(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년08월24일
(11) 등록번호 10-1650324
(24) 등록일자 2016년08월17일

(51) 국제특허분류(Int. Cl.)
H04N 19/117 (2014.01) *H04N 19/50* (2014.01)
H04N 19/80 (2014.01) *H04N 19/85* (2014.01)
(52) CPC특허분류
H04N 19/117 (2015.01)
H04N 19/50 (2015.01)
(21) 출원번호 10-2015-0028332
(22) 출원일자 2015년02월27일
심사청구일자 2015년02월27일
(56) 선행기술조사문헌
JP2013236358 A
WO2013029474 A1
"Reduced-Rank Unscented Kalman Filtering Using Cholesky-Based Decomposition" (J.Chandrasekar et al., 2008 American Control Conference, 2008.6.11.공개).*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
한밭대학교 산학협력단
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
(72) 발명자
류광기
XXXXXXXXXXXXXXX
신승용
XXXXXXXXXXXXXXX
(74) 대리인
추혁

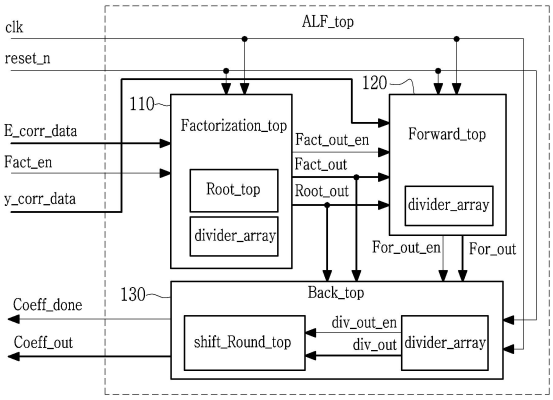
전체 청구항 수 : 총 5 항 심사관 : 김영태

(54) 발명의 명칭 **HEVC의 적응적 루프 필터**

(57) 요 약

HEVC의 적응적 루프 필터가 개시된다. 행렬 U를 계산하고 각 행에 대한 루트 연산 값과 행렬 U를 출력하는 팩토라이제이션 모듈, 상관정보와 행렬 U를 입력받아 벡터 d를 계산하는 포워드 모듈 및 행렬 U와 벡터 d를 입력받아 벡터 x를 계산하는 백 모듈을 구성한다. 따라서 연산량과 연산수행시간을 최소화할 수 있다.

대 표 도 - 도4



(52) CPC특허분류
H04N 19/80 (2015.01)
H04N 19/85 (2015.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 2014029505
부처명 교육부
연구관리전문기관 한국연구재단(교육부)
연구사업명 지역혁신인력양성사업
연구과제명 고령자 및 저시력 장애인용 전자독서확대기를 위한 고성능 시스템칩 개발
기 여 율 1/1
주관기관 연구사업지원팀
연구기간 2014.05.01 ~ 2015.04.30
공지예외적용 : 있음

명 세 서

청구범위

청구항 1

아래의 식 1의 왼쪽 항에 위치한 자기 상관행렬인 10×10 행렬구조를 갖는 제1 행렬(E_corr_data)이 입력되어 상기 제1 행렬(E_corr_data)을 아래의 식 2 및 3를 이용한 촐레스키 분해법을 이용해 10×10 행렬 구조를 갖는 제2 행렬(행렬 U)을 생성하고 상기 제2 행렬(행렬 U)의 각 행에 대한 루트 연산 결과값(Root_out)을 생성하여 제2 행렬(행렬 U)의 행렬 요소(Fact_out)와 루트 연산 결과값(Root_out)을 출력하는 팩토라이제이션 모듈,

상기 팩토라이제이션 모듈에 연결되어 있고, 상기 제2 행렬(행렬 U)의 상기 행렬 요소(Fact_out), 상기 루트 연산 결과값(Root_out) 및 식 2의 오른쪽 항에 위치한 상호 상관행렬인 10×1 행렬 구조를 갖는 제1 벡터(y_corr_data)를 입력받아, 입력된 상기 제2 행렬(행렬 U)을 곱셈 및 뺄셈 연산하고 곱셈 및 뺄셈 연산된 결과값과 상기 제1 벡터(y_corr_data)를 이용해 나눗셈 연산을 실시해 10×1 행렬 구조를 갖는 제2 벡터(For_out)를 생성하는 포워드 모듈, 그리고

상기 팩토라이제이션 모듈 및 상기 포워드 모듈에 연결되어 있고, 상기 제2 행렬(행렬 U)의 행렬 요소(Fact_out), 상기 루트 연산 결과값(Root_out) 및 상기 제 2 벡터(For_out)를 입력받아, 상기 제2 행렬(행렬 U)과 상기 제2 벡터(For_out)를 곱셈 및 뺄셈하고, 상기 곱셈 및 뺄셈 연산된 결과값과 상기 제2 행렬(행렬 U)을 나눗셈 연산하며, 나눗셈 연산 시 나머지 값에 쉬프트와 반올림 연산을 수행하고 10개의 필터 계수값(C0-C9)을 생성하는 백 모듈

을 포함하는 적응적 루프 필터.

식 1

$$\begin{bmatrix} \sum_{r=0}^{R-1} t'[r+p_0]t'[r+p_0] & \sum_{r=0}^{R-1} t'[r+p_1]t'[r+p_0] & \cdots & \sum_{r=0}^{R-1} t'[r+p_9]t'[r+p_0] \\ \sum_{r=0}^{R-1} t'[r+p_0]t'[r+p_1] & \sum_{r=0}^{R-1} t'[r+p_1]t'[r+p_1] & \cdots & \sum_{r=0}^{R-1} t'[r+p_9]t'[r+p_1] \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \sum_{r=0}^{R-1} t'[r+p_0]t'[r+p_9] & \sum_{r=0}^{R-1} t'[r+p_1]t'[r+p_9] & \cdots & \sum_{r=0}^{R-1} t'[r+p_9]t'[r+p_9] \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_9 \end{bmatrix} = \begin{bmatrix} \sum_{r=0}^{R-1} s[r]t'[r+p_0] \\ \sum_{r=0}^{R-1} s[r]t'[r+p_1] \\ \cdot \\ \cdot \\ \cdot \\ \sum_{r=0}^{R-1} s[r]t'[r+p_9] \end{bmatrix}$$

