정규논문 (Regular Paper)
방송공학회논문지 제18권 제5호, 2013년 9월 (JBE Vol. 18, No. 5, September 2013)
http://dx.doi.org/10.5909/JBE.2013.18.5.669
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

스테레오 비전을 위한 고성능 VLSI 구조

서 영 호[a)], 김 동 욱[a)‡]

High-Performance VLSI Architecture for Stereo Vision

Youngho Seo[a)] and Dong-Wook Kim[a)‡]

요 약

본 논문에서는 실시간으로 스테레오 정합을 수행하기 위한 VLSI(Very Large Scale Integrated Circuit)구조를 제안한다. 스테레오 정합의 연산을 분석하여 중간 연산 결과를 재사용하여 연산량과 메모리 접근수를 최소화한다. 이러한 동작을 수행할 수 있는 스테레오 정합 연산 셀의 구조를 제안하고, 이를 병렬적으로 확장하여 탐색 범위 내의 모든 비용함수를 동시에 연산할 수 있는 하드웨어의 구조를 제안한다. 이러한 하드웨어 구조를 확장하여 2차원 영역에 대한 비용함수를 연산할 수 있는 하드웨어의 구조와 동작을 제안한다. 구현한 하드웨어는 FPGA(Field Programmable Gate Array) 환경에서 최소 250Mhz의 클록 주파수에서 동작이 가능하고, 64화소의 탐색범위를 적용한 경우에 640×480 스테레오 영상을 약 805fps의 성능으로 처리할 수 있다.

Abstract

This paper proposed a new VLSI (Very Large Scale Integrated Circuit) architecture for stereo matching in real time. We minimized the amount of calculation and the number of memory accesses through analyzing calculation of stereo matching. From this, we proposed a new stereo matching calculating cell and a new hardware architecture by expanding it in parallel, which concurrently calculates cost function for all pixels in a search range. After expanding it, we proposed a new hardware architecture to calculate cost function for 2-dimensional region. The implemented hardware can be operated with minimum 250Mhz clock frequence in FPGA (Field Programmable Gate Array) environment, and has the performance of 805fps in case of the search range of 64 pixels and the image size of 640×480.

Keyword : stereo matching, cost function, SAD, hardware, parallel architecture, cell-based, FPGA, processor

Ⅰ. 서 론

최근 스테레오 비전은 다양한 연구가 수행되고 있다. 스테레오 영상으로부터 고품질의 변위의 획득, 객체 혹은 패턴 인식, 최적의 입체감 획득, 디스플레이, 스테레오 영상의 압축, 그리고 스테레오 정합의 고속화 등의 분야가 연구되고 있다[1-3]. 여기에서는 스테레오 정합 연산의 고속화를 위

a) 광운대학교 (Kwangwoon University)
‡ Corresponding Author : 김동욱 (Dong-Wook Kim)
E-mail: dwkim@kw.ac.kr
Tel: +82-2-940-5167 Fax: +82-2-919-5117
※ 이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2010-0026245).
· Manuscript received 7, March 2013 Revised 18, June 2013 Accepted 14, August 2013

한 분야를 다루고자 한다. 스테레오 정합은 영상의 모든 화소에 대해서 탐색 범위만큼의 비용함수 계산이 필요하기 때문에 연산량이 상당히 많은 과정이다[4]. 알고리즘에 따라서 차이는 있지만 소프트웨어를 이용하여 구현할 경우에 초당 1 프레임의 변위 영상을 얻는 것도 힘들다[5]. 따라서 실시간의 스테레오 비전 시스템을 구성하기 위해서는 스테레오 정합 과정을 하드웨어를 이용하여 구현해야 한다[6-11].

스테레오 정합을 고속으로 수행하기 위해서 하드웨어를 이용하여 다양한 연구가 진행되었다[6-10]. 스테레오 비전을 위한 전용 칩을 구현하는 것은 ASIC(Application Specific Integrated Circuit) 방식과 FPGA (Field Programmable Gate Array) 방식이 있다. Kimura는 SAZAN이라는 convolver-기반의 9시점 스테레오 머신을 제안하였다[6]. SAZAN은 320×240 크기의 영상과 25 화소의 탐색범위인 경우에 20fps(frame per sec)의 성능을 보인다. Kuhn은 SSD(Sum of Squared Differences)와 census 변환을 이용한 영역기반 방식과 폐색영역 검출 기법을 적용하여 고속의 면적 효율적인 ASIC 칩을 개발하였다[7]. 개발된 칩은 256×192 크기의 영상에 대해서 25 화소의 탐색범위를 적용했을 경우에 약 50fps의 속도로 변위영상을 생성할 수 있다. Darabiha는 local weighted phase-correlation이라는 multi-resolution, multi-orientation phase-기반의 알고리즘을 적용하여 깊이를 측정할 수 있는 Xilinx FPGA 기반의 하드웨어를 구현하였다[8]. 이 하드웨어는 256×360 크기의 영상에 대해서 20 화소의 탐색을 수행할 경우에 30fps의 성능으로 깊이 영상을 생성할 수 있다. Jia는 3시점의 카메라를 이용하여 고해상도의 변위영상을 생성할 수 있는 MSVM(Miniature Stereovision Machine)-III라는 전용 시스템을 개발하였다[9]. MSVM-III은 640×480 크기의 영상에 대해서 64 화소의 탐색을 수행할 경우에 약 30fps의 성능으로 깊이 영상을 생성할 수 있다. 이 시스템은 FPGA를 기반으로 하고 있는 60MHz의 속도로 동작할 수 있고, 카메라와 일체형 모듈로 개발되었다. Woodfill은 ASIC 형태의 DeepSea 프로세서를 이용하여 PCI 보드 형태의 DeepSea 스테레오 비전 시스템을 구현하였다[10]. 대기지연시간과 전력소모가 낮으면서 높은 프레임율로 스테레오 영상을 동시에 처리하도록 하였다. 이 시스템은 512×480 크기의 영상에 대해서 52 화소의 탐색을 수행할 경우에 약 200fps의 성능으로 깊이 영상을 생성할 수 있다. Jin은 보정, 스테레오 정합, 및 후처리를 수행할 수 있는 스테레오 비전 시스템을 개발하였다[11]. 이 시스템은 FPGA를 사용하고 있고 보드 형태이다. Census 변환을 이용하여 스테레오 정합을 수행하고, 부-화소 기반의 정밀도를 갖는다.

최근 대부분의 영상이 HD급 이상의 해상도를 사용하고 있기 때문에 CIF급으로 개발된 이전의 연구들[8][9]은 이러한 조건에 부합하지 못하고 성능을 만족시키지 못한다. MSVM-III[9]나 Jin[10]도 SD 영상을 대상으로 초당 30프레임의 성능을 보이고 있기 때문에 최근 영상들의 해상도를 처리하는 데에는 부족하다. 따라서 풀HD 영상과 최근의 UHD 영상을 만족하기 위한 새로운 고성능의 하드웨어가 필요하다.

