논문 2015-10-41

SoC FPGA 기반 실시간 객체 인식 및 추적 시스템 구현

(An Implementation of SoC FPGA-based Real-time Object Recognition and Tracking System)

김 동 진, 주 연 정, 박 영 석*
(Dong-Jin Kim, Yeon-Jeong Ju, Young-Seak Park)

Abstract : Recent some SoC FPGA Releases that integrate ARM processor and FPGA fabric show better performance compared to the ASIC SoC used in typical embedded image processing system. In this study, using the above advantages, we implement a SoC FPGA-based Real-Time Object Recognition and Tracking System. In our system, the video input and output, image preprocessing process, and background subtraction processing were implemented in FPGA logics. And the object recognition and tracking processes were implemented in ARM processor-based programs. Our system provides the processing performance of 5.3 fps for the SVGA video input. This is about 79 times faster processing power than software approach based on the Nios II Soft-core processor, and about 4 times faster than approach based the HPS processor. Consequently, if the object recognition and tracking system takes a design structure combined with the FPGA logic and HPS processor-based processes of recent SoC FPGA Releases, then the real-time processing is possible because the processing speed is improved than the system that be handled only by the software approach.

Keywords : SoC FPGA, HPS, Nios II processor, Real-Time, Object recognition, Tracking

Ⅰ. 서 론

불과 몇 년 전까지만 해도 임베디드 프로세서 시장은 PowerPC, RISC, MIPS, SPARC 프로세서가 서로 더 넓은 시장을 차지하기 위해서 경쟁하는 매우 분열적인 시장이었다. 하지만 시장이 성숙해지고 전문화되면서 ARM 프로세서가 임베디드 시스템의 지배적인 플랫폼이 되었다. 따라서 표준 제품에서부터 소프트 ARM IP 코어, 프로그래머블 로직 및 ASIC으로 구현하는 하드 IP에 이르기까지 갈수록 더 다양한 ARM 기반 솔루션들이 등장하고 있다[1].

*Corresponding Author(yspark@kyungnam.ac.kr)
Received: 30 Apr. 2015, Revised: 21 May 2015, Accepted: 6 July 2015.
D.J. Kim: ETRI
Y.J. Ju, Y.S. Park: Kyungnam University
※ 본 논문은 2014년도 경남대학교 학술연구 장려금 지원에 의한 것임.

ⓒ IEMEK J. Embed. Sys. Appl. 2015 Dec. 10(6) 363-372
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2015.10.6.363

하지만 주요 임베디드 시스템의 구현 옵션들이 첨단 디자인의 요구를 충족하기에는 부족함을 드러내고 있다. 멀티 칩(Multi chip) 솔루션은 비교적 구현하기가 용이하나 비용이 높고, 많은 경우에 설계자들이 필요로 하는 유연성과 성능 그리고 저전력 요구에 부응하지 못한다. 소프트 프로세서 코어를 채택하는 단일 칩 솔루션 역시 구현하기에는 비교적 용이하나 성능에 있어서 한계가 있다. 하드 ARM 코어를 채택한 ASIC SoC는 전력, 성능, 최적화는 뛰어나지만 개발 시간 때문에 제품 출시 시간이 길어지고, 유연성이 떨어지며, 대다수 애플리케이션에서 이용하기에는 비용이 너무 높다[2].

앞서 언급하였듯이 현재까지 개발되어 사용되고 있는 임베디드 영상처리 시스템들은 대부분이 ARM 프로세서를 기반으로 한다. 이러한 시스템은 PC 기반에 비해 시스템 처리 성능이 낮기 때문에 복잡한 알고리즘을 수행할 경우 처리 속도가 떨어진다.

하지만 최근 출시된 SoC FPGA는 ARM 프로세

서와 FPGA 패브릭을 통합하고 있어 통상적인 임베디드 영상처리 시스템에서 사용되는 ASIC SoC에 비해 우월한 성능을 구현할 수 있다.

본 연구에서는 SoC FPGA 기반 실시간 객체 인식 및 추적 시스템을 구현하여 Nios II 소프트코어 프로세서 및 ASIC SoC에 비해 SoC FPGA의 우월한 처리 성능을 확인하고자 한다. 영상 입출력과 전처리 과정, 배경 차분은 FPGA 논리로 구현하였고, 객체 인식 및 추적은 ARM 프로세서 기반의 응용 프로그램으로 구현하였다.

Ⅱ. 객체 인식 및 추적 알고리즘

1. 배경 차분을 이용한 객체 검출[3]

움직이는 객체를 검출하는 방법 중에서 많이 사용되는 방법은 연산이 간단한 배경 차분 방법이다. 배경 차분 방법(Background Subtraction)은 이전에 생성된 배경 영상과 현재 영상을 비교하여 차이점이 있다면 그 부분을 검출하는 방법이다. 이러한 배경 차분 방법은 화소 기반의 연산으로 식 (1)과 같이 표현할 수 있다.

$$D_k(x,y) = \begin{cases} 1 & \text{if } |I_k(x,y) - B_k(x,y)| > T \\ 0 & otherwise \end{cases}, \quad (1)$$

여기서 Ik(x,y)는 k번째 입력 영상에서 (x,y) 화소에 대한 값이며, Bk(x,y)는 k번째 배경 영상에서의 (x,y) 화소에 대한 값이다. Dk(x,y)는 k번째 배경 차분 영상의 (x,y) 화소에 대한 값이다.