식 2

$$u_{ii} = \sqrt{a - \sum_{k=1}^{i-1} u_{ki} \times u_{ki}}$$, 그리고

식 3

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} u_{ki} \times u_{kj}}{u_{ii}}$$

청구항 2

제1항에서,

상기 팩토라이제이션 모듈은,

상기 제1 행렬(E_corr_data)이 입력되어, 상기 제1 행렬(E_corr_data)의 각 행에 대한 루트 연산 결과값(Root_out)을 생성해 출력하는 루트 모듈,

상기 제1 행렬(E_corr_data)이 입력되어 상기 제1 행렬(E_corr_data)을 저장하는 레지스터,

상기 루트 모듈과 상기 레지스터에 연결되어 있고, 상기 루트 모듈에서 출력되는 루트 연산 결과값(Root_out)과

상기 제1 행렬(E_corr_data)에 나눗셈 연산을 수행하는 디바이더 모듈, 그리고

상기 루트 모듈 및 상기 디바이더 모듈에 연결되어 있고, 상기 디바이더 모듈의 결과값(div_out)에 곱셈 및 뺄셈 연산을 실시하여 상기 제2 행렬(행렬 U)을 생성하는 곱셈뺄셈기

를 포함하는 적응적 루프 필터.

청구항 3

제2항에서,

상기 루트 모듈은,

입력 신호(Root_in)와 뺄셈 모듈의 결과값(Sub_out)에 2비트인 01을 오른쪽에 결합한 값을 비교하는 비교기, 그리고

상기 비교기에 연결되어 있고, 상기 입력 신호(Root_in)와 상기 비교기의 결과값(Comp_out)을 뺄셈하는 뺄셈기

를 포함하는 적응적 루프 필터.

청구항 4

제1항에서,

상기 포워드 모듈은,

상기 팩토라이제이션 모듈에 연결되어, 상기 제2 행렬(행렬 U)의 행렬 요소(Fact_out), 상기 제1 벡터(y_corr_data) 및 상기 루트 연산 결과값(Root_out)을 입력받아 곱셈 뺄셈하는 곱셈뺄셈기,

상기 제1 벡터(y_corr_data)가 입력되어 상기 제1 벡터(y_corr_data)를 저장하는 레지스터, 그리고

상기 곱셈뺄셈기와 상기 레지스터에 연결되어 있고, 상기 곱셈뺄셈기의 결과값(Mul_sub_out)과 상기 레지스터에 저장되어 있는 상기 제1 벡터(y_corr_data)의 값(reg_out)을 나눗셈 연산 수행하는 디바이더 모듈

을 포함하는 적응적 루프 필터.

청구항 5

제1항에서,

상기 백 모듈은,

상기 제2 행렬(행렬 U)의 행렬 요소(Fact_out), 상기 루트 연산 결과값(Root_out) 및 상기 제2 벡터(For_out)가 입력되고, 상기 제2 행렬(행렬 U)과 상기 제2 벡터(For_out)를 곱셈 뺄셈하는 곱셈뺄셈기,

상기 제2 행렬(행렬 U)의 행렬 요소(Fact_out)가 입력되어 상기 제2 행렬(행렬 U)의 행렬 요소(Fact_out)를 저장하는 레지스터,

상기 곱셈뺄셈기와 상기 레지스터에 연결되어 있고, 상기 곱셈뺄셈기의 결과값(Mul_sub_out)과 상기 레지스터의 상기 제2 행렬(행렬 U)을 나눗셈 연산 수행하는 디바이더 모듈, 그리고

상기 디바이더 모듈과 연결되어 있고, 상기 디바이더 모듈의 나머지 값을 쉬프트와 반올림 연산을 수행하여 필터 계수값(C0-C9)을 생성하는 쉬프트라운드 모듈

을 포함하는 적응적 루프 필터.

발명의 설명

기 술 분 야

[0001] 본 발명은 HEVC의 적응적 루프 필터에 관한 것으로, 더욱 상세하게는 필터 계수를 출력하는 HEVC의 적응적 루프 필터에 관한 것이다.

배 경 기 술

[0002] 최근 UHD(Ultra High-Definition) 방송 상용화와 더불어 고화질 콘텐츠 시청에 대한 소비자들의 욕구가 증가하면서 UHD TV 대중화가 급속히 진행되고 있다. 또한, 모바일 시장에서도 기본화면 해상도가 HD급인 스마트폰이 출시되면서 고해상도 및 고화질의 영상에 대한 사용자들의 수요가 급증하고 있는 추세이다. 그러나 고해상도 및 고화질의 영상은 데이터의 양이 매우 방대하기 때문에 기존에 쓰이던 표준 영상 압축 기술인 H.264/AVC는 고해상도 및 고화질 영상서비스를 지원하기에는 한계가 있다. 이에 따라 ITU-T VCEG(Video Coding Experts Group)와 ISO/IEC MPEG(Moving Picture Experts Group)은 H.264/AVC보다 더 높은 압축률과 더 낮은 복잡도의 새로운 차세대 영상 압축 표준의 필요성에 의해, 2010년 초 HEVC(High Efficiency Video Coding)라는 표준화 활동을 시작했으며, 2013년 1월 초 스위스 제네바 회의에서 HEVC 최종 표준안(FDIS, Final Draft International Standard)을 완성하였다. 영상 압축 표준 HEVC는 기존의 H.264/AVC에 비해 약 35%의 부호화 효율을 보인다.

