차단(isolation), 우선순위, 공평한 대역폭(fair bandwidth) 할당으로 공통 자원을 효율적으로 관리하기 위하여 다양한 스케줄링 기법들이 설계되었으며, 이러한 기법들을 공평한 큐잉 기법(fair queueing methods)이라 한다. 예를 들어, Weighted Fair Queuing (WFQ), Self-Clocked Fair Queueing(SCFQ), and Deficit Round Robin/Surplus Round Robin 등이 이에 해당되며, 좀더 발전된 형태의 스케줄러로는 버츄얼 클록(virtual clock), RCSP(Rate-Controlled Static Priority), 캘린더 큐(calendar queue)등이 있다. 버츄얼 클록은 고속 패킷 교환망에 맞는 스케줄링 알고리즘으로써 데이터 흐름의 평균 전송 속도를 제어하고, 사용자 예약(reservation)에 따라 자원 사용을 강화한다. 또한, 흐름들 속에 파이어월 (firewall)을 구성하고, 멀티우선순위 전송을 지원한다. 이러한 버추얼 클록 알고리즘은 패킷교환망의 라우터에서 수행될 수 있다. RCSP는 지연, 대역폭, 무손실 보장을 제공하고, 분류된 우선순위 큐가 필요하지 않으므로 상대적으로 구현하기 쉽다.

특히, CQS는 속도제어 및 지연제어와 단대단 QoS 측면에서 수신자 말단에 해당하는 홈 게이트웨이의 스케줄러로 적합하다. CQ(Calendar Queue)는 상대적으로 적은 오버헤드를 가지면서 이벤트 셋 문제(event set problem)를 효과적으로 해결할 수 있는 큐잉 (queueing) 알고리즘을 Randy Brown

에 의해 처음으로 제안되었다.

이 방식은 주기적으로 동작하는 큐잉 방법이기 때문에 주기적인 트래픽의 특성을 보존하면서 추가적으로 비주기적인 트래픽을 전송하기에 알맞은 방식이다. 따라서 캘린더 큐를 기반으로 각각의 클래스로 구분되는 트래픽을 효율적으로 전송할 수 있는 스케줄러를 제안하였다. CQ(Calendar Queue)의 하드웨어 구현에 대한 적합한 설계 방법론을 제시한다.

Ⅲ. 제안하는 시스템 구조 및 구현

3.1 CQS(Calendar Queue Scheduler)

그림 1은 CQS의 외부 인터페이스를 보여준다. Host processor는 General CPU로 를 사용하였으며, CQS 내부의 메모리를 엑세스 하는 기능과 동작 상태 관리 및 제어 기능을 수행하게 된다. TSG(Time Slot Generator)는 CQS가 패킷을 전송하기 위해 사용하는 현재 타임 슬롯 정보(CTS)와 전송 가능 패킷 길이 정보(TS_CNT)를 제공하는 기능을 수행한다. Port processor는 패킷이 입력된 경우에는 Enqueue 동작을 수행하고 패킷을 출력하는 경우에는 Dequeue 동작을 수행한다. Port processor는 Enqueue 동작을 수행하기 위해서 CQS_Arbiter 로 요청 신호(EQ_REQS)를 보내고 CQS_Arbiter에서 선택되면 Prot processor 는 CQS로 패킷에 대한 정보를 보내게 된다. Prot processor 는 Dequeue 동작을 수행하기 위해 CQS_Arbiter로 요청신호(DQ_REQS)를 보내고 CQS_Arbiter에서 선택되면 CQS 는 Port processor로 해당 패킷에 대한 정보를 보내게 된다.