본 논문은 스테레오 비전을 위해 전용 칩을 구현하기 위한 VLSI 구조를 제안하고자 한다. 제안하고자 하는 하드웨어는 사용자의 요구와 적용하는 알고리즘에 따라서 다양한 동작을 할 수 있는 VLSI 구조이다. 비용함수로 SAD를 이용하여 하드웨어 구조를 소개하지만, 다양한 비용함수를 사용할 수 있도록 하였다. 스테레오 정합 과정을 분석하여 하드웨어에 적합하도록 스케쥴링을 한 후에 이를 하드웨어 구조로 쉽게 변환이 될 수 있도록 단위 하드웨어(셀)를 제안한다. 이 셀을 단순히 확장하고 FIFO로 구성된 탐색범위의 크기를 갖는 라인 버퍼를 조합하면 다양한 크기의 스테레오 정합을 수행할 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 스테레오 정합 과정에 대해서 설명하고, 3장에서는 스테레오 정합의 하드웨어 구현을 위한 병렬 알고리즘을 설명한 후에 구조를 설명한다. 4장에서 제안한 하드웨어의 구현 결과를 보이고, 5장에서 결론을 맺는다.

II. 스테레오 정합

1. 개요

스테레오 정합은 좌우의 두 영상 중에서 한 영상을 기준

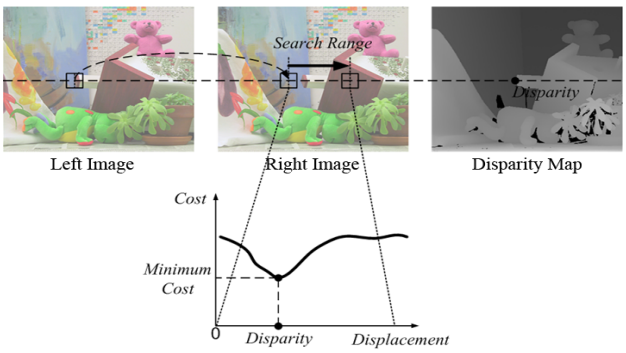

그림 1. 비용함수를 이용한 유사성 측정
Fig. 1. Similarity measurement using cost function

영상으로 정하고 다른 영상을 대상영상으로 정한 뒤 기준 영상의 특정 화소 혹은 영역을 대상영상에서 찾는 과정이다. 그림 1에는 좌측 영상에서 정합 윈도우로 정의된 특정 영역과 유사한 영역을 우측 영상에서 탐색하는 것을 보이고 있다. 유사성을 찾기 위해서 최소화 비용함수를 이용한다. 탐색구간에서 각각의 비용을 구한 뒤 최소의 비용을 갖는 변위를 변이벡터로 정의한다.

스테레오 정합 기법을 소프트웨어로 구현하여 각 비용함수 및 정합 윈도우의 크기가 갖는 특성을 분석하였다.

2. 비용함수

두 영상에서 대응점을 찾는 과정은 한 영상에서 특정 화소를 기준으로 인접 화소들의 집합을 의미하는 정합 윈도우와 유사성이 높은 윈도우를 다른 영상에서 찾는 것이다. 유사성에 대한 정량적인 판단기준은 일반적으로 비용함수로 정의한다. 비용함수는 비용 값에 대한 선택기준이 최소인지 최대인지에 따라서 SAD(Sum of Absolute Differences), SSD, 및 MAD(Minimization of Absolute Deviation) 등의 최소화 비용 함수와 NCC(Normalized Cross Correlation) 및 MNCC(Modified Normalized Cross Correlation)와 같은 최대화 비용 함수로 구분된다[12].

그림 2는 3×3 크기와 9×9 크기의 정합 윈도우를 사용하여 실험영상의 190번째 주사선에 대한 비용을 누적한 그림이다. 흰 선으로 표시된 부분은 최소비용을 추적하여 표시한 것이고, 높이는 각 위치에서의 변이정보에 해당한다. 여

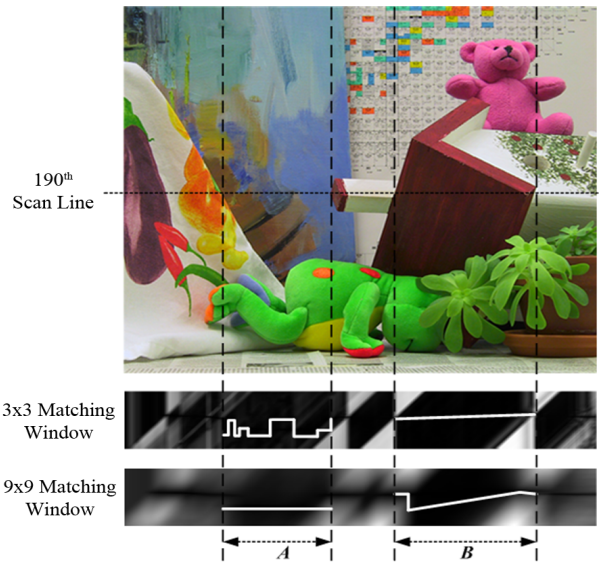

그림 2. 윈도우의 크기 변화에 대한 변이 공간의 예제
Fig. 2. An example of disparity space about a size of matching window

기서 A 영역과 B 영역은 각기 다른 영상의 특성으로 구분한 것인데, A 영역은 텍스쳐가 균일하여 비용함수의 변화가 적기 때문에 작은 크기의 정합 윈도우를 사용할 경우 변이정보가 제대로 추출되지 않음을 보이고 있다. B 영역은 물체의 윤곽부분에 대한 특성을 설명하고자 나타낸 영역으로, B 영역 경계는 영상 내 물체의 경계부분이며 배경과의 텍스쳐 변화가 심한 영역이다. 이러한 영역에서는 작은 크기의 정합 윈도우를 사용하는 경우, 비용 값이 명확하게 변화하여 비용 값에 대한 분포가 크다. 하지만 큰 크기의 정합 윈도우를 사용하는 경우 경계면에 위치한 비용 값이 왜곡되어 부정확한 변이정보를 추출하게 된다. 따라서 비용함수를 이용하여 유사성을 찾는 과정에 대한 효과를 높이기 위해서는 영상의 특성에 따라서 정합 윈도우의 크기를 적응적으로 사용하는 것이 좋다.