[0003] HEVC의 ALF(Adaptive Loop Filter) 기술은 기존의 H.264/AVC보다 더 높은 압축 성능과 주관적 화질을 향상시키기 위해 HEVC에 새로 삽입된 루프내(In-loop) 필터링 방법 중 가장 마지막으로 처리하는 필터이다. ALF 기술은 양자화 과정에서 발생하는 정보의 손실을 보상하기 위해서 복원된 영상에 적응적으로 필터링을 수행하는 기술이다. 즉, ALF는 위너 필터를 기반으로 원본 영상과 복원된 영상간의 평균자승오차를 최소화시키는 기술이다. ALF 기술은 적응적으로 필터링을 수행하기 위해 필터 계수를 추출하기 위한 10×10 행렬의 촐레스키 분해를 반복적으로 수행하기 때문에 복잡한 연산 과정과 상당한 연산 수행시간을 필요로 한다. 또한, ALF는 HEVC에서 처리하는 최대 블록 크기인 64×64 화소 단위로 연산을 수행하기 때문에 많은 연산량과 수행시간을 요구한다.

발명의 내용

해결하려는 과제

[0004] 상기와 같은 문제점을 해결하기 위한 본 발명의 목적은, 연산량과 수행시간을 줄이는 HEVC의 적응적 루프 필터를 제공하는데 있다.

과제의 해결 수단

[0005] 상기 목적을 달성하기 위한 본 발명은, 행렬 U를 계산하고 각 행에 대한 루트 연산 값과 행렬 U를 출력하는 팩토라이제이션 모듈, 상관정보와 행렬 U를 입력받아 벡터 d를 계산하는 포워드 모듈 및 행렬 U와 벡터 d를 입력받아 벡터 x를 계산하는 백 모듈을 제공한다.

[0006] 여기에서, 팩토라이제이션 모듈은 행렬 U와 상관정보에 루트 연산을 수행하는 루트 모듈, 상관정보를 저장하는 레지스터, 루트 모듈의 결과값과 상관정보에 나눗셈 연산을 수행하는 디바이더 모듈 및 디바이더 모듈의 결과값에 곱셈기와 뺄셈기로 행렬 U를 계산하는 곱셈뺄셈기를 포함한다.

[0007] 이때, 루트 모듈은 입력 신호와 뺄셈기의 결과값에 2비트 01을 오른쪽에 결합한 값을 비교하는 비교기 및 입력 신호와 비교기의 결과값을 뺄셈하는 뺄셈기를 포함한다.

[0008] 이때, 포워드 모듈은 상관정보와 행렬 U를 입력받아 곱셈 뺄셈하는 곱셈뺄셈기, 상관정보를 저장하는 레지스터 및 곱셈뺄셈기의 결과값과 레지스터의 상관정보를 나눗셈 연산을 수행하는 디바이더 모듈을 포함한다.

[0009] 이때, 백 모듈은 행렬 U와 상기 벡터 d를 곱셈 뺄셈하는 곱셈뺄셈기, 행렬 U를 저장하는 레지스터, 곱셈뺄셈기의 결과값과 레지스터의 행렬 U를 나눗셈 연산을 수행하는 디바이더 모듈 및 디바이더 모듈의 나머지 값을 쉬프트와 반올림 연산을 수행하여 필터 계수값을 생성하는 쉬프트라운드 모듈을 포함한다.

발명의 효과

[0010] 상기와 같은 본 발명에 따른 HEVC의 적응적 루프 필터를 이용할 경우에는 연산량과 연산수행시간을 최소화할 수 있다.

도면의 간단한 설명

[0011] 도 1은 HM 7.0 부호화기 블록 다이어그램을 보인 예시도이다.

도 2는 HM 7.0의 ALF 필터 형태를 보인 예시도이다.

도 3은 촐레스키 분해법을 이용한 가우시안 제거 방법을 보인 예시도이다.

도 4는 본 발명의 일실시예에 따른 적응적 루프 필터의 구성을 보인 블록도이다.

도 5는 2단 파이프라인 구조를 보인 예시도이다.

도 6은 도 4의 팩토라이제이션 모듈의 구성을 보인 블록도이다.

도 7은 루트 연산 방법을 보인 예시도이다.

도 8은 도 4의 루트 연산 하드웨어 구조를 보인 블록도이다.

도 9는 도 4의 포워드 모듈의 구성을 보인 블록도이다.

도 10은 도 4의 백 모듈의 구성을 보인 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0012] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세한 설명에 상세하게 설명하고자 한다. 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 각 도면을 설명하면서 유사한 참조부호를 유사한 구성요소에 대해 사용하였다.

[0013] 제1, 제2, A, B 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 본 발명의 권리 범위를 벗어나지 않으면서 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.

[0014] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다.

[0015] 본 출원에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0016] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥 상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0017] 이하, 본 발명에 따른 바람직한 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0018] 도 1은 HM 7.0 부호화기 블록 다이어그램을 보인 예시도이다.

[0019] HEVC Model(HM) 7.0 ALF(Adaptive Loop Filter)는 주관적 화질과 압축 효율을 향상시키는 기술로써, HEVC 부호화기에 새로 삽입된 루프내 필터링 방법이다. ALF의 특징은 각 영상에 대해 최적의 필터 계수를 적응적으로 적용함으로써 주관적 화질과 압축 효율을 상당히 향상시킬 수 있다.

[0020] ALF의 전체적인 연산 과정은 바운드리 패딩, 필터 계수 추출, 필터링을 포함하는 총 세 가지의 하위 과정으로 구성된다. 첫 번째 단계는 바운드리 패딩으로 복원된 영상의 바깥쪽 경계나 필터링 하고자 하는 블록의 경계에서 필요한 픽셀 값들이 존재하지 않는 부분을 이웃한 픽셀 값들로 채워준다. 두 번째 단계는 필터 계수들을 추출하는 과정이다. 필터 계수들을 추출하는 과정은 복원된 과거 영상 또는 현재 영상을 이용하여 위너 필터 기반의 통계적인 특성을 통해 필터 계수들을 계산한다. 두 번째 단계가 본 발명의 핵심부분에 해당하며, 중심적으로 다루고자 한다. 마지막 단계는 복원된 영상의 픽셀 값들과 계산한 필터 계수들을 이용하여 도 2와 같은 필터 형태로 필터링을 수행한다.

[0021] 도 2는 HM 7.0의 ALF 필터 형태를 보인 예시도이다.