2. 영상의 잡음 제거[3]

배경 차분 방법으로 영상을 배경과 객체로 분리하게 되면, 객체가 아님에도 객체로 인식되는 잡음이 나타난다. 이 잡음을 처리하기 위해 모폴로지(Morphology) 연산을 사용하는데, 그 중에서도 침식(Erosion)과 팽창(Dilation) 연산을 사용하였다.

침식은 영상 내에 있는 배경 위의 객체들을 일정하게 감소하는 방법이다. 이 알고리즘을 사용할 경우, 객체의 영역에서 배경과 접한 부분이 침식되며, 이는 물체의 크기를 축소시키고, 배경의 크기는 확장시키는 역할을 하게 된다. 한 화소 단위의 작은 잡음들은 효과적으로 제거되며 연결되어 있는 물체를 분리한다. 2차원 영상 A와 구조적 요소 B에 대한 침식연산은 다음과 같이 식 (2)로 정의할 수 있다.

$$A \ominus B = \{x | B_x \subseteq A\} \quad (2)$$

식 (2)에서 x는 화소를, A는 2차원 입력 영상, Bx는 화소 x에 대한 구조요소를 의미한다. 여기서 입력 영상 A는 객체가 1, 배경이 0인 값으로 구성된다. 입력 화소 x의 구조요소 B가 입력 영상 A의 객체에 부분집합이거나 동일하면, 출력 화소 x의 값은 1이 되고, 그 외의 출력 화소 x는 0이 된다.

팽창은 영상 내에 있는 배경 위의 객체들을 일정하게 확대하는 방법이다. 구조적 요소를 사용하여 물체 주변을 따라 화소를 확대한다. 객체의 영역에서 배경과 접한 부분이 팽창되며, 이는 물체의 크기를 확대시키고, 배경의 크기는 축소시키는 역할을 하게 된다. 물체 안의 홀과 같은 빈 공간을 메우거나 짧은 거리만큼 떨어진 영역을 연결시킨다.

2차원 영상 A와 구조적 요소 B에 대한 팽창 연산은 식 (3)으로 정의할 수 있다.

$$A \oplus B = \{x | (\hat{B})_x \cap A \neq \varnothing\} \quad (3)$$

식 (3)에서 x는 화소를, A는 2차원 입력 영상, Bx는 화소 x에 대한 구조요소를 의미한다. 여기서 입력 영상 A는 객체가 1, 배경이 0인 값으로 구성된다. 입력 화소 x의 반전된 구조 요소 $\widehat{B_x}$와 입력 영상 A의 교집합이 공집합이 아니라면, 출력 화소는 1이 되며, 공집합이라면 출력 화소 x는 0이 된다.

3. 객체 분할[4]

분할은 배경에서 객체를 추출하는 것을 뜻한다. 영상에서 관심 영역은 물체가 되고 나머지는 배경으로 취급된다. 객체 분할을 위해 라벨링(labeling)을 이용하였다. 라벨링은 인접하여 연결된 모든 화소에 동일한 번호를 붙이고 다른 연결성분에는 다른 번호를 붙이는 영역 구분 방법이다. 특히 이진영상의 경우에는 전경 값(foreground value) 화소 x에 대해 x와 인접한 화소들의 집합을 N(x)라 할 때 x와 연결된 화소 p들의 집합 C(x)는 식 (4)와 같이 나타낼 수 있다. 여기에서 F는 전경 값을 갖는 화소들의 집합을 의미한다.

$$C(x) = \{p | p \in N(x) \text{ and } p \in F\} \quad (4)$$

집합 N(x)는 인접화소를 어떻게 정의하는가에 의존하게 되는데, 4방향을 고려하는 것과 8방향을 고려하는 방법이 있다.

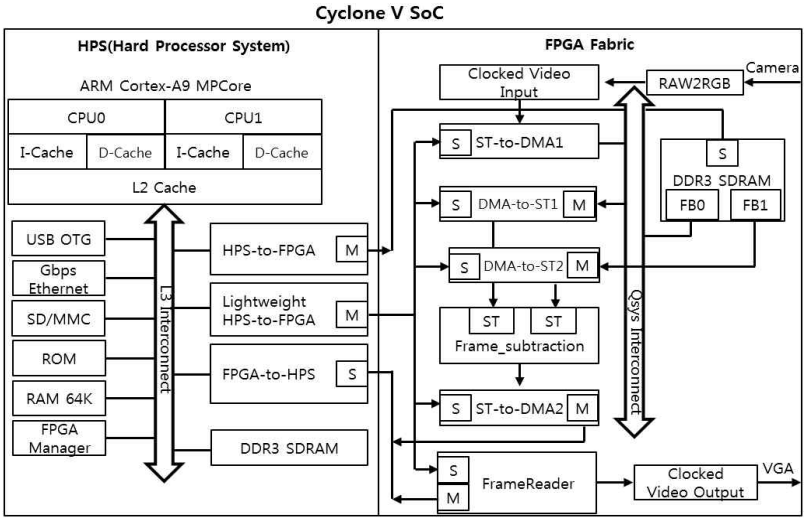

그림 1. SoC FPGA 시스템의 블록도
Fig. 1 Block diagram of SoC FPGA system

라벨링에서는 8방향에 대해 연결을 조사하는 것이 보통이다. 한 화소를 둘러싼 8개의 모든 화소들과의 연결성을 조사해 봐야 한다. 이때 화소들 간의 연결성은 식 (5)와 같이 쌍방향적이다. 즉, 화소 x가 화소 y에 연결되어 있다면 y도 x에 연결되어 있다는 것을 의미한다.