III. 병렬 알고리즘과 하드웨어 구조

1. 중간 연산과정의 분석

그림 3에 3×3 정합창을 이용하여 탐색거리가 10인 경우

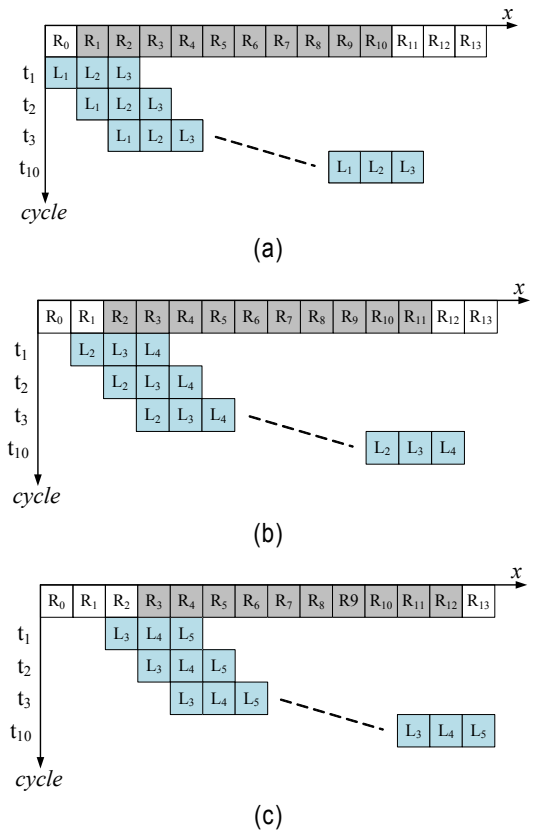

그림 3. 스테레오 정합 과정의 예
Fig. 3. Example of stereo matching

의 스테레오 정합을 수행하는 방법을 나타냈다. 그림 3은 논의의 간략화를 위해서 하나의 열에 대해서만 나타냈다. 그림 3에서 RX와 LX는 각각 오른쪽 및 왼쪽 영상의 화소를 나타낸다. 그림 3에서의 연산 과정을 표 1에 자세히 나타냈다. 표 1의 두 번째, 세 번째, 및 네 번째 행은 각각 그림 3(a), (b), 및 (c)에 해당한다.

SAD(L1, L2, L3)에 대한 첫 번째 연산 t1에서 |R0-L1|+|R1-L2|+|R2-L3|의 연산이 수행된다. 여기에서 |R1-L2|와 |R2-L3| 연산은 t1뿐만 아니라 SAD(L2, L3, L4)의 t1에서도 사용된다. 또한 SAD(L2, L3, L4)의 첫 번째 연산 t1는 |R1-L2|+|R2-L3|+|R3-L4|와 같고, 이 연산중에서 |R2-L3|와 |R3-L4|는 SAD(L3, L4, L5)에 대한 첫 번째 연산 t1에서 사용된다. 즉, 스테레오 정합을 구하는 중간 과정을 분석하면 중간 연산 결과들 사이에는 많은 연산의 중복성이 존재함을 확인할 수 있다. 이러한 중간 연산 결과의 중복성은 소프트웨어 방식으로 구현할 경우에 활용할 수 있는 방법이 없다. 중복성을 갖는 중간 연산 결과를 따로 처리하는 것과 매번 새로 계산하는 것과의 시간적인 차이가 없기 때문이다. 그러나 하드웨어로 구현할 경우에는 그러한 중간 연산 결과들은 메모리 요소에 의해서 저장되어질 수 있고, 시간적으로 적절히 관리한다면 연산량과 메모리 접근을 줄이면서 고속의 연산을 수행할 수 있게 된다.

표 1. SAD 연산의 예
Table 1. Example of SAD calculation

	SAD(L_1, L_2, L_3)	SAD(L_2, L_3, L_4)	SAD(L_3, L_4, L_5)
t_1	$\|R_0-L_1\|+\|R_1-L_2\|+\|R_2-L_3\|$	$\|R_1-L_2\|+\|R_2-L_3\|+\|R_3-L_4\|$	$\|R_2-L_3\|+\|R_3-L_4\|+\|R_4-L_5\|$
t_2	$\|R_1-L_1\|+\|R_2-L_2\|+\|R_3-L_3\|$	$\|R_2-L_2\|+\|R_3-L_3\|+\|R_4-L_4\|$	$\|R_3-L_3\|+\|R_4-L_4\|+\|R_5-L_5\|$
t_3	$\|R_2-L_1\|+\|R_3-L_2\|+\|R_4-L_3\|$	$\|R_3-L_2\|+\|R_4-L_3\|+\|R_5-L_4\|$	$\|R_4-L_3\|+\|R_5-L_4\|+\|R_6-L_5\|$
t_4	$\|R_3-L_1\|+\|R_4-L_2\|+\|R_5-L_3\|$	$\|R_4-L_2\|+\|R_5-L_3\|+\|R_6-L_4\|$	$\|R_5-L_3\|+\|R_6-L_4\|+\|R_7-L_5\|$
t_5	$\|R_4-L_1\|+\|R_5-L_2\|+\|R_6-L_3\|$	$\|R_5-L_2\|+\|R_6-L_3\|+\|R_7-L_4\|$	$\|R_6-L_3\|+\|R_7-L_4\|+\|R_8-L_5\|$
t_6	$\|R_5-L_1\|+\|R_6-L_2\|+\|R_7-L_3\|$	$\|R_6-L_2\|+\|R_7-L_3\|+\|R_8-L_4\|$	$\|R_7-L_3\|+\|R_8-L_4\|+\|R_9-L_5\|$
t_7	$\|R_6-L_1\|+\|R_7-L_2\|+\|R_8-L_3\|$	$\|R_7-L_2\|+\|R_8-L_3\|+\|R_9-L_4\|$	$\|R_8-L_3\|+\|R_9-L_4\|+\|R_9-L_5\|$
t_8	$\|R_7-L_1\|+\|R_8-L_2\|+\|R_9-L_3\|$	$\|R_8-L_2\|+\|R_9-L_3\|+\|R_9-L_4\|$	$\|R_9-L_3\|+\|R_{10}-L_4\|+\|R_{11}-L_5\|$
t_9	$\|R_8-L_1\|+\|R_9-L_2\|+\|R_{10}-L_3\|$	$\|R_9-L_2\|+\|R_{10}-L_3\|+\|R_{11}-L_4\|$	$\|R_{10}-L_3\|+\|R_{11}-L_4\|+\|R_{12}-L_5\|$
t_{10}	$\|R_9-L_1\|+\|R_{10}-L_2\|+\|R_{11}-L_3\|$	$\|R_{10}-L_1\|+\|R_{11}-L_3\|+\|R_{12}-L_4\|$	$\|R_{11}-L_3\|+\|R_{12}-L_4\|+\|R_{13}-L_5\|$

표 2. 중간 연산 결과의 공유를 고려한 SAD 연산
Table 2. SAD calculation for sharing of intermediate calculating results