[0022] ALF는 도 2와 같이 HEVC Model(HM)7.0에서 채택하고 있는 9×7 십자가 모양에 3×3 정사각형 모양이 합쳐진 필터 형태를 가지며, 필터의 대칭적인 구조에 따라 총 10개의 필터 계수를 갖는다.

[0023] ALF는 위너 필터 개념을 기반으로 원본 영상과 복원된 영상 간의 평균자승오차(MSE, Mean Square Error)를 최소화함으로써 복원된 영상이 원본 영상에 가까운 신호로 필터링을 수행한다. 수학식 1은 위너 필터 기반의 ALF 필터링 결과를 수식으로 나타낸 것이다. 수학식 1에서 f[r]은 ALF의 결과 영상을 의미하고, t[r]은 복원된 영상(Sample Adaptive Offset)을 의미한다. r은 2D 영상의 [x,y]에 해당하는 좌표 정보를 의미하고, c_n은 ALF의 필터 계수를 의미한다. p_n은 필터링을 수행하기 위한 위치 오프셋을 의미한다.

[0024] $$f[r]=\sum_{n=0}^{N-1} c_n t[r+p_n]$$ 수학식 1

[0025] 수학식 1을 이용하여 ALF 결과 영상 f[r]과 원본 영상 s[r]간의 최소의 오차제곱합(SSE, Sum of Squared Error)을 계산하기 위해 '0' 과 같다고 가정하고 ALF 필터 형태의 대칭적인 구조를 적용하여 계산하면, 수학식 2와 같이 위너-홉 방정식을 유도할 수 있다. 수학식 2에서 R은 복원된 영상 t[r]의 블록 단위 크기를 의미한다. 왼쪽 항의 10×10 행렬은 자기 상관관계를 의미하고, 오른쪽 항의 10×1 행렬인 10×1 벡터는 상호 상관관계를 의미한다. t'는 수학식 3을 이용하여 계산할 수 있다.

[0026] $$\begin{bmatrix} \sum_{r=0}^{R-1} t'[r+p_0]t'[r+p_0] & \sum_{r=0}^{R-1} t'[r+p_1]t'[r+p_0] & \cdots & \sum_{r=0}^{R-1} t'[r+p_9]t'[r+p_0] \\ \sum_{r=0}^{R-1} t'[r+p_0]t'[r+p_1] & \sum_{r=0}^{R-1} t'[r+p_1]t'[r+p_1] & \cdots & \sum_{r=0}^{R-1} t'[r+p_9]t'[r+p_1] \\ \vdots & \vdots & & \vdots \\ \sum_{r=0}^{R-1} t'[r+p_0]t'[r+p_9] & \sum_{r=0}^{R-1} t'[r+p_1]t'[r+p_9] & \cdots & \sum_{r=0}^{R-1} t'[r+p_9]t'[r+p_9] \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_9 \end{bmatrix} = \begin{bmatrix} \sum_{r=0}^{R-1} s[r]t'[r+p_0] \\ \sum_{r=0}^{R-1} s[r]t'[r+p_1] \\ \vdots \\ \sum_{r=0}^{R-1} s[r]t'[r+p_9] \end{bmatrix}$$ 수학식 2

[0027] $$t'[r+p_n]=t[r+p_n]+t[r+p_{18-n}]$$ 수학식 3

[0028] 수학식 2의 위너-홉 방정식에서 필터 계수 C0~C9를 계산하기 위해 가우시안 제거 방법을 이용한다. 기본적인 연립방정식을 이용한 가우시안 제거 방법은 10개의 미지수인 필터 계수들을 계산하는데 연산량과 복잡도가 매우 많기 때문에, 자기 상관행렬인 10×10 행렬 부분의 대칭적인 특성을 이용한 촐레스키 분해법을 이용하여 효율적으로 필터 계수들을 계산할 수 있다.

[0029] 도 3은 촐레스키 분해법을 이용한 가우시안 제거 방법을 보인 예시도이다.

[0030] 촐레스키 분해법을 이용한 전체 연산 과정은 도 3과 같다. 도 3에서 첫 번째 단계인 팩토라이제이션(Factorization) 연산 단계는 10×10 행렬 A를 수학식 4와 수학식 5를 이용하여 10×10 행렬 U와 U의 전치행렬 UT를 계산한다. 두 번째 단계인 포워드(Forward) 연산 단계는 팩토라이제이션 연산 단계에서 계산한 행렬 UT를 이용하여 10×1 벡터 d를 계산한다. 마지막으로 백(Back) 연산 단계는 팩토라이제이션 연산 단계에서 계산했던 행렬 U와 포워드 연산 단계에서 계산한 벡터 d를 이용하여 최종 해인 10×1 벡터 x를 계산한다.

[0031] 수학식 4, 5는 10×10 행렬 U와 U의 전치행렬 UT를 계산하기 위한 촐레스키 분해법의 방정식이다. 수학식 4, 5에서 a는 10×10 행렬 A에서 i위치와 j위치에 해당하는 행렬 요소를 의미하고, j는 i+1 위치부터 시작하여 행렬의 최대 크기까지인 10을 의미한다.

[0032] $$u_{ii}=\sqrt{a-\sum_{k=1}^{i-1} u_{ki}\times u_{ki}}$$ 수학식 4

[0033] $$u_{ij}=\frac{a_{ij}-\sum_{k=1}^{i-1} u_{ki}\times u_{kj}}{u_{ii}}$$ 수학식 5

[0034] 도 4는 본 발명의 일실시예에 따른 적응적 루프 필터의 구성을 보인 블록도이다.

[0035] 본 발명에서 제안하는 ALF 하드웨어 구조는 수행 사이클을 감소시키기 위해 촐레스키 분해의 특징적인 연산 관계를 분석하여 병렬적으로 연산을 수행하기 위한 2단 파이프라인 구조로 설계하였고, 촐레스키 분해에 사용되는

루트 연산은 연산량 및 연산 시간을 감소시키기 위해 멀티플렉서와 뺄셈기, 비교기를 이용하여 설계하였다. 도 4는 제안하는 ALF 하드웨어 구조를 나타낸다.