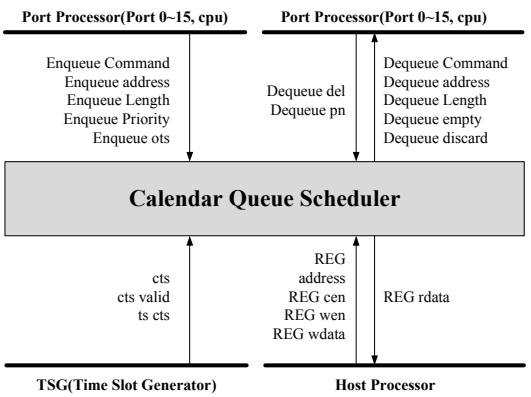


그림 1. Block Diagram for Exterior interface of CQS

그림 2는 CQS의 내부를 구성하는 블록들을 보여준다. CQS는 크게 입력 패킷의 Enqueue를 담당하는 Enqueue Manager와 출력 패킷의 Dequeue를 담당하는 Dequeue Manager, 그리고 패킷 정보 저장을 담당하는 Memory로 구성된다.

Enqueue Manager와 Dequeue Manager는 출력 타임 슬롯에 우선 순위에 따라서 패킷을 전송하기 위하여 우선 순위 별, 출력 타임 슬롯 별로 분류된 Memory 내부의 PTR_CUT(Pointer Counter)와 CQ_TBL(Calendar Queue Table)과 연결된다. 우선 순위 별 기본적으로 설정된 메모리를 사용한 경우에 INT_CQ_TBL을 사용하며, INT_CQ_TBL의 용량이 초과되는 경우에는 부가적인 메모리인 ADD_CQ_TBL을 사용하며, 이를 위한 어드레스를 제공하는 ADD_MEM_USAGE를 사용한다. 또한 Enqueue Manager와 Dequeue Manager는 출력 타임 슬롯에 우선 순위에 따라서 패킷을 전송할 수 없는 경우에 패킷을 폐기하기 위하여 DROP_PTR_CUT와 DROP_CQ_TBL을 이용한다.

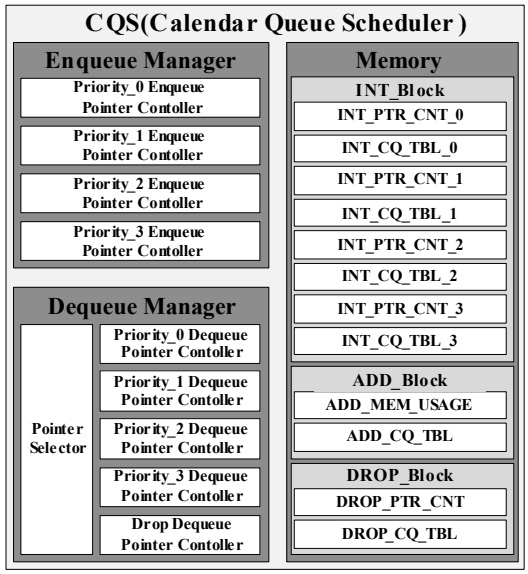


그림 2. CQS Block Diagram

3.2 Enqueue Manager

그림 3은 Enqueue Manager 블록도를 보여 준다. Enqueue Manager는 우선 순위별 패킷 정보를 저장하기 위한 4 개의 우선 순위 별 Enqueue Pointer Controller로 구성되며, 각각 INT_Block의 각 INT_PTR_CUT와 INT_CQ_TBL과 직접 연결되어

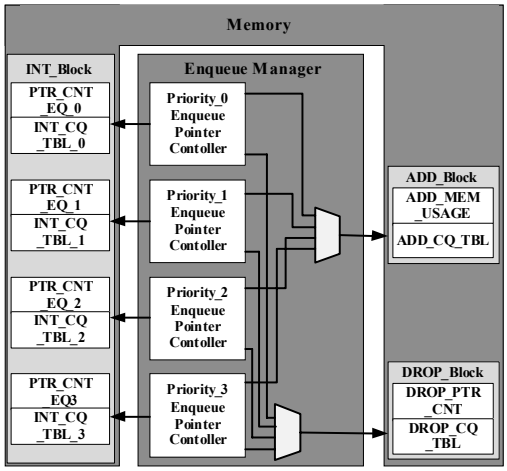


그림 3. Enqueue Manager Block Diagram

있다.