Cycle	SAD(L_1, L_2, L_3)	SAD(L_2, L_3, L_4)	SAD(L_3, L_4, L_5)
t_1	$\|R_0-L_1\|+\|\mathbf{R_1-L_2}\|+\|\mathbf{R_2-L_3}\|$		
t_2	$\|R_1-L_1\|+\|\mathbf{R_2-L_2}\|+\|\mathbf{R_3-L_3}\|$	$\|R_1-L_2\|+\|\mathbf{R_2-L_3}\|+\|\mathbf{R_3-L_4}\|$	
t_3	$\|R_2-L_1\|+\|\mathbf{R_3-L_2}\|+\|\mathbf{R_4-L_3}\|$	$\|R_2-L_2\|+\|\mathbf{R_3-L_3}\|+\|\mathbf{R_4-L_4}\|$	$\|R_2-L_3\|+\|\mathbf{R_3-L_4}\|+\|\mathbf{R_4-L_5}\|$
t_4	$\|R_3-L_1\|+\|\mathbf{R_4-L_2}\|+\|\mathbf{R_5-L_3}\|$	$\|R_3-L_2\|+\|\mathbf{R_4-L_3}\|+\|\mathbf{R_5-L_4}\|$	$\|R_3-L_3\|+\|\mathbf{R_4-L_4}\|+\|\mathbf{R_5-L_5}\|$
t_5	$\|R_4-L_1\|+\|\mathbf{R_5-L_2}\|+\|\mathbf{R_6-L_3}\|$	$\|R_4-L_2\|+\|\mathbf{R_5-L_3}\|+\|\mathbf{R_6-L_4}\|$	$\|R_4-L_3\|+\|\mathbf{R_5-L_4}\|+\|\mathbf{R_6-L_5}\|$
t_6	$\|R_5-L_1\|+\|\mathbf{R_6-L_2}\|+\|\mathbf{R_7-L_3}\|$	$\|R_5-L_2\|+\|\mathbf{R_6-L_3}\|+\|\mathbf{R_7-L_4}\|$	$\|R_5-L_3\|+\|\mathbf{R_6-L_4}\|+\|\mathbf{R_7-L_5}\|$
t_7	$\|R_6-L_1\|+\|\mathbf{R_7-L_2}\|+\|\mathbf{R_8-L_3}\|$	$\|R_6-L_2\|+\|\mathbf{R_7-L_3}\|+\|\mathbf{R_8-L_4}\|$	$\|R_6-L_3\|+\|\mathbf{R_7-L_4}\|+\|\mathbf{R_8-L_5}\|$
t_8	$\|R_7-L_1\|+\|\mathbf{R_8-L_2}\|+\|\mathbf{R_9-L_3}\|$	$\|R_7-L_2\|+\|\mathbf{R_8-L_3}\|+\|\mathbf{R_9-L_4}\|$	$\|R_7-L_3\|+\|\mathbf{R_8-L_4}\|+\|\mathbf{R_9-L_5}\|$
t_9	$\|R_8-L_1\|+\|\mathbf{R_9-L_2}\|+\|\mathbf{R_{10}-L_3}\|$	$\|R_8-L_2\|+\|\mathbf{R_9-L_3}\|+\|\mathbf{R_9-L_4}\|$	$\|R_8-L_3\|+\|\mathbf{R_9-L_4}\|+\|\mathbf{R_9-L_5}\|$
t_{10}	$\|R_9-L_1\|+\|\mathbf{R_{10}-L_2}\|+\|\mathbf{R_{11}-L_3}\|$	$\|R_9-L_2\|+\|\mathbf{R_{10}-L_3}\|+\|\mathbf{R_{11}-L_4}\|$	$\|R_9-L_3\|+\|\mathbf{R_{10}-L_4}\|+\|\mathbf{R_{11}-L_5}\|$
t_{11}		$\|R_{10}-L_1\|+\|\mathbf{R_{11}-L_3}\|+\|\mathbf{R_{12}-L_4}\|$	$\|R_{10}-L_3\|+\|\mathbf{R_{11}-L_4}\|+\|\mathbf{R_{12}-L_5}\|$
t_{12}			$\|R_{11}-L_3\|+\|\mathbf{R_{12}-L_4}\|+\|\mathbf{R_{13}-L_5}\|$

스테레오 정합이 갖는 연산의 중복성을 제거하고 연산의 효율성을 높이기 위해서는 중간 연산 결과들을 재사용하는 방법을 적용한다. 중간 연산결과들을 재사용하기 위해 표 1의 연산 과정을 시간적으로 재배치한 결과를 표 2에 나타냈다. 표 2에서 굵은 글씨로 나타낸 항들은 다음 사이클로 넘겨줄 값이다. 예를 들어 SAD(L2, L3, L4)의 t4에 대한 연산을 살펴보면 SAD(L1, L2, L3)의 t3에서 |R3-L2|와 |R4-L3|의 값을 넘겨받고, |R5-L4| 연산을 수행하여 |R3-L2| +|R4-L3|+|R5-L4|를 구한다. 그리고 SAD(L3, L4, L5)의 t5에서 |R4-L3|와 |R5-L4|의 값을 사용할 수 있도록 넘겨준다. 이러한 동작이 모든 위치에서 동일하게 발생한다는 점은 셀 기반의 하드웨어 구조를 만들 수 있는 좋은 단서를 제공하는 것이다.

2. 연산 셀의 구조

표 2에서 어느 한 칸에 대한 연산은 |Li-1-Rj-1|+|Li-Rj|+|Li+1-Rj+1|와 같고, |Li-1-Rj-1|와 |Li-Rj|는 이전 연산 단계에서 연산이 이루어져 넘어온 값이다. |Li+1-Rj+1|는 현재 단계에서 새롭게 연산하는 값이고, 이 값들 중에서 |Li-Rj|와 |Li+1-Rj+1|는 다음 연산 단계를 위해서 넘겨주어야 하는 값들이다. 여기에서 i와 j는 현재의 스테레오 정합에 사용되는 왼쪽과 오른쪽 영상의 화소의 위치에 대한 좌표이다. 하드웨어 셀의 구조를 결정하기 위해 이러한 동작을 정리하면 아래와 같다.

입력 : |Li-1-Rj-1|, |Li-Rj|, Li+1, Rj+1
연산 : |Li+1-Rj+1|, |Li-1-Rj-1|+|Li-Rj|+|Li+1-Rj+1|
출력 : |Li-1-Rj-1|+|Li-Rj|+|Li+1-Rj+1|, |Li-Rj|, |Li+1-Rj+1|

이와 같이 중간 연산결과를 임시적으로 저장했다가 다음 화소의 연산에서 사용할 수 있는 동작을 포함하는 하드웨어 셀의 구조를 그림 4와 같이 제안한다.

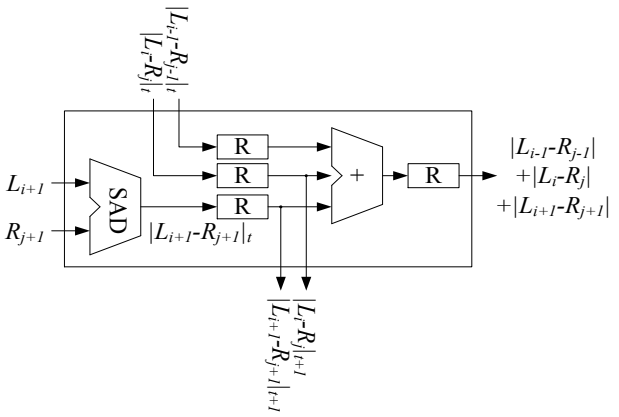

그림 4. 스테레오 정합을 위한 단위 셀(SMCU)의 구조
Fig. 4. Architecture of stereo matching calculating unit (SMCU)