[0036] 본 발명에서 제안하는 구조는 팩토라이제이션(Factorization_top) 모듈(110), 포워드(Forward_top) 모듈(120), 백(Back_top) 모듈(130)로써 총 3개의 상위 모듈로 구성되며, 각 모듈 내부에 루트(Root_top) 모듈, 디바이더(divider_array) 모듈 등의 하위 모듈로 구성된다. 적응적 루프 필터는 행렬 U를 계산하고 각 행에 대한 루트 연산 값과 상기 행렬 U를 출력하는 팩토라이제이션 모듈(110), 상관정보와 행렬 U를 입력받아 벡터 d를 계산하는 포워드 모듈(120) 및 행렬 U와 벡터 d를 입력받아 벡터 x를 계산하는 백 모듈(130)을 포함한다.

[0037] 팩토라이제이션 모듈(110)은 내부에 루트 연산을 수행하는 루트 모듈과 나눗셈 연산을 수행하고 레지스터 배열에 저장하는 디바이더 모듈로 구성되어 있고, 10×10 행렬 U를 계산하는 기능을 수행한다. 포워드 모듈(120)은 팩토라이제이션 모듈(110)에서 계산한 10×10 행렬 U의 값을 입력받아 10×1 벡터 d를 계산하는 기능을 수행한다.

[0038] 백 모듈(130)은 쉬프트와 라운딩을 연산하는 쉬프트라운드(Shift_Round_top) 모듈과 디바이더(divider_array) 모듈로 구성되어 있고, 팩토라이제이션 모듈(110)에서 계산한 10×10 행렬 U의 값과 포워드 모듈(120)에서 계산한 10×1 벡터 d의 값을 입력받아 최종 해인 10×1 벡터 x, 즉 필터 계수 C0~C9을 계산한다. 본 발명에서 제안하는 구조는 10×10 행렬 U를 계산하는 팩토라이제이션 모듈(110)과 10×1 벡터 d를 계산하는 포워드 모듈(120) 간의 촐레스키 분해법을 이용한 가우시안 제거 방법의 특징적인 연산 관계에 따라 10×10 행렬 U의 n번째 행의 계산이 완료되면, 10×1 벡터 d의 n-1번째 행을 계산할 수 있다.

[0039] 도 5는 2단 파이프라인 구조를 보인 예시도이다.

[0040] 팩토라이제이션 모듈(110)과 포워드 모듈(120)은 도 5와 같이 2단 파이프라인 구조로 설계하여 병렬적으로 동작하도록 구현함으로써, 수행 사이클을 감소시켰다.

[0041] 도 6은 도 4의 팩토라이제이션 모듈(110)의 구성을 보인 블록도이다.

[0042] 본 발명에서 제안하는 하드웨어 구조에서 첫 번째로 연산을 수행하는 팩토라이제이션 모듈(110)은 도 6과 같이 루트(Root_top) 모듈(610), 디바이더(divider_array) 모듈(620), 곱셈뺄셈(Mul_sub) 모듈(630)로 구성된다. 팩토라이제이션 모듈(110)은 행렬 U와 상관정보에 루트 연산을 수행하는 루트 모듈(610), 상관정보를 저장하는 레지스터, 루트 모듈(610)의 결과값과 상관정보에 나눗셈 연산을 수행하는 디바이더 모듈(620) 및 디바이더 모듈(620)의 결과값에 곱셈기와 뺄셈기로 행렬 U를 계산하는 곱셈뺄셈 모듈(630)을 포함한다.

[0043] 팩토라이제이션 모듈(110)은 수학식 4, 5를 이용하여 10×10 행렬 U를 계산하고, 각 행에 대한 루트 연산 결과값은 Root_out 신호로, 10×10 행렬 U의 행렬 요소들을 Fact_out 신호로 출력한다. 팩토라이제이션 모듈(110)은 상관관계 정보인 E_corr_data 신호를 입력받아 10×10 행렬 U를 계산하고, 루트 연산을 수행하는 루트 모듈(610)은 상관관계 정보에 따라 최대 32비트까지 연산 가능하며 루트 연산의 결과 값을 출력하기까지 15 사이클을 소모한다. 루트 연산의 결과 값은 디바이더 모듈(620)에서 나눗셈 연산을 수행할 때 나눗셈의 분모 역할을 수행하고, 나눗셈 연산을 수행한 결과 값은 각각의 레지스터 배열에 저장된다. 곱셈뺄셈 모듈(630)은 곱셈기와 뺄셈기 등의 연산기로 구성되어 있고, 결과 값은 루트 모듈(610)의 입력으로 사용되며 디바이더 모듈(620)에서 나눗셈 연산을 수행할 때 나눗셈의 분자 역할을 수행한다.

[0044] 도 6에서 루트 모듈(610)은 루트 연산을 수행하는 모듈로써, 입력 값의 비트 수가 적을 경우 룩업 테이블(LUT: Look up table)을 이용하여 비교적 간단한 구조로 설계할 수 있다. 그러나 ALF는 필터 계수를 추출하기 위한 입력 정보가 최대 32비트에 해당하는 상당히 큰 값이기 때문에, 룩업 테이블을 이용하여 설계할 경우 하드웨어 구조가 복잡해지고 면적과 연산량이 증가하는 단점이 있다. 따라서 루트 연산을 효율적으로 계산하기 위해 하드웨어 설계에 적합한 알고리즘을 분석하여 복잡도와 면적, 연산 수행시간을 감소시킨 루트 연산 하드웨어 구조를 제안한다.

[0045] 도 7은 루트 연산 방법을 보인 예시도이다.

[0046] 도 7은 하드웨어 설계를 위한 루트 연산 방법을 나타낸 것이다. 도 7에서는 '93' 이라는 값을 샘플로 루트 연산 방법을 이용한 연산 과정을 나타낸 것으로, 루트 연산 방법을 이용한 결과 값과 실제 결과 값의 차이는 소수점 부분에서 미세하게 존재하지만 실제 사용되는 값은 정수 부분만을 사용하기 때문에 결과 값에는 영향을 미치지 않는다.

[0047] 도 8은 도 4의 루트 연산 하드웨어 구조를 보인 블록도이다.