Enqueue Manager의 동작 순서는 다음과 같다.

1) CQS_Arbiter는 Port Processor들로부터 Enqueue를 요청하는 EQ_REQS 신호를 받는다.
2) CQS_Arbiter는 여러 개의 EQ_REQS 신호들 가운데 하나를 선택한 후에 해당 Port Processor로 EQ_SELS 신호를 보낸다.
3) 선택된 Port Processor는 CQS로 패킷이 저장된 주소 정보(eq_pkt_addr), 패킷의 길이 정보(eq_pkt_len), 패킷의 우선 순위 정보(eq_pkt_pri), 패킷의 출력 타임 슬롯 정보(eq_pkt_ots), 그리고 현재 보내고 있는 정보들이 유효한 것임을 알려주는 별도의 신호(eq_cmd)를 보낸다.
4) CQS는 Port Processor로부터 전달받은 패킷의 출력 타임 슬롯 정보(eq_pkt_ots)를 이용하여 비트맵의 LSB를 출력 타임 슬롯 0으로 MSB를 출력 타임 슬롯 2047로 변환하여 패킷 스케줄 테이블의 출력 타임 슬롯 큐를 결정한다.
5) CQS는 Port Processor로부터 전달받은 패킷의 우선 순위 정보(eq_pkt_pri)를 이용하여 비트맵의 LSB를 우선 순위 0으로 MSB를 우선 순위 3로 변환하여 패킷 스케줄 테이블의 패킷 우선 순위 큐를 결정한다.
6) CQS는 Port Processor로부터 전달받은 패킷의 출력 타임 슬롯 정보(eq_pkt_ots), 그리고 패킷의 우선 순위 정보(eq_pkt_pri)를 수신하여 패킷 스케줄 테이블의 우선 순위 큐에 패킷이 저장된 주소 정보(eq_pkt_addr)와 패킷의 길이 정보(eq_pkt_len)를 저장하고 우선 순위 큐의 포인터 카운터(PTR_COUNTER)는 저장된 패킷의 개수만큼 카운트한다.
7) CQS는 특정 우선 순위 큐로 패킷의 정보들을 저장하기 위하여 시도하는 도중에 우선 순위 큐가 모두 Full이 발생한 경우에 DROP_PTR_COUNTER와 DROP_CQ_TBL을 이용하여 패킷을 폐기한다.

그림 4는 EQ_PTR_CTL 블록은 패킷 정보를 이용하여 생성된 메모리 주소 정보를 이용하여 PTR_COUNTER의 카운터 값을 읽는 것을 보여준다. 패킷 우선 순위 별 Queue에는 최대 15 개의 패킷 정보를 저장할 수 있다. 만약에 카운터 값이 15 미만인 경우에는 해당 메모리 주소의 INT_CQ_TBL에 패킷 정보를 저장한다. 만약에 카운터 값이 15 인 경우에는 부가적인 메모리를 사용하기 위하여 ADD_MEM_USAGE의 상태 정보를 읽는다. 만약에 부가적인 메모리를 사용할 수 있는 경우에는 ADD_MEM_USAGE로부터 전달받은 부가적인 메모리의 Entry 주소 정보를 이용하여 추가적으로 패킷 정보를 저장할 수 있다.