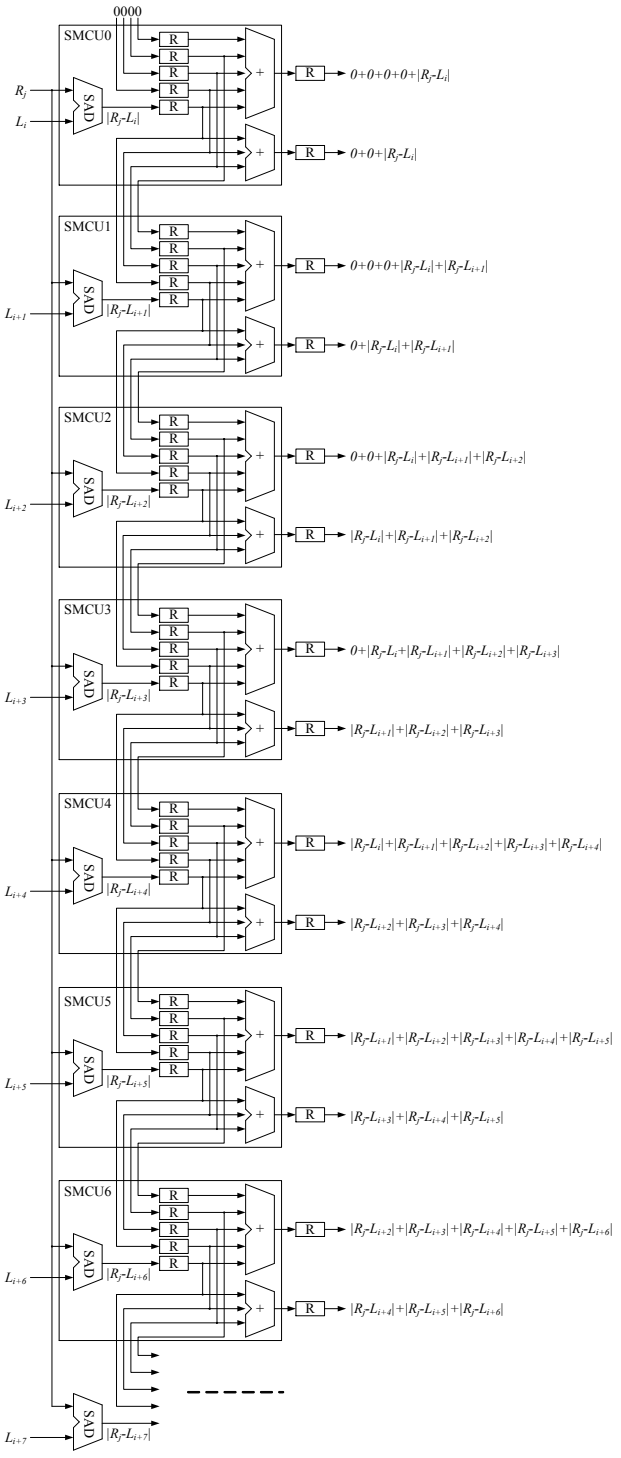

그림 5. 다중 정합창의 연산을 위한 SMHC의 구조
Fig. 5. Architecture of SMHC with multiple matching windows

3. 연산 셀의 구조수평방향의 병렬화

표 2에서 t5의 시간에 |R4-L1|+|R5-L2|+|R6-L3|, |R4-L2|+|R5-L3|+|R6-L4|, 및 |R4-L3|+|R5-L4|+|R6-L5| 연산이 수행된다. 이 세 연산을 살펴보면 첫 번째 항은 R4, 두 번째 항은 R5, 그리고 세 번째 항은 R6이 포함된다.

그림 4의 단위 셀을 그림 5와 같이 확장한다면 그림 3의 연산을 간단히 수행할 수 있고, 이 하드웨어를 스테레오 정합 가로 연산기(Stereo Matching Horizontal Calculator, SMHC)라 한다. 그림 5에서 3×3 정합창에 대한 결과는 세 번째 셀부터 출력되고, 처음 두 개의 결과는 무시한다. 5×5의 경우에는 다섯 번째 셀부터 연산 결과가 출력된다. 즉, 다중 정합창에 대한 연산을 병렬적으로 연산할 수 있다. 각 셀별 입력을 살펴보면 참조영상의 화소에 해당하는 Ri는 모든 셀에 공통적으로 입력되고, 매 클록 단위로 순서대로 입력된다.

SMHC를 살펴보면 정합창의 크기에 따라서 초기 SMCU의 개수와 이에 따른 대기 지연시간이 결정된다. 그리고 탐색 범위만큼의 SMCU 개수를 연결하면 완전한 병렬화가 가능하고, 탐색 범위에 상관없이 자동으로 동작이 연결된다. SMHC는 SMCU의 단순한 직렬 연결로 구성되어 있기 때문에 탐색 범위를 조절할 수 있고, SMHC 내의 SMCU를 그룹으로 나누고 탐색 범위를 줄이면 동시에 2개의 두 열에 대한 연산도 가능하다.

4. 연산 셀의 구조수평방향의 병렬화2D 영역을 위한 연산

일반적으로 스테레오 정합에서 사용되는 정합창은 2차원 구조를 갖는다. 2차원 연산을 위해서 이전 절에서 제안한 SMHC을 정합창의 수직 크기만큼 사용하여 연산하는 것이 가장 쉬운 방법이다. 이 경우에 연산은 비교적 단순하고 하드웨어의 구조도 단순하며 구현이 용이하다. 그러나 동일한 화소를 계속 호출해야 하기 때문에 메모리 접근 횟수와 하드웨어의 양이 과다하다. 또한 수직 방향으로 중복되는 중간 연산 결과들을 이용할 수 없는 단점이 있다.

3×3 크기의 정합창을 이용하여 스테레오 정합을 수행하고 있다고 가정하자. 그림 6(a)는 왼쪽 영상의 화소들을 나타내고, 그림 6(b)는 오른쪽 화소들을 나타낸다. 왼쪽 영상의 화소와 가장 유사한 오른쪽 영상에서의 위치를 찾는 과정을 수행하고 있다. 왼쪽 영상의 화소들은 이미 외부 메모리로부터 호출하여 보유하고 있고, 오른쪽 영상의 화소들을 순차적으로 호출하여 SAD 연산을 수행한다. 수직 방향의 연산을 포함한 2차원 정합창에 대한 SAD를 구하는 과정은 아래와 같이 3가지 단계(Step)로 구성된다.

Step1 : 수평 방향의 병렬 스테레오 정합 연산 과정 (표 2)
Step2 : 수평 방향의 비용 함수 결과들을 누적하는 과정
Step3 : 정합창에 포함되지 않는 수평방향의 열(row)을 누적된 비용 함수의 값에서 제거하는 과정

위의 3 단계를 시간 순서대로 하드웨어의 스케쥴링으로 나타내면 표 3과 같다. 표 3에서 Px,y는 한 열에 대한 SAD 결과들의 누적값이고, Sx,y는 Px,y를 누적한 결과이다. x와 y는 현재의 스테레오 정합에 사용되는 화소의 x축 및 y축 좌표를 나타낸다. 표 3에는 이해를 돕기 위해서 그림 6(b)의 회색 부분에 대한 연산 결과만을 나타내었다. 이러한 제약을 하지 않는다면 표 4와 같이 Step3에서 매 클록마다 3×3