[0048] 도 8은 도 7의 루트 연산 방법을 멀티플렉서와 뺄셈기, 비교기만을 이용한 루트 연산 하드웨어 구조를 나타낸 것이다. 도 8에서 루트 모듈은 비교기(Comparator2~23)(810), 뺄셈(Sub_decision) 모듈(820)로 구성된다. 루트 모듈은 입력 신호와 뺄셈 모듈(820)의 결과값에 2비트 01을 오른쪽에 결합한 값을 비교하는 비교기(810) 및 입력 신호와 비교기(810)의 결과값을 뺄셈하는 뺄셈 모듈(820)을 포함한다.

[0049] 비교기(Comparator2~23)(810)은 Root_in 입력 신호와 뺄셈(Sub_decision) 모듈(820)의 결과 값인 Sub_out 신호에 2비트 '01'을 오른쪽에 결합한 값과 비교하는 기능을 수행한다. 결과 값인 Comp_out 신호는 레지스터에 저장되고, 입력 신호 Root_in의 마지막인 LSB(Least Significant Bit)까지 연산을 완료하면 레지스터에 저장했던 결과 값을 모두 결합하여 Root_out 신호로 출력한다.

[0050] 도 9는 도 4의 포워드 모듈(120)의 구성을 보인 블록도이다.

[0051] 본 발명에서 제안하는 하드웨어 구조에서 첫 번째로 동작하는 팩토라이제이션 모듈(110) 간의 파이프라인 구조로써 병렬적으로 동작하는 포워드 모듈(120)은 도 9와 같이 디바이더 모듈(910), 곱셈뺄셈 모듈(920)로 구성된다. 포워드 모듈(120)은 상관정보와 행렬 U를 입력받아 곱셈 뺄셈하는 곱셈뺄셈 모듈(920), 상관정보를 저장하는 레지스터 및 곱셈뺄셈 모듈(920)의 결과값과 레지스터의 상관정보를 나눗셈 연산을 수행하는 디바이더 모듈(910)을 포함한다.

[0052] 포워드 모듈(120)은 상관관계 정보인 y_corr_data 신호와 팩토라이제이션 모듈(110)에서 계산한 10×10 행렬 UT 결과 값을 입력 받아 10×1 벡터 d의 결과를 For_out 신호로 출력한다. 내부의 곱셈뺄셈 모듈(920)은 곱셈기와 뺄셈기의 연산기로 구성되고, 디바이더 모듈(910)에서 곱셈뺄셈 모듈(920)의 결과값은 나눗셈 연산을 수행할 때 나눗셈의 분자 역할을 수행한다.

[0053] 도 10은 도 4의 백 모듈(130)의 구성을 보인 블록도이다.

[0054] 본 발명에서 제안하는 하드웨어 구조의 백 모듈(130)은 도 10과 같이 디바이더(divider_array) 모듈(1010), 곱셈뺄셈 모듈(Mul_sub)(1020), 쉬프트라운드(Shift_Round_top) 모듈(1030)로 구성된다. 백 모듈(130)은 행렬 U와 벡터 d를 곱셈 뺄셈하는 곱셈뺄셈 모듈(1020), 행렬 U를 저장하는 레지스터, 곱셈뺄셈 모듈(1020)의 결과값과 레지스터의 행렬 U를 나눗셈 연산을 수행하는 디바이더 모듈(1010) 및 디바이더 모듈(1010)의 나머지 값을 쉬프트와 반올림 연산을 수행하여 필터 계수값을 생성하는 쉬프트라운드 모듈(1030)을 포함한다.

[0055] 백 모듈(130)은 팩토라이제이션 모듈(110)에서 계산한 10×10 행렬 U 결과 값과 포워드 모듈(20)에서 계산한 10×1 벡터 d 결과 값을 입력 받아 10×1벡터 x의 결과를 Back_out 신호로 출력한다. 내부의 곱셈뺄셈 모듈(1020)은 곱셈기와 뺄셈기 등의 연산기로 구성되어 있고, 디바이더 모듈(1010)에서 나눗셈 연산을 수행할 때 나눗셈의 분자 역할을 수행한다. 또한, 백 모듈(130) 내부의 쉬프트라운드 모듈(1030)은 디바이더 모듈(1010)에서 계산한 나머지 값을 쉬프트와 반올림 연산을 수행하여 최종적인 필터 계수 값을 생성한다.

[0056] 상기에서는 본 발명의 바람직한 실시예를 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