$$x \in C(y) \Leftrightarrow y \in C(x) \tag{5}$$

Ⅲ. SoC FPGA 기반 임베디드 시스템 설계

본 장에서는 SoC FPGA 기반 실시간 객체 인식 및 추적 시스템을 구현하고자 한다. 그림 1은 시스템 전체 블록도로서 HPS(Hard Processor System)[5] 영역과 FPGA 영역으로 구성된다. FPGA 영역의 RAW2RGB 모듈은 카메라 영상을 RGB 영상으로 변환하여 Qsys의 Clocked Video Input 모듈로 전송한다. 수신된 영상은 Avalon Streaming 인터페이스[6]를 통해 ST-to-DMA1 모듈로 전송되어 DDR3 SDRAM에 저장된다. 저장된 배경영상과 현재영상은 DMA-to-ST1 모듈과 DMA-to-ST2 모듈에 의해 Frame-Subtraction 모듈로 전송되며, Frame-Subtraction 모듈에서는 Avalon Streaming 인터페이스로 배경영상과 현재영상을 수신한 후 프레임 차를 수행하여 결과 영상을 ST-to-DMA2 모듈로 전송한다. ST-to-DMA2 모듈은 결과 영상을 FPGA-to-HPS 브릿지(Bridge)를 통해 HPS로 전송한다. HPS에서는 FPGA로부터 수신된 프레임 차 영상을 리눅스 기반 디바이스 드라이버를 통해 DDR3 SDRAM에 저장한다. 저장된 영상은 리눅스 어플리케이션에서 객체 인식 및 추적 알고리즘을 수행한 후 프레임버퍼 디바이스 드라이버와 Lightweight HPS-to-FPGA 브릿지(Bridge)를 통해 FPGA의 FrameReader 모듈로 전송되고, 마지막으로 Clocked Video Output 모듈을 통해 VGA로 출력된다.

1. SoC FPGA 시스템의 하드웨어 설계

본 절에서는 Quartus II 설계 소프트웨어[7]를 사용하여 SoC FPGA 기반 실시간 객체 인식 및 추적 시스템을 위한 영상 입·출력과 전처리 과정, 배경 차분을 FPGA 논리로 구현하고자 한다.

(1) ST-to-DMA 모듈

ST-to-DMA 모듈은 카메라의 RGB 영상을 Avalon Streaming Interface로 수신한 후 외부 메모리에 저장하는 기능을 담당하는 DMA이다. 그림 2는 ST-to-DMA 모듈의 블록도이다. RGB 데이터는 Avalon Streaming Interface(st_in_data, st_in_sop, st_in_eop, st_in_valid, st_in_ready)로 수신되며, st_in_valid(유효 데이터 판별 신호)와 st_in_ready (데이터 수신 준비 신호)가 '참'일 때 scfifo에 저장된

다. st_in_ready는 scfifo의 usedw 신호에 의해 참, 거짓이 판별된다. scfifo에 저장된 RGB 데이터는 increment_address가 '참'일 때 Avalon-MM Master에 의해 외부 메모리에 저장된다. Avalon-MM Master는 데이터를 burst 방식으로 Slave에 전송하며, 버스로부터 제어권을 확보 했을 때 일정한 양의 데이터를 한꺼번에 전송함으로써 버스 대역폭을 효율적으로 사용한다. DMA address generator는 control_go(DMA 시작 신호), write_length(전송하고자 하는 데이터 크기), increment_address(master address 증가 신호), fb_address(저장하고자 하는 메모리 주소) 신호를 이용하여 length(전송 길이)와 control_done(전송 완료 신호), m_address(avalon-mm master address) 신호를 생성한다. control logic은 control_done 신호를 이용하여 control_go 신호를 생성한다. DMA burst counter는 control_go, m_waitrequest(Avalon-MM Master의 전송 대기 신호), length, usedw(현재 fifo에 저장된 데이터의 양)에 의해 m_burstcount(Avalon-MM Master의 burstcount 신호)를 생성한다. Avalon-MM Slave는 s_address(Avalon-MM Slave address), s_byteenable(Avalon-MM Slave byteenable), s_write(Avalon-MM Slave write), s_writedata(Avalon-MM Slave data) 신호를 이용하여 write_length(전송하고자하는 데이터 크기), fb_address(영상을 저장하고자 하는 메모리 주소) 신호를 생성한다.

(2) DMA-to-ST 모듈

DMA-to-ST 모듈은 외부 메모리에 저장된 영상을 Avalon Streaming Source Interface로 전송하기 위한 DMA이다. 그림 3은 DMA-to-ST 모듈의 블록도이다. Avalon-MM Master는 버스 대역폭을 효율적으로 사용하기 위해 Slave에 저장된 데이터를 burst 방식으로 읽어온 후 m_readdatavalid가 '참'일 때 scfifo에 저장한다. DMA address generator는 control_go(DMA 시작 신호), read_length(전송하고자 하는 데이터 크기), increment_address(주소 증가 신호), fb_address(읽기위한 데이터가 저장된 메모리 주소) 신호를 이용하여 length(읽고자하는 데이터 길이)와 control_done(전송완료신호), m_address (Avalon MM Master address) 신호를 생성한다. control logic은 control_done 신호를 이용하여 control_go 신호를 생성한다. DMA burst counter는 control_go, m_waitrequest (Avalon-MM Master의 전송 대기 신호), length, usedw(현재 fifo에 저장된 데이터의 양)에 의해 m_burstcount(Avalon MM Master의 burstcount 신호)를 생성한다. Avalon-MM Slave는 s_address(Avalon-MM Slave address), s_byteenable(Avalon-MM Slave byteenable), s_write(Avalon-MM Slave write), s_writedata(avalon-mm slave data) 신호를 이용하여 write_length(읽고자 하는 데이터 크기), fb_address(영상을 저장하고자 하는 메모리 주소) 신호를 생성한다. scfifo에 저장되어 있는 RGB 데이터는 st_out_ready와 st_out_valid가 '참'일 때 scfifo로부터 출력되어 Avalon Streaming Source Interface로 전송된다. Streaming Interface Signal Generator는 sop(start of packet)과 eop(end of packet) 신호를 생성한다.

(3) Frame-Subtraction 모듈

Frame-Subtraction 모듈은 현재 영상과 배경 영상을 수신하여 차 영상을 생성한다. 그림 4는 Frame-Subtraction 모듈의 블록도이다. 현재 영상과 배경 영상을 각각 2개의 Avalon Streaming Sink Interface로 수신하여 프레임 차를 수행한 후 결과 영상을 Avalon Streaming Source Interface로 전송한다. Avalon Streaming Sink Interface로 수신되는 현재 영상과 배경 영상은 st_in_ready와 st_in_valid가 '참'일 때 scfifo1과 scfifo2에 저장된다. scfifo1과 scfifo2에 저장되어 있는 영상은 st_out_valid와 st_out_ready가 '참'일 때 각각의 scfifo로부터 출력되어 Image Processing 로직에서 RGB2GRAY와 프레임 차를 수행한 후 Avalon Streaming Source Interface로 전송된다. st1_in_ready generator와 st2_in_ready generator는 fifo1_usedw와 fifo2_usedw 신호를 이용하여 st1_in_ready, fifo1_read, st2_in_ready, fifo2_read 신호를 생성한다. Streaming Interface Signal Generator는 sop(start of packet)과 eop(end of packet) 신호를 생성한다.

2. SoC FPGA 시스템의 소프트웨어 설계

본 절에서는 SoC FPGA 시스템의 소프트웨어를 설계하고자 한다. 그림 5는 SoC FPGA 시스템의 소프트웨어 블록도로서 FPGA에서 수행된 프레임 차 영상은 FPGA-to-HPS 브릿지(Bridge)를 통해 리눅스 디바이스 드라이버(/dev/fb0)를 통해 HPS의 DDR3 SDRAM에 저장된다. 저장된 프레임 차 영상은 리눅스 어플리케이션에서 객체 인식 및 추적 알