표 3. 3×3 정합창을 이용한 연산 순서의 예시
Table 3. Example of calculation sequence using the 3×3 matching window

t	D	Step1	Step2	Step3
0	$R_{0,0}$			
1	$R_{0,1}$	$\|R_{0,1}-L_{0,1}\|$		
2	$R_{0,2}$	$\|R_{0,1}-L_{0,1}\|+\|R_{0,2}-L_{0,2}\|$		
3	$R_{0,3}$	$\|R_{0,1}-L_{0,1}\|+\|R_{0,2}-L_{0,2}\|+\|R_{0,3}-L_{0,3}\|=P_{0,2}$		
4	$R_{0,4}$		$P_{0,2}+0=S_{0,2}$	
5	$R_{1,0}$			
6	$R_{1,1}$	$\|R_{1,1}-L_{1,1}\|$		
7	$R_{1,2}$	$\|R_{1,1}-L_{1,1}\|+\|R_{1,2}-L_{1,2}\|$		
8	$R_{1,3}$	$\|R_{1,1}-L_{1,1}\|+\|R_{1,2}-L_{1,2}\|+\|R_{1,3}-L_{1,3}\|=P_{1,2}$		
9	$R_{1,4}$		$P_{1,2}+S_{0,2}=S_{1,2}$	
10	$R_{2,0}$			
11	$R_{2,1}$	$\|R_{2,1}-L_{2,1}\|$		
12	$R_{2,2}$	$\|R_{2,1}-L_{2,1}\|+\|R_{2,2}-L_{2,2}\|$		
13	$R_{2,3}$	$\|R_{2,1}-L_{2,1}\|+\|R_{2,2}-L_{2,2}\|+\|R_{2,3}-L_{2,3}\|=P_{2,2}$		
14	$R_{2,4}$		$P_{2,2}+S_{1,2}=S_{2,2}$	
15	$R_{3,0}$			$S_{2,2}-0=SAD_{1,2}$
16	$R_{3,1}$	$\|R_{3,1}-L_{3,1}\|$		
17	$R_{3,2}$	$\|R_{3,1}-L_{3,1}\|+\|R_{3,2}-L_{3,2}\|$		
18	$R_{3,3}$	$\|R_{3,1}-L_{3,1}\|+\|R_{3,2}-L_{3,2}\|+\|R_{3,3}-L_{3,3}\|=P_{3,2}$		
19	$R_{3,4}$		$P_{3,2}+S_{2,2}=S_{3,2}$	
20	$R_{4,0}$			$S_{3,2}-S_{0,2}=SAD_{2,2}$
21	$R_{4,1}$	$\|R_{4,1}-L_{4,1}\|$		
22	$R_{4,2}$	$\|R_{4,1}-L_{4,1}\|+\|R_{4,2}-L_{4,2}\|$		
23	$R_{4,3}$	$\|R_{4,1}-L_{4,1}\|+\|R_{4,2}-L_{4,2}\|+\|R_{4,3}-L_{4,3}\|=P_{4,2}$		
24	$R_{4,4}$		$P_{4,2}+S_{3,2}=S_{4,2}$	
25	$R_{5,0}$			$S_{4,2}-S_{1,2}=SAD_{3,2}$
26	$R_{5,1}$	$\|R_{5,1}-L_{5,1}\|$		
27	$R_{5,2}$	$\|R_{5,1}-L_{5,1}\|+\|R_{5,2}-L_{5,2}\|$		
28	$R_{5,3}$	$\|R_{5,1}-L_{5,1}\|+\|R_{5,2}-L_{5,2}\|+\|R_{5,3}-L_{5,3}\|=P_{5,2}$		
29	$R_{5,4}$		$P_{5,2}+S_{4,2}=S_{5,2}$	
30				$S_{5,2}-S_{2,2}=SAD_{4,2}$

표 4. 3×3 정합창을 이용한 병렬 연산 순서의 예시
Table 4. Example of parallel calculation sequence using the 3×3 matching window

t	D	Step1	Step2	Step3
0	$R_{0,0}$			
1	$R_{0,1}$	$P_{0,0}$		
2	$R_{0,2}$	$P_{0,1}$	$S_{0,0}$	
3	$R_{0,3}$	$P_{0,2}$	$S_{0,1}$	
4	$R_{0,4}$	$P_{0,3}$	$S_{0,2}$	
5	$R_{1,0}$	$P_{0,4}$	$S_{0,3}$	
6	$R_{1,1}$	$P_{1,0}$	$S_{0,4}$	
7	$R_{1,2}$	$P_{1,1}$	$S_{1,0}$	
8	$R_{1,3}$	$P_{1,2}$	$S_{1,1}$	
9	$R_{1,4}$	$P_{1,3}$	$S_{1,2}$	
10	$R_{2,0}$	$P_{1,4}$	$S_{1,3}$	
11	$R_{2,1}$	$P_{2,0}$	$S_{1,4}$	
12	$R_{2,2}$	$P_{2,1}$	$S_{2,0}$	
13	$R_{2,3}$	$P_{2,2}$	$S_{2,1}$	$SAD_{1,0}$
14	$R_{2,4}$	$P_{2,3}$	$S_{2,2}$	$SAD_{1,1}$
15	$R_{3,0}$	$P_{2,4}$	$S_{2,3}$	$SAD_{1,2}$
16	$R_{3,1}$	$P_{3,0}$	$S_{2,4}$	$SAD_{1,3}$
17	$R_{3,2}$	$P_{3,1}$	$S_{3,0}$	$SAD_{1,4}$
18	$R_{3,3}$	$P_{3,2}$	$S_{3,1}$	$SAD_{2,0}$
19	$R_{3,4}$	$P_{3,3}$	$S_{3,2}$	$SAD_{2,1}$
20	$R_{4,0}$	$P_{3,4}$	$S_{3,3}$	$SAD_{2,2}$
21	$R_{4,1}$	$P_{4,0}$	$S_{3,4}$	$SAD_{2,3}$
22	$R_{4,2}$	$P_{4,1}$	$S_{4,0}$	$SAD_{2,4}$
23	$R_{4,3}$	$P_{4,2}$	$S_{4,1}$	$SAD_{3,0}$
24	$R_{4,4}$	$P_{4,3}$	$S_{4,2}$	$SAD_{3,1}$
25	$R_{5,0}$	$P_{4,4}$	$S_{4,3}$	$SAD_{3,2}$
26	$R_{5,1}$	$P_{5,0}$	$S_{4,4}$	$SAD_{3,3}$
27	$R_{5,2}$	$P_{5,1}$	$S_{5,0}$	$SAD_{3,4}$
28	$R_{5,3}$	$P_{5,2}$	$S_{5,1}$	$SAD_{4,0}$
29	$R_{5,4}$	$P_{5,3}$	$S_{5,2}$	$SAD_{4,1}$
30	...	$P_{5,4}$	$S_{5,3}$	$SAD_{4,2}$
31	...	...	$S_{5,4}$	$SAD_{4,3}$
32	...	...	...	$SAD_{4,4}$

크기의 정합창에 대한 SAD 누적 결과(SAD1,0~SAD4,4)가 출력될 수 있다. 출력된 누적 SAD 값들 중에서 최소의 값을 선택하면 그 것이 스테레오 정합의 결과인 변위(Disparity)이다. 최소값을 선택하는 과정은 어려운 과정이 아니고 단순한 비교기이므로 자세한 설명은 생략한다.