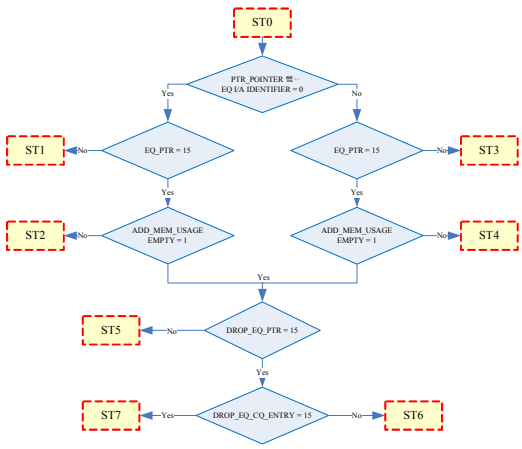


그림 4. Enqueue Pointer Controller State Diagram

3.3 Dequeue Manager

Dequeue Manager는 우선 순위 별 패킷 정보를 추출하기 위한 4 개의 우선 순위 별 Dequeue Pointer Controller와 패킷을 제거하기 위한 1 개의 Drop Dequeue Pointer Controller, 그리고 상기 5 개의 Pointer Controller들로부터 한 개의 패킷 정보를 선택하기 위한 Pointer Selector로 구성된다. DQ_PROCESSOR(Dequeue Processor) 블록은

PTR_COUNTER 블록의 정보를 이용하여 CQ_TBL에 저장된 패킷의 정보(패킷이 저장된 주소 정보, 패킷의 길이 정보)를 Port Processor로 전달하는 기능을 가진다. DQ_PROCESSOR는 패킷의 우선 순위와 패킷의 길이 정보를 이용하여 여러 패킷 정보들 가운데 하나를 선택한다. Dequeue Manager의 동작은 다음과 같다.

1) CQS_Arbiter는 Port Processor들로부터 Dequeue를 요청하는 DQ_REQS 신호를 받는다.
2) CQS_Arbiter는 여러 개의 DQ_REQS 신호들 가운데 하나를 선택한 후에 해당 Port Processor로 DQ_SELS 신호를 보낸다.
3) 선택된 Port Processor는 CQS로 전송될 패킷과 관련된 패킷 스케줄 테이블의 엔트리를 제거하라는 dq_del 신호와 패킷 스케줄 테이블의 목적지 포트 번호를 나타내는 dq_pn 신호를 보낸다.
4) CQS는 선택받은 Port Processor로 패킷이 저장된 주소 정보(dq_pkt_addr), 패킷의 길이 정보(dq_pkt_len), 그리고 현재 보내고 있는 정보들이 유효한 것임을 알려주는 별도의 신호(dq_cmd)를 보낸다.
5) CQS는 패킷 스케줄 테이블에 전송할 패킷이 존재하지 않는 경우에는 메모리가 비워짐을 알리는 dq_mem_empty 신호를 Port Processor로 보낸다.

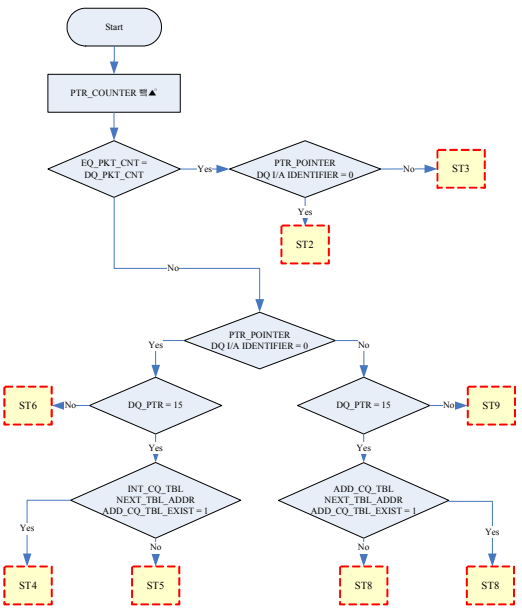


그림 5. Dequeue Pointer Controller State Diagram

6) CQS는 멀티캐스트 패킷인 경우에 동일한 패킷이 저장된 주소 정보(eq_pkt_addr)와 패킷의 길이 정보(eq_pkt_len)들이 멀티캐스트 포트 개수만큼 출력이 이루어 진다.

3.4 Pointer Selector

PTR_SELECTOR(Pointer Selector) 블록은 네 개의 우선 순위 별 PRI_PTR_CTL 블록과 한 개의 DROP_PTR_CTL 블록으로부터 패킷 우선 순위 별 포인터 정보(pkt_addr, pkt_len, valid)들을 전달받는다. PTR_SELECTOR 블록은 TS_COUNTER 블록으로부터 현재 전송할 수 있는 패킷 길이 정보(ts_cnt)를 전달받는다. PTR_SELECTOR 블록은 PRI_PTR_CTL 블록으로부터 전달받은 패킷 우선 순위 별 포인터 정보(pkt_addr, pkt_len, valid)와 TS_COUNTER 블록으로부터 전달받은 현재 전송할 수 있는 패킷 길이 정보(ts_cnt)를 이용하여 Dequeue 할 패킷을 선택한다. PTR_SELECTOR 블록은 Port Processor으로부터 dq_del, dq_pn 신호를 전달받은 후에 패킷 버퍼에 저장된 패킷을 전송하기 위하여 Port Processor에 해당 패킷 정보(dq_cmd, dq_pkt_addr, dq_pkt_len)를 전달한다. PTR_SELECTOR 블록은 Port Processor으로부터 dq_del, dq_pn 신호를 전달받은 후에 패킷 버퍼에 저장된 패킷을 폐기하기 위하여 Port Processor에 해당 패킷 정보(dq_discard, dq_pkt_addr, dq_pkt_len)를 전달한다. PTR_SELECTOR 블록은 PRI_PTR_CTL 블록으로 패킷 정보 전송이 완료되었음을 알리는 신호(select)를 전달한다.

3.5 Dequeue Pointer Controller

PRI_PTR_CTL(Priority Pointer Controller) 블록은 패킷 우선 순위별로 존재한다. PRI_PTR_CTL 블록은 메모리(PTR_COUNTER, ADD_MEM_USAGE)를 초기화 시킨 후에 초기화가 완료되었음을 알려주는 init_done 신호를 PTR_SELECTOR 블록에 전달한다. PRI_PTR_CTL 블록은 현재 타임 슬롯 정보(cts)를 이용하여 가장 우선 순위가 높은 패킷을 검색하고 PTR_SELECTOR 블록에 전달한다. 각각의 PRI_PTR_CTL 블록은 독립적으로 동작하고 항상 가장 높은 우선 순위를 가지는 포인터 정보를 PTR_SELECTOR 블록에 전달하고 있어야 한다. 각각의 PRI_PTR_CTL 블록은 가장 높은 우선 순위를 가지는 포인터의 출력 타임 슬롯(ots)이 현재 타임 슬롯(cts) 보다 같거나 작은 경우에 유효함을

알리는 valid 신호를 전달한다.

3.6 Memory

3.6.1 CQ_TBL

그림 6에서 보면, NoC 소자의 CQS는 미리 정해진 출력 타임 슬롯에 패킷들을 우선 순위에 따라서 전송하기 위하여 캘린더 큐 테이블(Calendar Queue Table, 이하 CQ_TBL)을 관리한다. 캘린더 큐 테이블은 목적지 출력 포트 큐(OUTPUT_PORT_QUEUE), 패킷 우선 순위 큐(PACKET_PRIORITY_QUEUE), 그리고 출력 타임 슬롯 큐(TIME_SLOT_QUEUE)들로 구성된다. 목적지 출력 포트 큐는 패킷을 전송하기 위하여 미리 분류된 패킷의 출력 포트를 나타낸다.

패킷 우선 순위 큐는 패킷을 전송하기 위하여 미리 정해진 패킷의 우선 순위를 나타낸다.

출력 타임 슬롯 큐는 패킷을 전송하기 위하여 미리 정해진 패킷의 타임 슬롯을 나타낸다.

각 패킷 우선 순위 큐 별로 패킷의 저장 어드레스(Packet Pointer)를 최대 15개까지 저장한다.

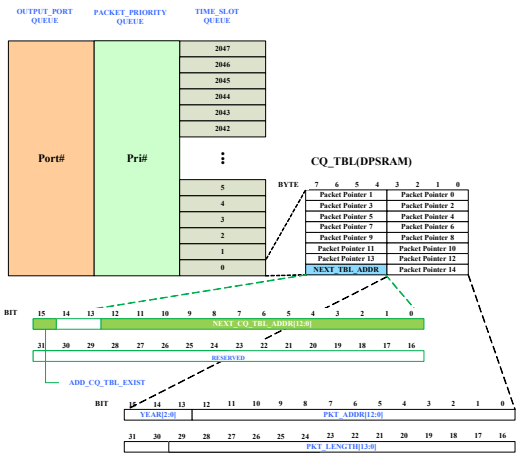
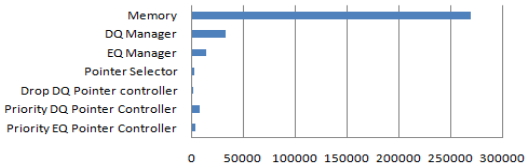


그림 6. Calendar Queue Table(CQ_TBL)

Ⅳ. 결 론

그림 7은 Synopsys Design Compiler를 사용하여 magnachip 0.18 CMOS 라이브러리로 합성하였을 때 NAND(2x1) 게이트(11.09) 기준의 각 모듈의 면적을 나타내고 있다. 비메모리의 경우Priority EQ Pointer Controller, Priority DQ Pointer Controller는 각각 4개의 모듈이 포함되었으며, 메모리의 경우

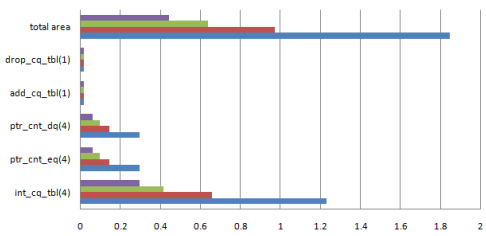


	Priority EQ Pointer Controller	Priority DQ Pointer Controller	Drop DQ Pointer controller	Pointer Selector	EQ Manager	DQ Manager	Memory
Area[Gate]	3118	7234	1528	1878	13324	32798	269352

그림 7. Area for each module(0.18 CMOS technologies)

int_cq_tbl(4k*32) 4개, ptr_cnt_eq(512*32) 4개, ptr_cnt_dq(512*32) 4개 , 나머지 모듈은 1개씩 포함되어 있다. 합성은 제한된 조건 아래서 최악의 케이스로 이뤄졌다. 각 세부 모듈 중 Priority EQ Pointer Controller, Priority DQ Pointer Controller은 우선순위의 개수와 동일하게 개수가 증감하며, Drop DQ Pointer와 Pointer Selector의 경우엔 우선순위에 개수와 상관없이 면적의 크기가 거의 유사하다. DQ_Manager, EQ Manager, Memory를 모두 더하면 전체 CQS 의 area가 되며 Memory의 비중이 전체 CQS에서 85.38%를 나타내고 있음을 알 수 있다. CQS가 속도 제어와 지연 제어에 적합하더라도 가장 큰 문제는 메모리 사이즈가 커진다는 점이다. CQS 내부의 메모리는1year에 해당하는 day의 개수에 따라 메모리size가 증감하며, 우선순위 개수에 따라 메모리 개수가 증감하게 된다.

그림 8은 우선순위가 4개일 때를 가정하여, 1year에 해당하는 day size각각 1k, 2k, 4k, 8k로 변화 되었을 때 각 메모리 싸이즈의 크기를 CACTI 5.3을 통하여 추출한 것이며, 단위는 mm^2이다. 1year가 1k Day일 경우에는 int_cq_tbl의 entry가 1k개가 되며, ptr_cnt_eq와 ptr_cnt_dq의 entry는 각각 1k를 16으로 나눈128가 된다. 동일하게 1year가



	int_cq_tbl(4)	ptr_cnt_eq(4)	ptr_cnt_dq(4)	add_cq_tbl(1)	drop_cq_tbl(1)	total area
1 year = 1k days	0.29356	0.05868	0.05868	0.01467	0.01467	0.44026
1 year = 2k days	0.41516	0.09676	0.09676	0.01467	0.01467	0.63802
1 year = 4k days	0.65488	0.1426	0.1426	0.01467	0.01467	0.96942
1 year = 8k days	1.22892	0.29356	0.29356	0.01467	0.01467	1.84538

그림 8. Area for each memory(CACTI 5.3)

2k Day일 경우에는 int_cq_tbl가 2k*32, ptr_cnt_eq와 ptr_cnt_dq는 256*32가 된다. Add_cq_tbl과 drop_cq_tbl의 싸이즈는 128*32로 고정하여 싸이즈를 추출하였다. Total area를 살펴보면 Day의 싸이즈가 1k, 2k, 4k, 8k로 증가함에 따라 각각 약 1.449, 1.519, 1.906배 증가함을 알수 있다. Memory의 entry가 증가함에 따라 Memory area의 증가폭은 점점 큰폭으로 증가되므로, 1year 에 해당하는 day size의 결정이 전체 CQS area에 절대적은 영향을 미치게 되며, 이와 더불어 우선순위의 개수에 따라 int_cq_tbl, ptr_cnt_eq, ptr_cnt_dq 개수가 선형적으로 증가하게 됨을 알 수 있다.

본 논문에서 CQS를 하드웨어로 설계할 때 각 모듈의 설계 방법론과 각 모듈의 동작에 대하여 논하였다. 또한 CQS 설계시 가장 큰 문제점으로 지적되는 Memory size가 우선순위의 개수와 1year에 해당하는 Day size의 결정이 매우 중요함을 알 수 있다.

참 고 문 헌

〔1〕 R. M. Hinden, "IP Next Generation Overview," *Commun. ACM*, Vol.39, No.6, 1996, pp.61 - 71.

〔2〕 Management Committee Multimedia Communications Forum, Inc., "Final Multimedia Communication Forum (MMCF) Document", 1995.

〔3〕 RANDY BROWN, "Calendar Queues:Daniel Sleator Editor A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem", Communications of the ACM, October 1988 Vol.31 No.10.

〔4〕 SeungHyun Oh and JongSuk Ahn , "Dynamic Calendar Queue".

〔5〕 H. Suzuki, et. Al, "Leading-Zero Anticipatory Logic for High-Speed Floating Point Addition," *IEEE Journal of Solid-State Curcuits*, Vol.31, No. 8, August 1996.

김 진 실 (Jinsil Kim) 정회원

2003년 2월 인하대학교 전기공학과

2003년 2월~현재 삼성전자 디지털프린팅사업부 연구원

2008년 3월~현재 연세대학교 전기전자공학과 석사과정

<관심분야> 네트워크 프로세서, 컴퓨터 아키텍처

정 원 영 (Won-young Chung) 정회원

2005년 8월 연세대학교 전기전자공학과 졸업

2005년 9월~현재 연세대학교 전기전자공학과 석박통합과정

<관심분야> 네트워크 프로세서, 컴퓨터 아키텍처

이 정 희 (Jung-hee Lee) 정회원

1984년 2월 경북대학교 전자공학과 학사

1990년 2월 경북대학교 전자공학과 석사

1984년 3월~현재 한국전자통신 연구원 광대역통합망 연구원

<관심분야> 네트워크 자원관리, 인터넷 QoS

이 용 석 (Yong-Surk Lee) 정회원

1973년 2월 연세대학교 전기공학과 학사

1977년 2월 University of Michigan, Ann Arbor 석사

1981년 2월 University of Michigan, Ann Arbor 박사

1993년~현재 연세대학교 전기전자공학과 교수

<관심분야> 마이크로프로세서, 네트워크 프로세서, 암호화 프로세서, SoC