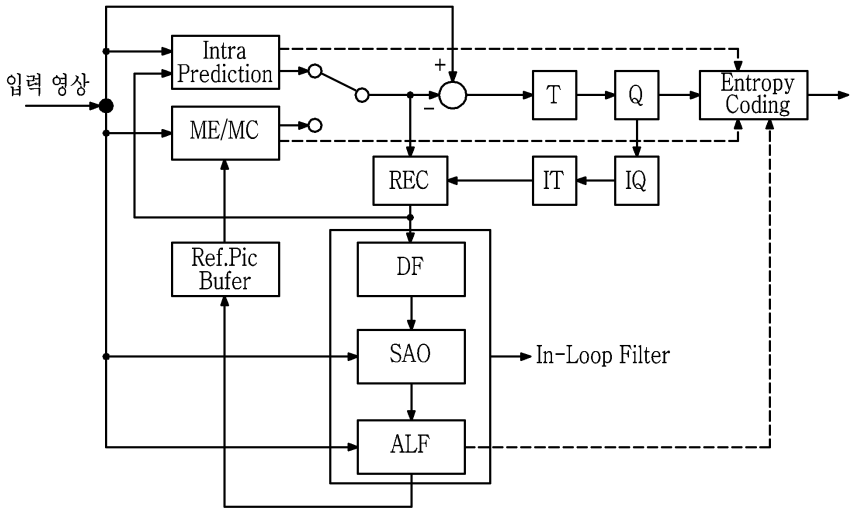
부호의 설명

[0057] 110: 팩토라이제이션 모듈 120: 포워드 모듈

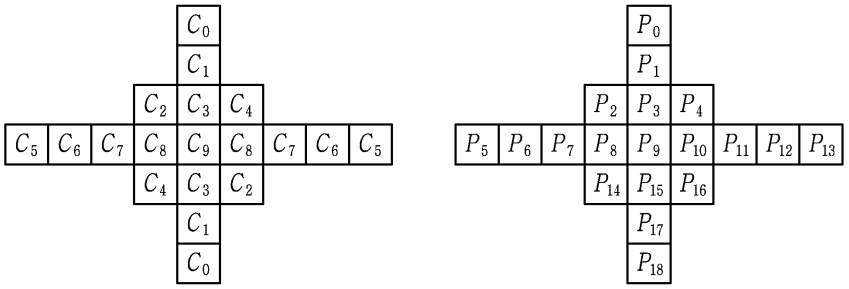
130: 백 모듈

도면

도면1

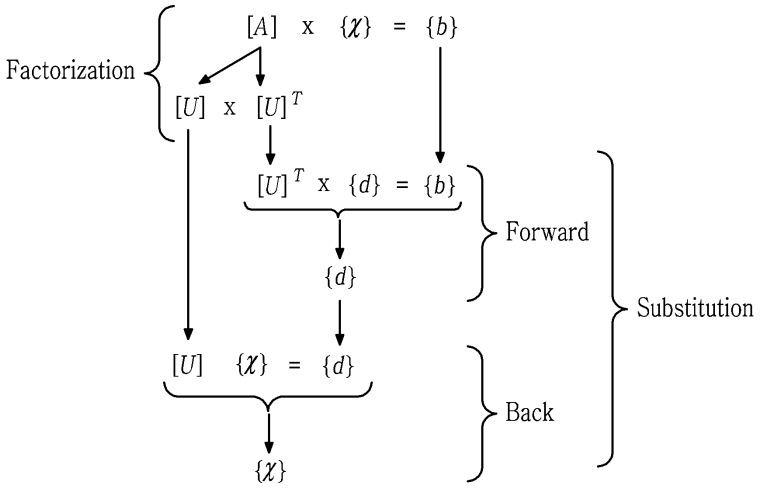


도면2

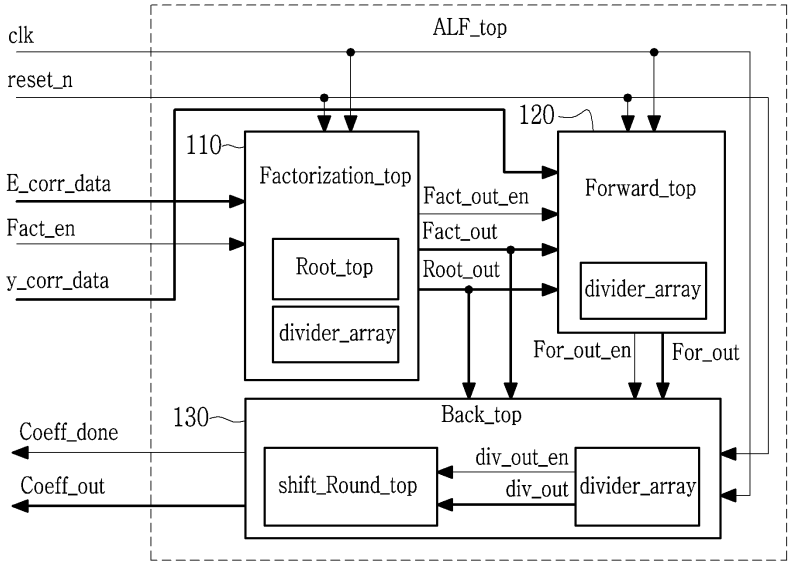


(a) Filter coefficients (b) Reconstructed pixel position sffsets

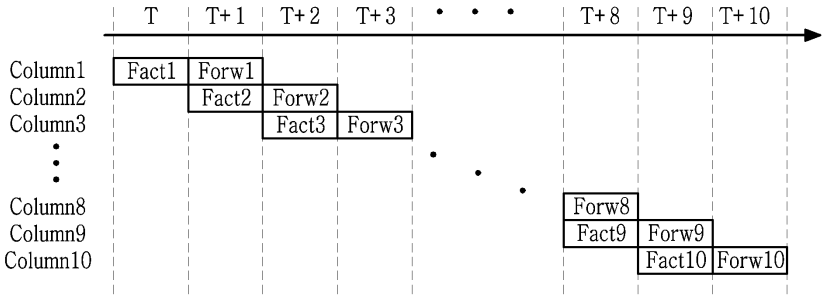
도면3



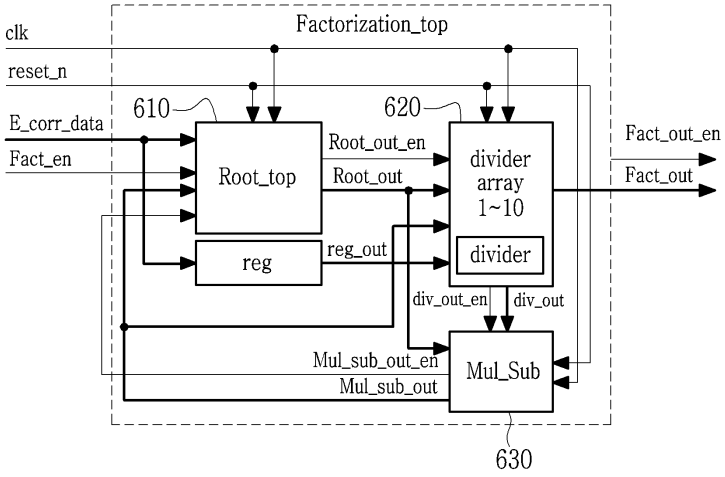
도면4


ALF_top
clk
reset_n
E_corr_data
Fact_en
y_corr_data
110
Factorization_top
Root_top
divider_array
Fact_out_en
Fact_out
Root_out
120
Forward_top
divider_array
For_out_en
For_out
130
Back_top
Coeff_done
Coeff_out
shift_Round_top
div_out_en
div_out
divider_array

도면5

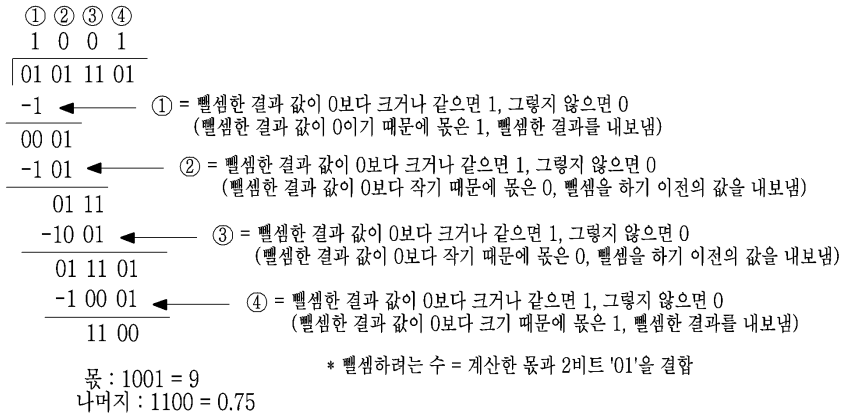

T
T+1
T+2
T+3
T+8
T+9
T+10
Column1
Column2
Column3
Column8
Column9
Column10
Fact1
Forw1
Fact2
Forw2
Fact3
Forw3
Forw8
Fact9
Forw9
Fact10
Forw10

도면6

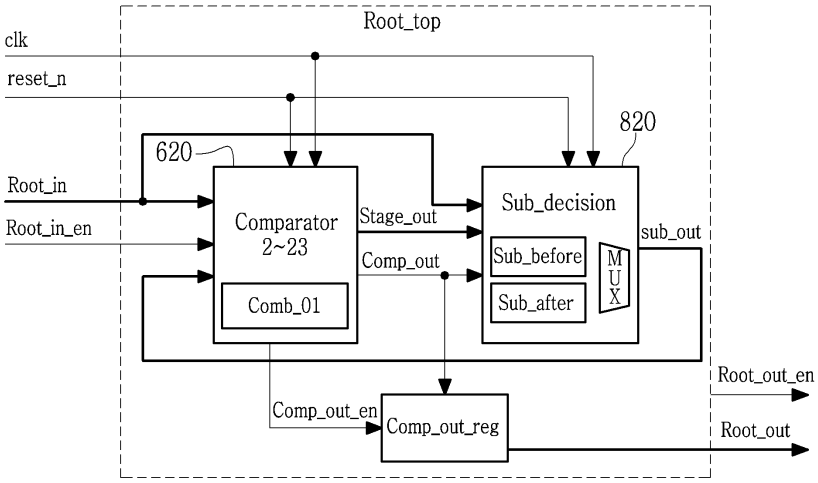

Factorization_top
clk
reset_n
E_corr_data
Fact_en
610
Root_top
reg
Root_out_en
Root_out
reg_out
620
divider
array
1~10
divider
div_out_en
div_out
Fact_out_en
Fact_out
Mul_sub_out_en
Mul_sub_out
Mul_Sub
630

도면7

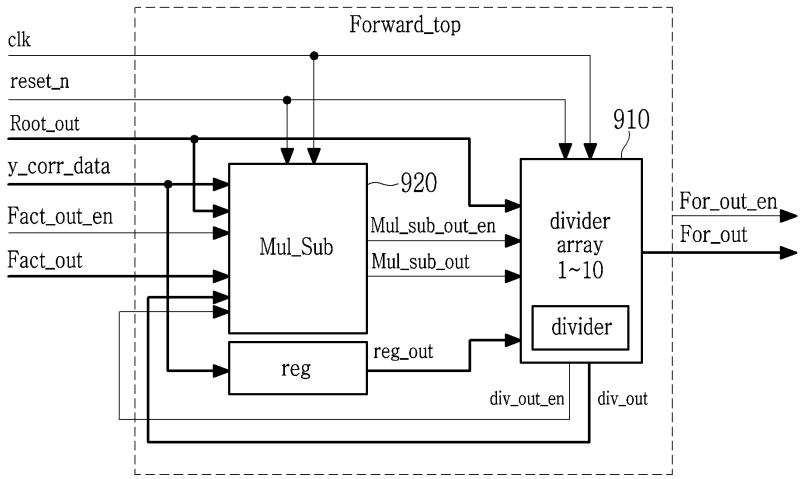
Ex) $\sqrt{93}$ = 9.64365076 ... 93 → 01 01 11 01



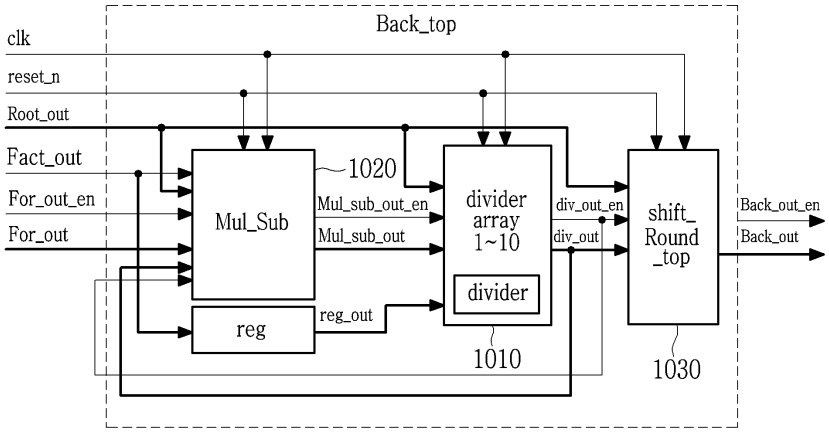
도면8



도면9



도면10


Back_top
clk
reset_n
Root_out
Fact_out
For_out_en
For_out
Mul_Sub
1020
Mul_sub_out_en
Mul_sub_out
reg
reg_out
divider
array
1~10
divider
1010
div_out_en
div_out
shift_
Round
_top
1030
Back_out_en
Back_out