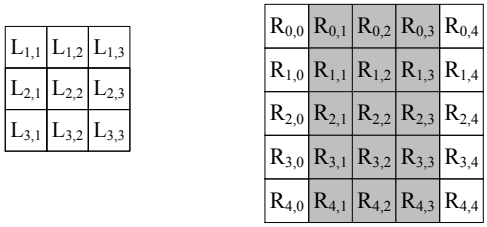

그림 6. 3×3 크기의 SAD 연산을 위한 화소 예시 (a) 왼쪽, (b) 오른쪽 영상의 화소
Fig. 6. Pixel example for SAD calculation of 3×3 size in (a) left, and (b) right image

표 4에서 제안한 2차원 스테레오 정합 스케쥴링을 위한 하드웨어 구조는 그림 7과 같다. 그림 7에서 S는 탐색 범위(Search Range)이고, M과 N은 각각 정합창의 가로 및 세로 크기를 나타낸다.

5. 하드웨어 특징

구현한 SMP(Stereo Matching Processor)는 아래와 같은 특성을 갖는다.

- 셀 기반의 구조 - 스테레오 정합의 중간 연산 결과에 대한 중복성을 분석하여 연산을 최소화시켜서 성능을 극대화하고 메모리 참조 횟수를 최소화한다. 다수 개의 셀을 단순히 연결만 해주면 가로 방향의 비용함수 연산

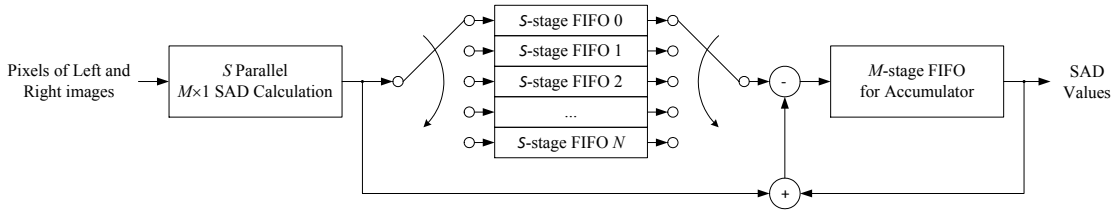

그림 7. 2차원 비용 함수 연산을 위한 하드웨어 동작
Fig. 7. Hardware operation for 2-dimensional cost function calculation

이 중간 결과가 중복되어 연산되지 않고 수행된다.

- 확장성 - 가로 방향의 연산을 담당하는 기본 하드웨어의 구조가 셀 단위로 되어 있고, 셀을 단순히 확장하면 그와 비례하여 성능을 향상시킬 수 있다. 셀의 확장은 단순히 추가하고자 하는 셀을 마지막 셀의 뒤에 연결하기만 하면 된다. 가로 방향의 연산 단위로 독립적인 하드웨어로 구성되므로 세로 방향의 연산은 가로 방향의 하드웨어를 병렬적으로 추가하기만 하면 쉽게 확장시킬 수 있는 구조를 갖는다. 또한 연산 순서를 분석하여 스테레오 정합을 위한 탐색 범위와 동일한 크기의 FIFO를 조합하면 하드웨어의 추가를 최소화하면서 2D 비용함수 계산을 할 수 있다.
- 단순한 제어 - 제안한 셀 기반의 스테레오 정합을 위한 연산기는 복잡한 제어가 필요 없고, 단순히 두 개의 영상에 대한 화소값을 순차적으로 입력시키면 약간의 대기 지연시간 이후부터 변위가 순서대로 출력된다. 1D 비용함수 계산기는 거의 제어가 필요하지 않고, 사용할 정합창을 선택하는 제어만 필요하다. 2D 비용함수 계산시는 FIFO의 선택을 위한 Line Select 신호의 제어 이외에 별다른 제어가 필요하지 않다.
- 프로그래밍성 - 제안한 하드웨어는 탐색범위, 정합창의 크기, 그리고 영상크기와 같은 항목에 대해서 프로그래밍이 가능하고, 그에 따른 다양한 스테레오 정합 연산을 수행할 수 있다. 탐색범위는 최대 64 화소까지 가능하고, 그림 8과 같이 16 화소의 단위로 프로그래밍이 가능하도록 하였다. FIFO를 더욱 작게 나눌 경우에는 더욱 세밀한 탐색 범위의 조절이 가능할 것이다. 탐색범위는 영상의 해상도 및 촬영 환경에 따라서 달라진다. 정합창은 최대 17×17 크기까지 가능하다. 영상크기에 대한 범위는 제한이 없으나 외부 메모리의 크기와 초당 성능에 따라서 응용분야에 따라 결정될 수 있다.

Ⅳ. 구현 결과

제안한 하드웨어의 구조는 VHDL을 이용하여 설계하였고, Altera FPGA 환경에서 구현하였다. 사용한 FPGA는 Stratix III EP3SL340F이고, 구현한 SMP를 FPGA에 사상한 결과를 표 5에 정리하였다.

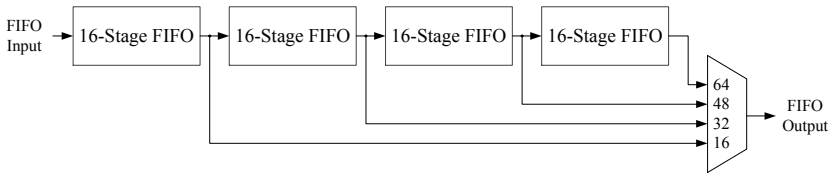

그림 8. 탐색범위의 프로그래밍을 위한 FIFO의 구성표
Fig. 8. FIFO configuration for programming search range

표 5. 하드웨어 자원
Table 5. Hardware resource

	Combinational ALUTs	Memory ALUTs	ALMs	Dedicated Logic Registers	Block Memory Bits	M9K
SMCU (bypass)	17	0	9	0	0	0
SMCU	683	15	421	299	0	0
SMHC	44,197	960	27,296	3,203	0	0
DC	5,907	0	3,566	2,399	0	0
FIFO	2,112	0	0	6,656	131,072	64
Logics	1,397	0	5,628	4,531	0	0
Total	56,362	960	36,490	11,797	131,072	64
Stratix III EP3SL340F	270,400	135,200	-	270,400	16,662,528	-
Utilization	20.84%	0.71%	-	4.36%	0.79%	-

제안한 하드웨어는 다양한 프로그래밍이 가능하고, 그에 따른 다양한 스테레오 정합 연산을 수행할 수 있다. 두 번째 열(Row)의 SMCU (Bypass)는 유효한 출력이 나오기 전까지의 SMCU를 의미한다. 이 SMCU는 데이터를 다음 SMCU로 넘겨주는 역할만 수행하고 연산기는 포함하지 않는다. 다섯 번째 열의 DC는 변위 계산기를 나타내고 여섯 번째 열의 Logics는 여러 제어기들과 타이밍을 맞추기 위한 버퍼들, 그리고 데이터 버스 등을 포함한 회로를 의미한다. 여덟 번째 열의 Total이 전체 구현 결과에 대한 자원 사용량 나타낸다. 아홉 번째 열은 Stratix III EP3SL340F의 전체 자원을 나타내고 열 번째 열은 사용한 FPGA의 전체 자원에 대한 구현한 하드웨어의 자원 사용률을 나타낸다. Combinational ALUT는 56,362개를 사용하여 20.84%의 사용률을 보였고, Memory ALUT는 135,200개를 사용하여 0.71%의 사용률을 보인다. 그리고 FIFO가 주로 사용하는 Memory Bit는 131,072비트를 사용하여 0.79%의 자원 사용률을 보였다.