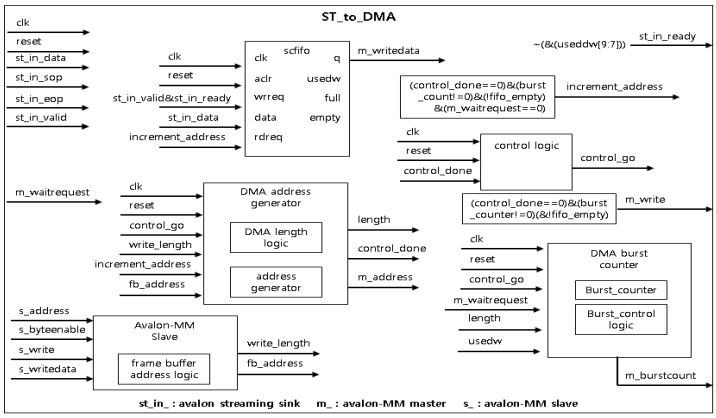

그림 2. ST-to-DMA 모듈의 블록도

Fig. 2 Block diagram of ST-to-DMA

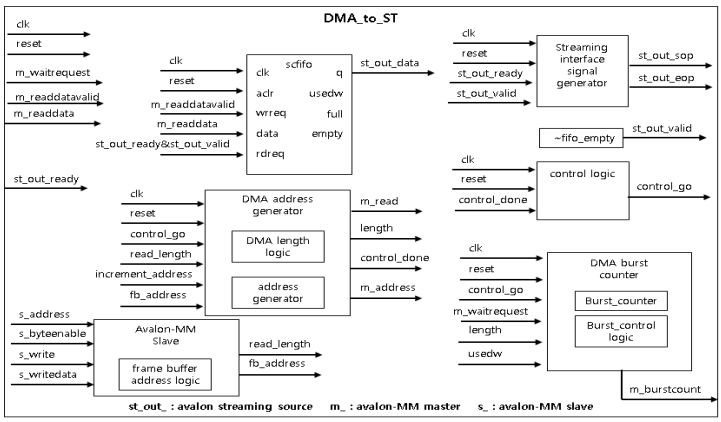

그림 3. DMA-to-ST 모듈의 블록도

Fig. 3 Block diagram of DMA-to-ST

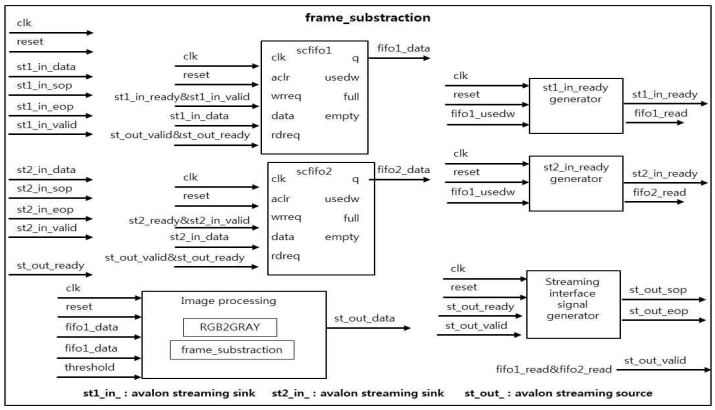

그림 4. Frame-Subtraction 모듈의 블록도

Fig. 4 Block diagram of frame_subtraction

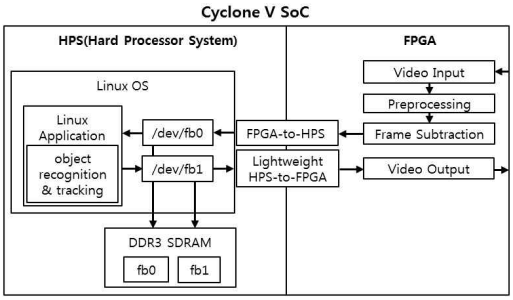

그림 5. SoC FPGA의 소프트웨어 블록도
Fig. 5 Software block diagram of SoC FPGA

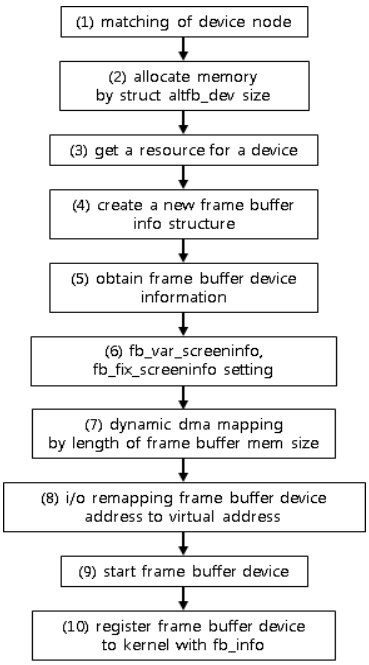

그림 6. 프레임버퍼의 흐름도
Fig. 6 Flow-chart of framebuffer

고리즘을 수행한 후 "/dev/fb1"을 통해 DDR3 SDRAM에 저장된다. 결과 영상은 Lightweight HPS-to-FPGA 브릿지(Bridge)를 통해 FPGA의 FrameReader에 의해 VGA로 출력된다.

그림 6은 영상 입·출력을 위한 디바이스 드라이버의 흐름도이다. (1) 디바이스 노드 리스트로부터 매칭되는 디바이스 노드가 있는지 확인하고, (2) fb_dev 구조체 크기만큼 메모리를 할당하고, 시작 주소를 fbdev에 할당한다. (3) platform device에 대한 리소스를 획득하고, (4) 프레임 버퍼 info 구조체를 생성한다. (5) 프레임 버퍼 info 구조체 정보를 획득하고 (6) fb_var_screeninfo, fb_fix_screeninfo 구조체 변수를 설정한다. (7) 프레임 버퍼 메모리 크기의 길이만큼 동적 할당을 하고 (8) 프레임 버퍼 디바이스 주소를 가상주소에 맵핑한다. (9) 프레임 버퍼 디바이스를 시작한다. (10) 프레임 버퍼 디바이스를 fb_info와 함께 커널에 등

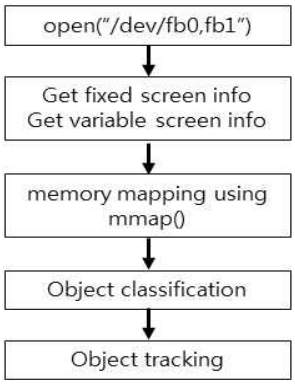

그림 7. 객체추적 프로그램의 흐름도
Fig. 7 Flow-chart of object tracking program

표 1. 실험 환경
Table 1. Experiment environment

Category	Specifications
FPGA Design Software	► Quartus II 13.1 ► Nios II EDS 13.1 ► Modelsim-Altera 10.0d ► SoC EDS 13.1
SoCKit Development Board	► FPGA Device ▪ Cyclone V SX SoC-5CSXFC6D6F31C6N ► ARM-based hard processor system(HPS) ▪ 800MHz, A Dual-Core ARM Cortex-A9 MPCore Processor
ASIC SoC	► Raspberry Pi 2 Model B ▪ Broadcom BCM2836 SoC ▪ Quad-core ARM Cortex-A7 900MHz ▪ Dual Core VideoCore IV @ Multimedia Co-Processor ▪ 1GBLPDDR2
PC	► Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz ► 8.00GB RAM

록한다.

그림 7은 FPGA에서 수행된 프레임 차 영상을 수신하여 객체 분류 및 추적을 위한 응용 프로그램의 흐름도이다. FPGA에서 수행된 프레임 차 영상은 FPGA-to-HPS 브릿지를 통해 HPS로 전송된다. 수신된 차 영상을 획득하기 위해 디바이스 파일(“dev/fb0, fb1")을 열고 프레임 버퍼의 가변 정보와 고정 정보를 획득한다. mmap() 함수를 이용하여 차 영상이 저장된 디바이스 파일에 대응하는 영역의 주소를 획득한다. 획득한 프레임 버퍼의 주소를 이용하여 객체를 분류하고, 추적하는 알고리즘을 수행한 후 영상 출력을 위한 디파이스 파일에 저장한다.

Ⅳ. 실험결과 및 고찰

1. 실험환경

본 연구의 실험을 위한 시스템 구성은 표 1과 같이 Altera Cyclone V SoC 디바이스가 장착된 SoCKit 개발 보드와 ASIC SoC 개발 보드는 Raspberry 2 Model B를 사용하였으며, 인텔 i7-2630QM 프로세서와 8GB RAM이 장착된 PC를 사용하였다.

2. RTL 시뮬레이션 검증

구현회로 검증은 윈도우 7 OS 환경에서 Modelsim-Altera 시뮬레이션 소트프웨어[8]를 사용하여 RTL 회로를 검증하였다.

그림 8은 ST-to-DMA 모듈의 시뮬레이션 결과이다. control_write_go 신호가 ‘참’일 때 Avalon Memory-Mapped Master는 Avalon Memory Mapped Slave로 데이터 전송을 시작한다. 이때 length 신호가 전송하고자 하는 전체 영상의 크기에서 0으로 변했을 때 control_done 신호가 ‘참’이 되어 데이터 전송 완료를 나타낸다. master address 신호는 사용자 어플리케이션으로부터 수신된 control_write_base(데이터를 전송하고자 하는 메모리의 주소) 신호와 length(전체 영상 길이) 신호에 의해 생성된다. master_write 신호는 control_done 신호가 ‘거짓’이고, burst_counter 신호가 ‘참’일 때 ‘참’이 된다.

그림 9는 DMA-to-ST 모듈의 시뮬레이션 결과이다. Avalon Memory-Mapped Master는 fifo_full 신호가 ‘거짓’이고, master_readdatavalid 신호가 ‘참’일 때 Avalon Memory-Mapped Slave로부터 데이터를 읽어온 후 scfifo에 저장한다. scfifo에 저장된 데이터는 stream_ready 신호와 stream_valid 신호가 참일 때 Avalon Streaming Source의 stream_data 통해 Avalon Streaming Sink로 전송된다. stream_startofpacket 신호는 Master의 상태가 ‘idle' 상태이고 reading_first_pixel_in_frame (pixel_address 신호로부터 생성) 신호가 ‘참’일 때 ‘참’이 된다. stream_endofpacket 신호는 reading_last_pixel_in_frame(pixel_address 신호로부터 생성) 신호가 ‘참’일 때 ‘참’이 된다. stream_valid 신호는 fifo_empty(scfifo의 empty 신호) 신호가 ‘거짓’일 때 ‘참’이 된다. stream_ready 신호는 Avalon Streaming Sink가 Avalon Streaming Source로부터 데이터 수신 준비가 되었음을 나타내는 신호이다.

그림 10은 Frame-Subtraction 모듈의 시뮬레이션 결과이다. stream_in_*와 av_stream_in_*는 Avalon Streaming Sink 인터페이스이다. stream_in_ready 신호와 stream_in_valid 신호가 ‘참’일 때 fifo1에 저장되고, av_stream_in_ready 신호와 av_stream_in_valid 신호가 ‘참’일 때 fifo2에 저장된다. fifo1과 fifo2에 저장된 데이터는 fifo1_usedw와 fifo2_usedw에 의해 생성된 fifo_read 신호에 의해 출력된 후 그레이 영상으로 변환되어 프레임 차를 수행하여 stream_out_data로 출력된다. stream_out_valid 신호는 fifo_read 신호와 동일하며, stream_out_startofpacket 신호와 stream_out_endofpacket 신호는 fifo_read 신호가 ‘참’이고 stream_out_ready 신호가 ‘참’일 때 카운트 되는 값에 의해 생성된다.

3. 성능평가 및 고찰

그림 11은 실내 환경에서 실험한 객체추적 결과 영상이다. 좌상단은 배경영상, 우상단은 현재영상, 좌하단은 배경차영상, 우하단은 3개의 물체를 추출 및 추적한 객체추적 결과 영상이다.

표 2는 처리구조별 성능 평가표로서 VGA와 SVGA 해상도를 적용하여 실험하였다. VGA 영상의 경우 Nios II 프로세서 기반 응용 프로그램으로 객체 표시까지 수행하였을 경우 초당 0.1 프레임의 처리 성능을 제공하였다. HPS 기반 응용 프로그램으로 객체 표시까지 수행하였을 때에는 초당 1.8 프레임의 처리성능을 제공하였다. ASIC SoC 기반 응용 프로그램으로 객체 표시까지 수행하였을 경우 초당 1.49 프레임의 처리성능을 제공하였다. SoC FPGA(HPS+FPGA) 기반 객체 인식 및 추적 시스

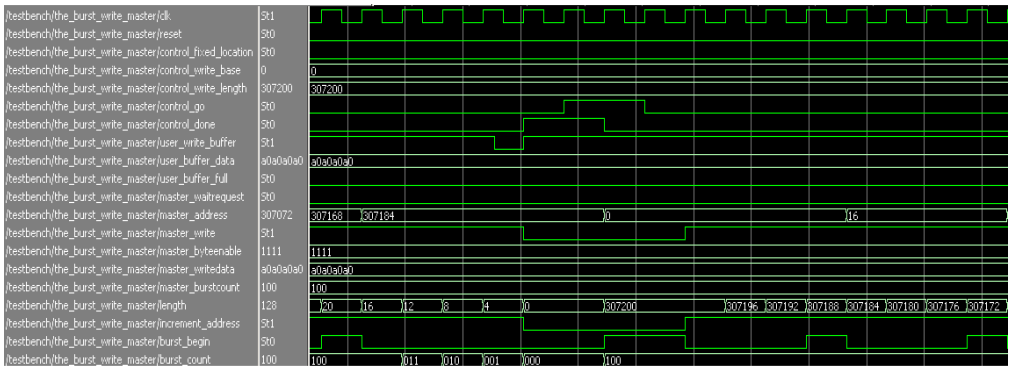

그림 8. ST-to-DMA 모듈의 시뮬레이션 결과

Fig. 8 Simulation result of ST-to-DMA module

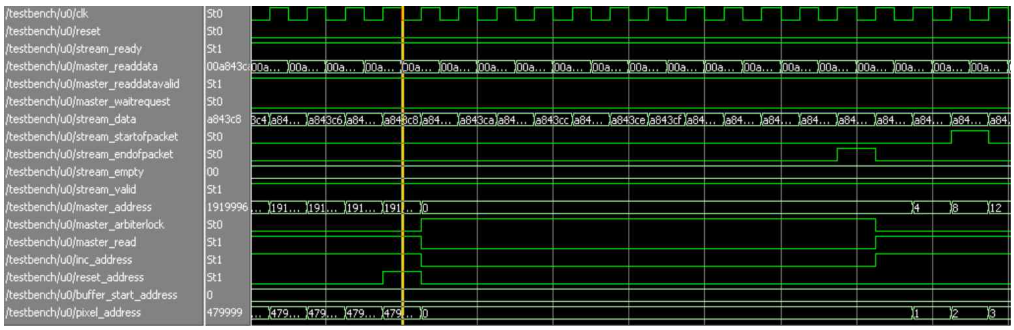

그림 9. DMA-to-ST 모듈의 시뮬레이션 결과

Fig. 9 Simulation result of DMA-to-ST module

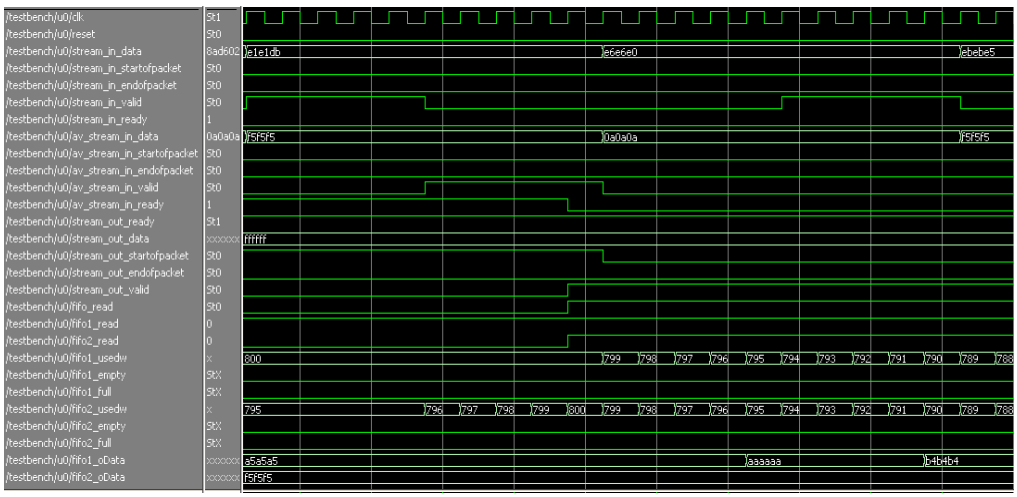

그림 10. Frame-Subtraction 모듈의 시뮬레이션 결과

Fig. 10 Simulation result of Frame-Subtraction module

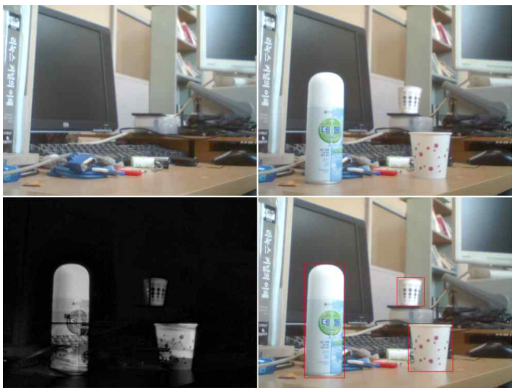

그림 11. 객체추적 결과 영상
Fig. 11 Result image of object tracking

표 2. 성능 평가
Table 2. The performance evaluation

Classification	Nios II Soft-core @100MHz (Software)	HPS (Software)	ASIC SoC	HPS+FPGA (Hardware+Software)
Resolution	VGA	VGA	VGA	VGA
Preprocessing	1	9	25	60
Object Detection	0.5	7	16.12	60
Erosion Dilation	0.125	3.4	2.04	60
Object Classification	0.111	2	1.67	15
Object Display	0.1	1.8	1.49	7.5
Resolution	SVGA	SVGA	SVGA	SVGA
Preprocessing	0.667	8	16.66	60
Object Detection	0.286	6	10.52	60
Erosion Dilation	0.08	3	1.39	60
Object Classification	0.0714	1.4	1.15	10
Object Display	0.067	1.2	1.09	5.3

템의 경우에는 초당 7.5 프레임의 처리성능을 제공하였다. 이는 Nios II 프로세서 기반 응용 프로그램으로 처리한 경우보다 약 75배, HPS 기반 응용 프로그램으로 처리하였을 경우보다 약 4.2배 그리고 ASIC SoC 기반 응용 프로그램으로 처리하였을 경우보다 약 4.5배 빠른 처리 성능을 확인하였다.

또한 SVGA 영상의 경우 Nios II 프로세서 기반 응용 프로그램으로 수행하였을 경우 초당 0.0667 프레임의 처리 성능을 제공하였다. HPS 기반 응용 프로그램으로 수행했을 경우에는 초당 1.2 프레임의 처리 성능을 제공하였다. ASIC SoC 기반 응용 프로그램으로 객체 표시까지 수행하였을 경우 초당 1.09 프레임의 처리성능을 제공하였다. SoC FPGA(HPS+FPGA) 기반 시스템의 경우 초당 5.3 프레임의 처리 성능을 제공하였다. 이는 Nios II 프로세서 기반 응용 프로그램과 비교하였을 때 약 79배, HPS 기반 응용 프로그램과 비교하였을 때 4.4배 그리고 ASIC SoC 기반 응용 프로그램으로 처리하였을 경우보다 약 4.9배 빠른 처리 성능을 확인하였다.

V. 결 론

최근 FPGA 제조업체들은 FPGA와 최적화된 ARM 프로세서를 결합한 새로운 유형의 SoC 디바이스를 출시하였다. ARM 기반 SoC FPGA는 하드 ARM 프로세서, 메모리 컨트롤러, 다양한 주변장치 그리고 FPGA 패브릭을 단일 SoC에 통합함으로서 설계자들이 직면한 많은 기술적 과제들을 해결하면서 전례가 없는 제품 차별화, 가격 및 성능 최적화, 타임-투-마켓 효율성, 제품 수명 장기화 등을 제공한다.

본 연구에서는 SoC FPGA 기반 실시간 객체 인식 및 추적 시스템을 구현하였다. 영상 입·출력과 전처리 과정, 배경 차는 FPGA 논리로 구현하였으며, 객체 인식 및 추적은 ARM 프로세서 기반의 응용 소프트웨어로 구현하였다. 구현된 시스템은 SVGA 영상을 초당 5.3 프레임의 처리성능을 제공하였다. 이는 Nios II Soft-core 프로세서 기반 응용 소프트웨어로 처리하였을 경우보다 약 75배 그리고 HPS 기반 응용소프트웨어로 처리하였을 경우보다 약 4배 빠른 처리 성능을 확인하였다.

References

[1] Todd Koelling, User-customized ARM-based SoC FPGA for Next-Generation Embedded

Systems, Embedded News, Dec., 2011 (in Korean).
[2] E.J. Hwang, User-Customized ARM-based SoC, Embedded News, July, 2012 (in Korean).
[3] SangHaifeng, XuChao, "Moving Object Detection Based on Background Subtraction of Block Updates," IEEE International Conference on Intelligent Networks and Intelligent Systems, pp.51-54, 2013.
[4] K.I. Kim, "Binary Connected-component Labeling with Block-based Labels and a Pixel-based Scan Mask", Journal of The Institute of Electronics Engineers of Korea, Vol, 50, No. 5, pp.1307-1314, 2013 (in Korean).
[5] Cyclone V Hard Processor System Technical Reference Manual, pp.1-3323, 2014.
[6] Altera Corp., Avalon Interface Specifications, pp.1-58, 2015.
[7] Altera Corp., Quartus II Handbook, pp.1-1767, 2014
[8] Altera Corp., Using ModelSim to Simulate Logic Circuits for Altera FPGA Devices, pp.1-30, 2011.

Dong-Jin Kim (김 동 진)

In 2007 and 2009, respectively B.S. and M.S. degrees in Dept. of Information and Communication Engineering from Kyungnam University. In 2011, Ph.D. degree in Advanced Eng. from Kyungnam University. From 2012, Currently Senior Research Engineer in Electronics and Telecommunications Research Institute.
Email : rivaldo2000@etri.re.kr

Yeon-Jeong Ju (주 연 정)

In 2009 and 2015, respectively B.S. and M.S. degrees in Dept. of Information and Communication Engineering from Kyungnam University. From 2015, Currently Research Engineer in SCHNTechnology Inc.
Email : inyean14@naver.com

Young-Seak Park (박 영 석)

In 1979, B.S. degree in Electronic Engineering from Youngnam University. In 1981 and 1985, respectively M.S. and Ph.D. degrees in Electronic Engineering from Hanyang University. From 1990 to 1991, Invited Scientist of Communications Research Laboratory(or KARC) in Ministry of Posts and Telecommunications(Japan). From 2001 to 2002, Visiting Professor of Det. of Computer Eng. in North Carolaina State Univ. (United State). From 2010 to 2011, Dean of Engineering College in Kyungnam University. From 1985, currently Professor of Dept. of Information and Communication Eng. in Kyungnam University. His Research Interest: Software Engineering, Web-based Software Design & Development, Pattern Recognition, Image Processing, Computer Network & Network Computing, Embedded Processor System HW/SW
Email : yspark@kyungnam.ac.kr