구현된 하드웨어는 입력되는 영상의 속도와 동일한 속도로 변위가 출력되므로 거의 스테레오 정합이 가질 수 있는 거의 최대 성능을 갖는다. 구현된 하드웨어는 전체적으로 100% 파이프라인화되어 있고, 임계경로는 8비트 덧셈기에 위치하여 고속의 동작이 가능하다. 17×17 크기의 정합창을 이용하는 경우에 9개의 8비트 데이터를 더해야 한다. 그러나 이러한 동작은 모두 파이프라인화하여 클록에 따라 2개의 8비트 덧셈으로 나누어져 있기 때문에 동작 속도에 영향을 주지 않는다. 사용한 FPGA에서는 최소 250MHz 이상의 동작 주파수에서 동작할 수 있다. 250MHz에서 동작할 경우에 250Mbps(byte per sec)의 성능을 갖는다. 즉, 640×480의 영상에 대해서 약 805fps(frames per sec)의 성능을 갖는다.

표 6에는 이전 연구결과들과 구현된 결과들의 성능을 요약하여 비교하였다. Jin의 H/W와 비교할 때 640×480 크기의 영상에 대해서 약 4배 만큼 성능이 향상된 것을 볼 수 있다. 또한 GPU를 이용한 방법에 비해서 대체적으로 성능이 높다. 그러나 정합 방법의 난이도가 다르고, 내장된 기능에 차이가 있기 때문에 절대적으로 성능이 우수하다고 주장하기는 어렵다.

Ⅳ. 결 론

본 논문에서는 실시간으로 스테레오 정합을 수행하기 위한 고성능 하드웨어 구조를 제안하고 이를 구현하였다. 하드웨어의 동작과 구조에 적합하게 스테레오 정합의 연산을 분석하였고, 이를 바탕으로 중간 연산 결과를 재사용할 수 있는 셀 기반의 병렬 하드웨어 구조를 제안하였다. 먼저 스테레오 정합 연산 셀(Stereo Matching calculating unit, SMCU)의 구조를 제안하고, 이를 열 방향으로 병렬적으로 확장하여 탐색 범위 내의 모든 비용함수를 동시에 연산할 수 있는 하드웨어(1D Stereo Matching Calculator, 1DSMC)를 제안하였다. SMHC의 결과에 대해서 FIFO와 간단한 덧셈 및 뺄셈 연산기를 이용하여 2차원 영역에 대한 비용함수를 연산할 수 있는 하드웨어 구조(2D Stereo Matching Calculator, 2DSMC)를 제안하였다. 제안

표 6. 성능
Table 6. Performance Comparison

Implemented System	Platform	Image Size	Matching Method	Disparity Range	Window Size	Frames per Second
MSVM-III [9]	FPGA	640×480	SSAD/Trinocular	64	N/A	30
DeepSea [10]	ASIC	512×480	Census	52	Census (N/A) Corr (N/A)	200
Jin [11]	FPGA	640×480	Census	64	Census (11×11) Corr (15×15)	230
Ours	FPGA	640×480	Adaptive SAD	64	Multiple (1×1~17×17)	805

된 하드웨어는 스테레오 정합을 위한 칩셋 개발 분야에 고성능 VLSI 구조로 이용될 수 있고, 스테레오 비전 분야에서 다양하게 응용될 수 있을 것으로 사료된다.

참 고 문 헌

[1] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” International Journal of Computer Vision, Vol. 47, Issue 1-3, pp. 7-42, April 2002.

[2] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, Issue 8, pp. 993-1008, Aug. 2003.

[3] Y. Wang, J. Ostermann, and Y. Q. Zhang, "Video Processing and Communications," Prentice hall, 2002.

[4] J. Woodfill, B. von Herzen, and R. Zabih, Frame-rate robust stereo on a PCI board. Available: http://www.cs.cornell.edu/rdz/ Papers/Archive /fpga.pdf

[5] J. Sun, N. N. Zheng, and H. Y. Shum, "Stereo matching using belief propagation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, Issue 25, pp. 787-800, July 2003.

[6] S. Kimura, T. Shinbo, H. Yamaguchi, E. Kawamura, and K. Nakano, “"A convolver-based real-time stereo machine (SAZAN),"" in Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit., Fort Collins, CO, vol. 1, pp. 457-463, Jun. 1999.

[7] M. Kuhn, S. Moser, O. Isler, F. K. Gurkaynak, A. Burg, N. Felber, H. Kaeslin, and W. Fichtner, “"Efficient ASIC implementation of a real-time depth mapping stereo vision system,"" in Proc. IEEE Int. Circuits Syst. (MWSCAS ’03), Cairo, Egypt, vol. 3, pp. 1478-1481, Dec. 2003.

[8] A. Darabiha, J. Rose, and W. J. Maclean, “ideo-rate stereo depth measurement on programmable hardware,”in Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit., Madison, WI, vol. 1, pp. 203 –210, Jun. 2003.

[9] Y. Jia, X. Zhang, M. Li, and L. An, “A miniature stereo vision machine (MSVM-III) for dense disparity mapping,” in Proc. 17th Int. Conf. Pattern Recognit., Cambridge, U.K., vol. 1. pp. 728 – 731, Aug. 2004

[10] J. I. Woodfill, G. Gordon, and R. Buck, “"Tyzx DeepSea high speed stereo vision system,"" in Proc. IEEE Comput. Soc. Workshop Real-Time 3-D Sensors Use Conf. Comput. Vision Pattern Recog., Washington D.C., Jun. 2004, pp. 41 – 46.

[11] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S.-K. Park, M. Kim, and J. W. Jeon,, “FPGA Design and Implementation of a Real-Time Stereo Vision System,“ IEEE Transactions on Circuits and Systems for Video Technology, Vol. 20, No. 1, pp. 15-26, Jan. 2010.

[12] R. C. Gonzales and R. E. Woods, "Digital image processing," Prentice Hall, 2nd edtion, 2001.

저 자 소 개

서 영 호

- 1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
- 2001년 2월 : 광운대학교 일반대학원 졸업(공학석사)
- 2004년 9월 : 광운대학교 일반대학원 졸업(공학박사)
- 2003년 9월 ~ 2004년 6월 : 한국전기연구소 연구원
- 2005년 9월 ~ 2008년 2월 : 한성대학교 조교수
- 2008년 3월 ~ 현재 : 광운대학교 교양학부 부교수
- 주관심분야 : 실감미디어, 2D/3D 영상 신호처리, 디지털 홀로그램

김 동 욱

- 1983년 2월 : 한양대학교 전자공학과 졸업(공학사)
- 1985년 2월 : 한양대학교 전자공학과 졸업(공학석사)
- 1991년 9월 : Georgia 공과대학 전기공학과 졸업(공학박사)
- 1992년 3월 ~ 현재 : 광운대학교 전자재료공학과 교수
- 2000년 3월 ~ 2001년 12월 : 인티스닷컴(주)연구원
- 2006년 3월 ~ 현재 : (사)실감미디어 산업협회 이사
- 2009년 3월 ~ 현재 : 광운대학교 실감미디어 연구소장
- 주관심분야 : 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication