論文97-34C-11-11

뉴런의 생성 및 병합 학습 기능을 갖는 자기 조직화 신경망을 이용한 n-각형 공업용 부품의 중심추정

(Center estimation of the n-fold engineering parts using self organizing neural networks with generating and merge learning)

成 孝 慶 *, 崔 興 文 *

(Hyo-Kyung Sung and Heung-Moon Choi)

요 약

본 논문에서는 생성 및 병합기능을 갖는 자기 조직화 신경망을 이용하여 n-각형 부품의 중심을 강건하게 추출할 수 있는 방법을 제안하였다. 신경망을 이용하여 n-각형의 중심을 추출하기 위하여 관심 부품의 분할된 테두리를 직선으로 근사화한 후, 근사화된 직선과 기준점사이에서 정의되는 코사인정리를 이용하여 해당되는 테두리가 만드는 중심을 ($\rho-\theta$) 인수벡터로 표현하였다. 이렇게 정의된 인수 벡터들은 자기 조직화 신경망의 뉴런의 생성 및 병합 학습에 의하여 최종적으로 관심을 가진 n-각형의 중심을 추출한다. 병합기능에 적응적인 학습율을 채용하고, 생성과정에 연계된 자기 조절 활성함수를 사용함으로써 자기 조직화 신경망에 의한 중심 추출과정을 가속화시켰다. 여러 공업용 부품에 대한 컴퓨터 시뮬레이션 결과 추정된 중심의 오차 분포 및 입력된 영상의 테두리 정보를 모를 경우에도 제안된 방법이 부품의 중심을 양호하게 추출함을 확인하였다.

Abstract

A robust center estimation technique of n-fold engineering parts is presented, which use self-organizing neural networks with generating and merging learning for training neural units. To estimate the center of the n-fold engineering parts using neural networks, the segmented boundaries of the interested part are approximated to straight lines, and the temporal estimated centers by the cosine theorem which formed between the approximated straight line and the reference point, are indexed as ($\rho-\theta$) parametric vectors. Then the entries of parametric vectors are fed into self-organizing neural network. Finally, the center of the n-fold part is extracted by mean of generating and merging learning of the neurons. To accelerate the learning process, neural network uses an adaptive learning rate function to the merging process and a self-adjusting activation to generating process. Simulation results show that the centers of n-fold engineering parts are effectively estimated by proposed technique, though not knowing the error distribution of estimated centers and having less information of boundaries.

* 正會員, 慶北大學校 電子電氣工學部

(School of Electronic & Electrical Engineering, Kyungpook National University)

接受日字:1997年6月12日, 수정완료일:1997年10月17日

Ⅰ. 서 론

컴퓨터시각을 이용하여 인쇄회로기판(printed circuit board: PCB)의 불량여부 검사 및 부품삽입과 같은 자동화 시스템에서 원형을 포함한 n-각형 부품

또는 물체의 중심 및 그 크기를 추정하는 연구가 많이 이루어지고 있다[1-8]. 일반적으로 물체의 위치추출에는 다양한 종류의 Hough변환이 이용되고 있다[2-5]. 그러나 Hough변환은 부적절한 조명방법 등으로 추출된 물체의 테두리가 왜곡될 수 있을 뿐만 아니라 그 에지 폭 및 방향이 Hough공간의 매개인수 누적에 부적절한 영향을 미쳐 원하는 매개인수값을 나타내는 첨두치를 효과적으로 선택하기 위해 사용하는 에지의 방향 정보가 잡음의 영향을 많이 받아 하나의 물체가 2개이상의 물체로 추출되는 경우가 많다. 이러한 누적 매개인수 영역에서 첨두치를 효과적으로 찾기 위하여 Leavers 등[3]은 정합여파기법을 사용하였으나 정합여파기를 설계하기 위하여 입력 영상과 누적공간에서 추출 가능한 특징의 차수정보를 미리 알아야 하며, 실제 첨두치가 여러 개의 누적 셀에 분산되어 나타나거나 추출이 되지 않을 수도 있다. Ibrahim 등[4,5]은 이진화 과정을 거치지 않고 영상을 미분하고 미분된 각 화소들이 만들어 내는 중심 매개인수들에 원영상의 명암도를 가중하여 첨두치를 찾는 가중 Hough변환(weighted Hough transform)을 제안하였으나, 영상이 잡음에 오염될 경우 미분에 명암도 가중치를 부가할 경우 원하는 목적을 이루지 못할 수 있다. 또한, Wallice 등[6-7]은 누적 매개인수공간내의 첨두치를 효과적으로 추출하기 위하여 누적된 확률 변수(random variable)의 분포에 근거한 인정 및 거절에 대한 가설(t-test)을 기반으로 물체의 추출을 시도하였다. 그러나 이러한 가설에 근거한 성능은 매개인수 분포의 함수로 주어지는 해당 매개인수에 대한 여확률(probability of rejection)에 대한 급수함수(power function)에 의해 결정된다. 한편 McMichael[8]은 BARTIN(BAyesian Real-TIme Network)이라는 신경망을 제안하고, 이를 이용하여 매개인수 누적공간의 누적분포를 확률적으로 분석하여 첨두치를 찾는 방법으로 공업용 물체의 매개인수를 추정하였다. 이 방법은 실제 응용에서 높은 성능을 나타내지만 이용할 수 있는 원시 데이터에 대한 정보가 부족하거나 추출된 매개인수의 확률적인 분포를 정확히 모를 경우 만족할 만한 결과를 얻지 못할 수 있다.

본 논문에서는 생성 및 병합기능을 갖는 자기 조직화 신경망을 이용하여 n-각형 부품의 중심을 강건하게 추출할 수 있는 방법을 제안하였다. 신경망을 이용하여 n-각형의 중심을 추출하기 위하여 관심 부품의 분할된 테두리를 직선으로 근사화한 후, 근사화 된 직선과 기준점사이에서 정의되는 코사인정리를 이용하여 해당되는 테두리가 만드는 중심을 ($\rho-\theta$) 인수벡터로 표현하였다. 이렇게 정의된 인수 벡터들은 자기 조직화 신경망의 뉴런의 생성 및 병합 학습에 의하여 최종적으로 관심을 가진 n-각형의 중심을 추출한다. 입력 매개인수를 벡터의 형태로 표현함으로써 하나의 근사화 직선에 의해 추정되는 중심매개인수의 크기 및 방향을 쉽게 알 수 있도록 하였다. 중심 추정을 위해 사용된 제안된 신경망은 신경망의 분류기능(clustering)을 이용하여 뉴런을 생성시키고, 유사한 입력을 하나의 출력으로 모아주는 일반화 성질을 이용하여 서로 유사한 중심 매개인수 벡터들을 합병하도록 하였다. 또한, 병합기능에 적응적인 학습율을 채용하고, 생성과정에 연계된 자기 조절 활성함수를 사용함으로써 선형 근사화과정에서 발생하는 정보의 손실 및 잡음에 의한 오차 등에 둔감하게 n-각형 부품의 중심을 추정할 수 있다.

Ⅱ. 자기 조직화 신경망을 이용한 n-각형 부품의 중심 추정

제안한 자기 조직화 신경망을 이용한 n-각형 부품의 중심 추정은 그림 1에서 보는 바와 같이 n-각형을 분할 근사화하는 직선들에 코사인 정리를 적용하여 근사화 직선별로 추정된 중심 매개인수들을 ($\rho-\theta$) 좌표의 인수 벡터의 형태로 변환한 후, 이들 매개인수 벡터들을 자기 조직화 신경망에 입력하고, 신경망의 적응적인 뉴런 생성 및 병합기능을 이용하여 n-각형 부품의 중심을 최종 추정한다.

1. 근사화 직선을 이용한 중심 매개인수 추출

n-각형의 분할 근사화 직선과 중심 매개인수의 크기 ρ 및 방향 θ 와의 관계를 n-각형 및 원에 대하여 그림 2(a) 및 그림 2(b)에 각각 도시하였다.

P_i를 극좌표(ρ , θ)에 존재하는 화소점(ρ_i, θ_i) 이라 할 때 n-각형 물체를 선형 근사화 하는 직선은 그림 2(a)와 같이 일정한 상수 ρ에 대하여 다음 식을 만족하는 (ρ_o, θ_o)를 찾는 것[4,5]으로 귀결된다.

$$\rho_o = \rho \cdot \cos(\theta - \theta_o) \tag{1}$$

여기서 ρ_o 및 θ_o는 원점을 지나면서 직선과 직교하는

법선의 길이 및 각도를 의미한다.

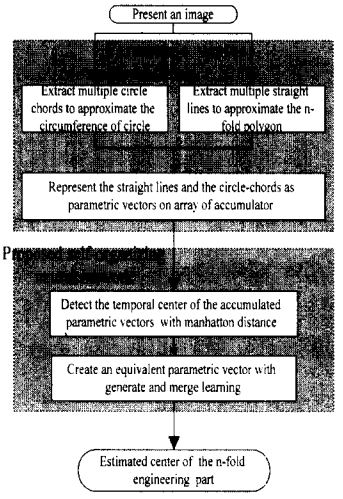


그림 1. 중심 추정을 위한 처리 구조
Fig. 1. The overall architecture for estimating center position.

한편 원의 경우에는 그림 2(b)에서 보는 바와 같이 데두리상의 두 점 $P_i(\rho_i,\ \theta_i)$ 및 $P_j(\rho_j,\ \theta_j)$가 이루는 근사화 직선(현) k_{ij}의 가운데 점 P_o를 지나면서 그 현에 직교하는 법선상에 존재한다.

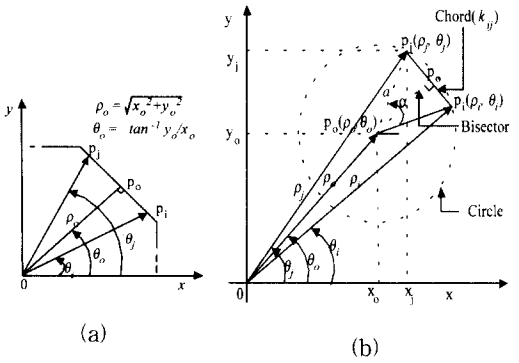


그림 2. 극좌표계에서의 (a) n-각형의 일부 및 (b) 원의 구조
Fig. 2. The geometry of (a) partial portion of the n-fold polygon and (b) circle in polar coordinate.

즉, 원점, 원의 중심 $P_o(\rho_o,\ \theta_o)$ 및 원주상의 한 점 $P_i(\rho_i,\ \theta_i)$가 만드는 삼각형 ΔOP_oP_i 와 원주상의 다른 점 $P_j(\rho_j,\ \theta_j)$가 만드는 ΔOP_oP_j에 대하여 각각 코사인정리를 적용하면 다음과 같은 관계가 성립한다.

$$\begin{aligned}\rho_o^2+\rho_i^2-2\rho_o\rho_i\cos(\theta_o-\theta_i)&=a^2\\ \rho_o^2+\rho_j^2-2\rho_o\rho_j\cos(\theta_o-\theta_j)&=a^2\end{aligned}\tag{2}$$

그림 2(b)에 도시한 현 k_{ij}의 양 끝점은 동일한 원주상에 있으므로 현과 현의 중심을 지나는 법선과의 관계를 이용하여 원의 중심을 나타내는 매개인수를 유추할 수 있다.

$$\begin{aligned}\rho_i^2-\rho_j^2&=2\rho_i\rho_o\cos(\theta_o-\theta_i)-2\rho_j\rho_o\cos(\theta_o-\theta_j)\\&=2\rho_o\cos\theta_o\{\rho_i\cos\theta_i-\rho_j\cos\theta_j\}+2\rho_o\sin\theta_o\{\rho_i\sin\theta_i-\rho_j\sin\theta_j\}\end{aligned}\tag{3}$$

따라서 이를 바탕으로 원의 중심 x_o 및 y_o는

$$\begin{aligned}x_o&=\rho_o\cos\theta_o=(\rho_i^2-\rho_j^2)/2(\rho_i.\cos\theta_i-\rho_j.\cos\theta_j)\\ y_o&=\rho_o\sin\theta_o=(\rho_i^2-\rho_j^2)/2(\rho_i.\sin\theta_i-\rho_j.\sin\theta_j)\end{aligned}\tag{4}$$

와 같이 주어지므로 극좌표계상의 $(\rho_o,\ \theta_o)$를 구하면 해당 현들에 의해 분할 근사화 되는 원에 대한 매개인수를 추정할 수 있다. 따라서 다각 근사화된 n-각형은 매개인수 누적망에 누적된 n개의 독립적인 직선에 대한 매개인수$(\rho_o,\ \theta_o)$들을 평균하는 등의 통계적인 방법으로 n-각형 물체의 중심 및 크기를 추정할 수 있으며, n개의 화소로 이루어진 원형 물체의 경우에도 $nC_2 = n(n-1)/2$ 개의 현들의 누적매개인수로부터 통계적으로 그 중심 및 크기를 측정할 수 있다[13].

2. n-각형 물체의 중심 추출을 위한 자기 조직화 신경망

일반적으로 자기 조직화 신경망[9-11]은 입력되는 벡터들의 유사도가 비슷하도록 입력벡터들을 분류해 주는 기능이 있기 때문에 입력 인수들의 분포 분석에 응용할 수 있다. n-각형을 분할 근사화하는 직선에 코사인 정리를 적용하여 추출된 중심 매개인수벡터의 분포를 분석함으로써 그 중심을 추정할 수 있다. 중심 매개인수벡터의 분포를 분석하기 위한 자기 조직화 신경망의 구조는 그림 3과 같이 입력층과 출력층이 완전 연결된 전방향 망(feedforward network)으로 근사화 직선들로부터 추출된 M개의 크기성분 ρ를 가로축에 배열하고, N개의 각도성분 θ를 세로축에 배열할 수 있도록 MN크기의 입력층과 하나의 뉴런이 학습과정에서 일반, 우세 및 승리 누적 뉴런의 기능을 하는 뉴런들간에 측방향 억제신호로 완전 연결된 2차원의 출력층으로 구성된다.

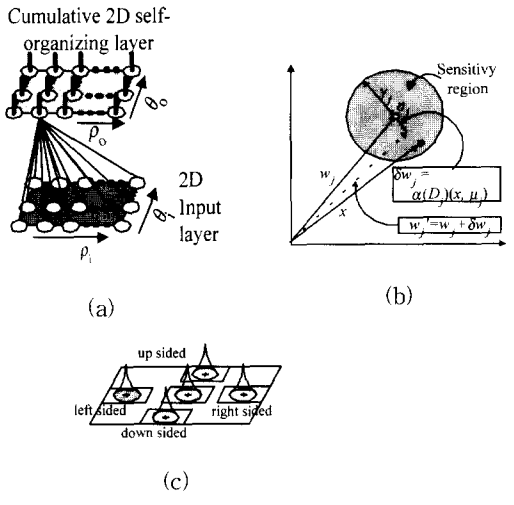


그림 3. (a) 매개인수 분포 분석에 사용된 자기 조직화 신경망의 구조 (b) 활성화된 뉴런의 반응 영역 및 연결세기 갱신 그리고 (c) 누적 자기 조직화 층

Fig. 3. (a) The architecture of the cumulative self-organizing neural networks for analysis of parameter distribution, (b) sensitivity region and the weight update of the activated neuron, and (c) cumulative 2D self-organizing layer.

또한 출력층의 뉴런 n_j는 활성 기준(attraction threshold) γ_j를 가지며, 입력층의 뉴런 i_i사에 MN차의 참조 벡터 μ_{ij}로 완전 연결되어 있다. 활성기준은 학습과정에서 입력 인수 벡터가 가장 민감하게 반응하는 뉴런의 반응 영역(sensitivity region) $A(n_j)$를 결정한다.

제안한 신경망은 입력 인수 벡터에 가장 가까운 일반 뉴런을 우세 뉴런 n_d로 할당한다. 이 때 이 우세 뉴런이 근처에 있는 승리 누적 뉴런 n_c의 반응영역내에 포함되면 그 거리에 따른 흡인율(학습율)을 사용하여 승리 누적 뉴런의 가중치를 우세 뉴런쪽으로 학습함과 동시에 해당 승리 누적 뉴런의 활성 회수를 1 증가시키며, 그렇지 않을 경우에는 n_d를 새로운 승리 누적 뉴런으로 활성화시킨다. 그러나 승리 누적 뉴런의 생성에 의해 인접 누적 뉴런의 반응 영역이 중복되면 두 누적 뉴런을 하나의 누적 뉴런으로 병합하여 누적 뉴런의 누적도 및 그 세력을 강화시킨다.

제안한 신경망은 인수 벡터 $x_i(\rho_i, \theta_i)$에 대하여 먼저 뉴런 j의 반응영역 $A(n_j)$내의 뉴런들과의 맨하탄 거리(Manhattan) 거리

$$D_j = \sum_{k \in A(n_j)} |x_i - \mu_{ik}(n)|$$

를 구한 후 그 길이가 가장 짧은 뉴런 n_j를 인 우세 뉴런 n_d에 대하여 흡인율(attraction ratio) χ를 가지고 다음과 같이 연결세기 μ_{ij}를 가변한다.

$$\begin{aligned} \mu_{ij}(n+1) &= \mu_{ij}(n) + \chi(D_j)\{x_i - \mu_{ij}(n)\}, && \text{if } n_j = n_a \\ &= \mu_{ij}(n) && \text{otherwise} \end{aligned} \quad (5)$$

이때, 반응 영역 내에 존재하는 뉴런의 학습정도를 나타내는 흡인율 $\chi(D_j)$는 입력 벡터와 우세 뉴런과의 거리 및 우세 뉴런의 활성화 회수에 따라 적응적으로

$$\chi(D_j) = D_j * \exp(-D_j^{2\zeta}) \quad (6)$$

와 같이 결정되며, 거리 D 및 활성 회수 ζ에 대하여 그림 4와 같은 특성을 나타난다.

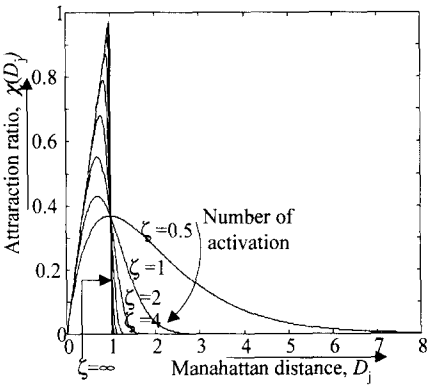


그림 4. 반응정도 결정을 위한 흡인 함수

Fig. 4. Attraction function to decide the attraction ratio.

이 때, 제안된 신경망은 누적 뉴런의 활성화되는 회수 ζ가 증가함에 따라 활성 기준 γ를

$$\gamma = \frac{\gamma_{in}}{\zeta + 1} \quad (7)$$

와 같이 선형적으로 감소시켜 매개인수의 분포가 조밀한 영역에 있는 우세 뉴런의 세력을 점점 강화하여 잡음이나 입력된 원시 데이터에 대한 정보의 부족 등에 대한 영향을 줄여 나아간다.

식 (5)를 뉴런의 강화가 일어나는 우세뉴런 n_d가 승리 누적 뉴런 n_c의 반응 영역 $A(n_c)$에 존재할 경우와 새로운 누적 뉴런이 생성되는 그렇지 않을 경우에 대하여 다음과 같이

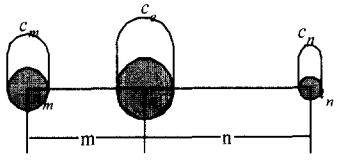
$$\begin{aligned}\mu_{ij}(n+1) &= \mu_{ij}(n)+\chi(D_j)\{x_i-\mu_{ij}(n)\} && n_j=n_w \text{ and } n_j\in A(n_c)\\ &= \mu_{ij}(n)+\alpha(D_j)\chi(D_j)\{x_i-\mu_{ij}(n)\} && n_j=n_w \text{ and } n_j\notin A(n_c)\\ &= \mu_{ij}(n) && \text{otherwise}\end{aligned} \tag{8}$$

학습과정을 재 정의하였다. 여기서, 일반 뉴런 n_j 의 활성을 억제하기 위한 측방향 억제율 $\alpha(D_j)$는 거리 D_j 의 함수로 다음과 같다.

$$\alpha(D_j)=\frac{D_j}{\Gamma+D_j} \tag{9}$$

여기서, Γ는 뉴런 n_j의 억제량(amount of inhibition)을 나타낸다.

이렇게 $\alpha(D_j)$를 정함으로써 현재의 우세 뉴런 n_d=n_j와 승리 누적 뉴런 n_c간의 거리 D_j 가 가까워지면 $\alpha(D_j)$는 0에 근접하여 뉴런 n_j의 활성을 억제시킨다. 한편 입력된 벡터가 현재 활성화된 승리 누적 뉴런의 반응 영역 내에 존재하지 않으면, 즉 $D_j \rightarrow \infty$ 이면 $\alpha(D_j) \rightarrow 1$가 되어 입력된 매개인수 벡터가 새로운 물체에 대한 매개인수 벡터라 가정하고 우세 뉴런을 새로운 승리 누적 뉴런으로 할당된다. 이 때 입력된 벡터를 새로이 활성화된 누적 뉴런의 참조 벡터로 할당하고, 초기 활성값 γ_{in}으로 새로이 활성화된 누적 뉴런의 활성기준으로 정하고 이를 근거로 뉴런의 반응영역을 설정한다. 반면에 인접 누적 뉴런 n_m과 n_n의 반응 영역 중심 사이의 거리가 반응 영역의 크기보다 작을 경우[1]에는 자기 조직화 신경망은 하나의 n-각형에 대한 인수가 두 개의 다른 n-각형으로 추정되는 것을 방지하기 위하여 누적 뉴런 n_m과 n_n을 비활성화시키면서 두 누적 뉴런의 누적값을 그림 5와 같이 병합한다.



c : cumulative value
n : neuron(node)

그림 5. 등가 반응 영역 중심 및 등가 누적값 생성
Fig. 5. Creating the center position of the equivalent sensitive region.

그림과 같이 뉴런 n_m의 누적 값 c_m과 뉴런 n_n의 누적 값 c_n가 있을 때 등가 반응 영역 중심은 다음과 같은 관계를 만족하도록 구한다.

$$c_m \cdot m = c_n \cdot n \tag{10}$$

여기서 m 및 n은 각각 n_m과 n_e 및 n_n와 n_e 사이의 거리를 의미한다.

한편, 등가 누적값, c_e는 누적값 c_m 과 c_n을 단순히 더하며, 두 반응 영역의 중심간 거리, $d = m+n$에 대하여 등가 반응 영역 중심은 뉴런 n_m으로부터 다음 거리에 생성된다.

$$m=\frac{d}{1+\frac{c_m}{c_n}} \tag{11}$$

이렇게 등가 중심을 설정함으로써 누적값이 큰 누적 뉴런쪽에 치우쳐 새로운 누적 뉴런 $n^e{}_c$가 생성된다. 이 때, 등가 활성 기준 γ는 두 누적 뉴런의 세력과 거리에 따라 다음과 같이 정해진다.

$$\gamma_e=\frac{n\gamma_m+m\gamma_n}{m+n} \tag{12}$$

또한, 등가 맨하탄 거리 D_e를 다음과 같이

$$D_e=D_m+m \tag{13}$$

로 정하고, 이를 바탕으로 등가 흡인율 χ(D)와 등가 활성 억제량 α(D)가 결정된다.

전술한 자기 조직화 신경회로망의 뉴런 생성 및 병합을 통한 학습과정을 그림 6에 간단히 도시하였다.

그림 6(a)와 같이 우세한 벡터 n_d가 승리 누적 뉴런 n_c의 반응 영역 내에 있을 경우에는 그림 6(b)와 같이 매개인수 집합의 중심쪽으로 학습이 이루어진다. 또한, 그림 6(c)와 같이 입력된 벡터가 누적 뉴런의 반응 영역에 없을 경우에는 그림 6(d)와 같이 입력된 벡터를 새로운 누적 뉴런으로 설정하고 반응 영역을 결정하는 활성기준은 γ_{in}으로 둔다. 한편 그림 6(e)에 나타낸 바와 같이 인접 누적 뉴런의 반응 영역이 과도하게 겹칠 경우 두 뉴런을 병합하여 새로운 누적 뉴런을 만든다. 전술한 뉴런의 생성 및 병합 기능을 가지는 자기 조직화 신경망은 식 (6)으로 정의된 매개인수의 중심을 나타내는 승리 누적 뉴런 n_c의 활성 기준이

$\gamma \leq 1$

이 될 때 신경망은 수렴하게 되고, 학습을 종료한다.

1) $\Sigma|n_m-n_n| < \min(\gamma_m, \gamma_n)$

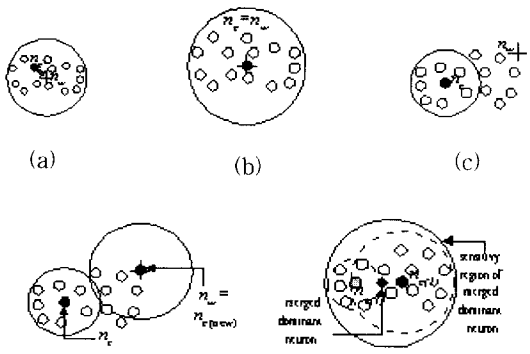


그림 6. 누적 분포의 중심 추정 체계 (a) 우세한 뉴런이 누적뉴런의 반응영역내에 있는 경우 (b) (a)의 분포에 대한 학습결과 (c) 우세한 뉴런이 누적뉴런의 반응영역내에 없는 경우 (d) (c)의 분포에 대해 새로운 누적 뉴런을 생성시킨 학습결과 그리고 (e) 인접 누적 뉴런의 반응 영역이 과도하게 겹쳐 두 누적뉴런을 합병시킨 학습결과

Fig. 6. Dynamics for creating the center of the cluster of parameters; In case of (a) the winning neuron is in the sensitive region of dominant neuron, (b) the training result for (a) which the dominant neuron is attracted by the winning neuron, (c) the winning neuron is not in the sensitive region of dominant neuron, (d) the training result for (c) which a new dominant neuron is generated by the winning neuron, and (e) the training result which a new dominant neuron is generated from the neighbor dominant neurons because of their sensitive regions are extremely overlapped.

결과적으로 승리 누적 뉴런이 입력된 인수 벡터 $x(\rho, \theta)$집합의 중심에 접근함에 따라 그 움직임이 둔화되어 누적 뉴런 n_c는 입력 벡터 $x(\rho, \theta)$집합의 중심 가까이에 상주하게 된다. 제안된 자기 조직화 신경망의 학습 알고리즘은 다음과 같이 정리될 수 있다.

Step 0: n-각형을 근사화하는 다수의 직선 또는 현들로부터 유추된 중심매개 인수들을 자기 조직화 신경망에 입력하기 위하여 $\rho-\theta$ 극좌표 형태인 특징벡터 $x(\rho, \theta)$들을 만든다.

Step 1: 출력누적망에 있는 MN개의 일반 뉴런중 k개를 임의로 선택하여 누적뉴런 n_c라 둔다.

Step 2: 학습할 특징 벡터 x_i를 임의로 선택한다.

Step 3: 입력된 특징 벡터와 출력 누적망의 일반 및 누적 뉴런사이의 맨하탄 거리를 계산하고, 입력 특징 벡터와 가장 가까운 뉴런을 우세 뉴런 n_d를 구한다.

Step 4: 우세 뉴런 n_d, 누적 뉴런 n_c 및 반응영역 $A(n_c)$사이의 관계에 의해 다음과 같이 학습한다.

$$\begin{aligned}\mu_{ij}(n+1) &= \mu_{ij}(n)+\chi(D_j)\{x_i-\mu_{ij}(n)\} && n_j=n_d \text{ and } n_j\in A(n_c)\\ &= \mu_{ij}(n)+\alpha(D_j)\chi(D_j)\{x_i-\mu_{ij}(n)\} && n_j=n_d \text{ and } n_j\notin A(n_c)\\ &= \mu_{ij}(n) && \text{otherwise}\end{aligned}$$

Step 5: 학습과정에서 인접한 누적 뉴런들의 반응영역이 겹치면 식 (11), 식(12) 및 식(13)에 따라 두 누적 뉴런을 병합하고, 그렇지 않으면 누적뉴런 n_c의 활성 회수 γ를 1 증가시킨다.

Step 6: 신경망이 수렴할 때까지 단계 2에서 단계 5를 반복한다.

Ⅲ. 실험고찰

영상잡음이 존재하는 환경에서 추정된 중심위치의 강건성을 평가하기 위하여 Bresenham알고리즘[12]을 이용하여 300×300크기의 영역내 임의의 위치에 대하여 반지름의 크기를 30-50 화소범위의 원 20개를 발생시켰다. 이렇게 발생시킨 원들을 분산이 2인 가우시안 잡음을 첨가하여 무게 중심법(centroid)[13], Hough 방법[13], Landau 방법[13], 기존의 확률적인 신경망을 이용한 방법[8] 및 제안된 방법(10,000회 반복 수행 후)으로 실험한 결과를 평균자승오차(mean square error: MSE)측면에서 표 1에 나타내었다.

표 1. 알고리즘별 평균 중심위치 오차 및 평균 반지름 오차(단위: pixels)

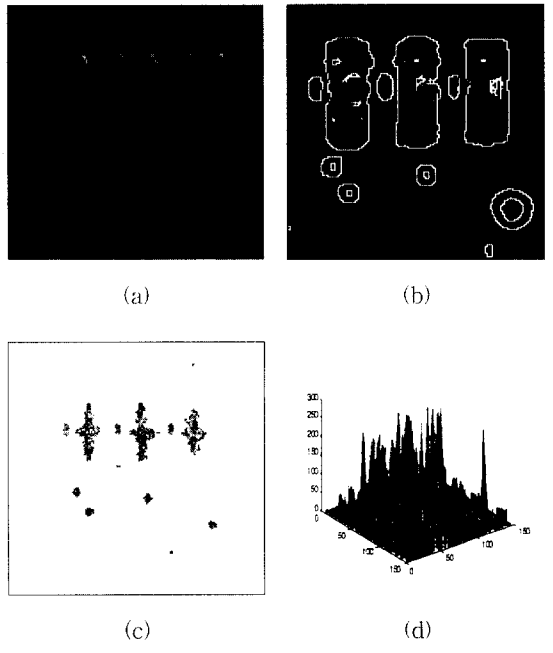
Table 1. Estimation errors of center position and radius by each algorithm. (Unit: pixels)

	Centroid	Hough	Landau	Neural Networks	Proposed
Δ_c	2.07	6.73	2.99	5.28	1.39
Δ_r	3.11	6.12	3.26	5.15	1.89

표에서 Hough변환을 이용한 방법의 경우에는 발생

된 모든 원의 반지름별로 원의 중심을 추정한 결과이다. 무작위로 발생시켜 잡음을 첨가한 원들에 대한 실험 결과에서 알 수 있는 바와 같이 제안한 방법으로 인수를 추정한 결과가 실제 중심위치에 대하여 MSE 측면에서 1.39 화소, 반지름의 크기는 1.89 화소정도의 오차 분포를 나타내어 기존의 방법보다 더 우수한 결과를 나타냄을 확인할 수 있었다. 이 때 Hough변환을 사용한 경우에는 한번에 주어진 반지름을 갖는 물체만을 추출할 수 있었으나, 무게중심을 이용한 방법 및 제안된 방법을 이용하여 중심 추정의 경우에는 무작위 발생된 원들을 한번에 쉽게 찾을 수 있었다.

256 명암도와 0.038mm/pixel의 분해능을 갖는 CS-3330카메라를 사용하여 영상을 획득하고, 획득된 영상 내에 존재하는 n-각형 부품의 중심을 추출하였다. 이 때 획득된 256 명암도 영상을 p_tile 방법과 LoG연산자를 이용하여 입력된 영상에서 n-각형 공업용 부품의 테두리를 추출하는 전처리 과정을 거쳤다. 그림 7은 사각형 및 타원 부품과 원형 구멍이 있는 PCB영상의 전처리부터 자기 조직화 신경회로망을 이용하여 부품 및 구멍의 중심을 추정하는 과정을 반복 학습 단계별로 100, 1000, 10000 그리고 최종 수렴한 20000회 반복 수행하였을 때 구하여진 누적 뉴런의 위치적 분포 및 각 누적 뉴런이 갖는 누적값을 도시한 것이다.



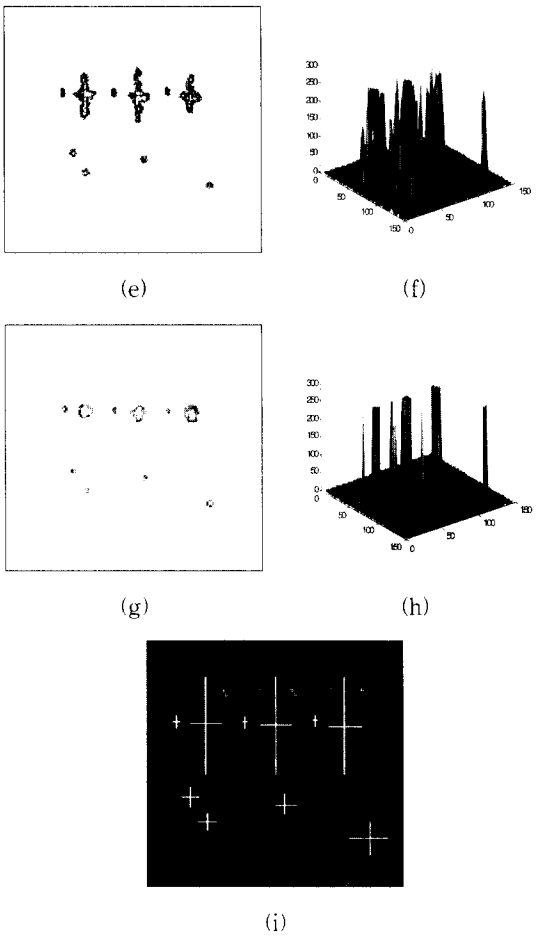
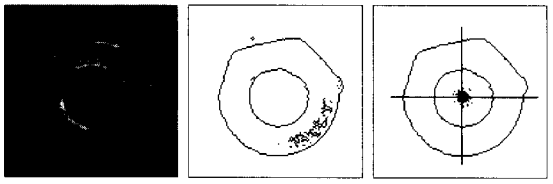


그림 7. PCB영상에 실장된 부품들의 중심 추정: (a) 사각형 부품 3개, 타원 부품 3개, 구멍 4개를 갖는 PCB 영상, (b) 테두리 영상, (c) 신경망을 100회 반복 학습하였을 때 활성화된 뉴런 분포, (d) 100회 반복 학습하였을 때 뉴런들의 누적값 분포, (e) 1000회 반복 학습하였을 때 활성화된 뉴런 분포, (f) 1000회 반복 학습하였을 때 뉴런들의 누적값 분포, (g) 10000회 반복 학습하였을 때 활성화된 뉴런 분포, (h) 10000회 반복 학습하였을 때 뉴런들의 누적값 분포 그리고, (i) 수렴하였을 때 최종적으로 구하여진 각 부품들의 추정된 중심 값들

Fig. 7. Centers estimation procedure of parts mounted on PCB image; (a) PCB image with 3 quadrilateral parts, 3 elliptic parts, and 4 circular holes, (b) boundary image, (c) the distribution of activated neurons when the proposed self-organizing neural networks are learned for 100 iterations, (d) the accumulated value distribution of activated neurons after 100 iterations, (e) the distribution of activated neurons when the networks are learned for 1000 iterations, (f) the accumulated value distribution of

activated neurons after 1000 iterations, (g) the distribution of activated neurons when the networks are learned for 10000 iterations, (h) the accumulated value distribution of activated neurons after 10000 iterations, and (i) the final center estimation values of each parts at which the proposed networks are converged to equilibrium state.

그림 8은 6각형의 외형과 원형의 내부 테두리를 가지고 있는 나사 영상에 대한 전처리 결과 및 제안된 방법으로 15,000회 반복 수행하여 신경망이 수렴 된 후에 추출된 중심을 도시하였다.



(a) (b) (c)

그림 8. 나사의 중심 추정 (a) 입력 영상, (b) 테두리 영상, (c) 제안된 신경망으로 추정한 중심

Fig. 8. Center estimation of a nut; (a) input image, (b) envelope image, and (c) the estimated center using the proposed neural network.

그림에서 보는 바와 같이 6각형 외부 테두리와 내부의 원형 테두리를 가지는 나사 영상에 대한 실험에서 알 수 있듯이 제안된 방법은 각 테두리에 대하여 각각의 중심위치를 추정하지 않고 나사를 하나의 물체로 인식하여 하나의 중심위치를 추출하였다. 그러나 동일한 영상을 일반적인 Hough변환에 적용하여 본 결과 원형인 물체와 육각형인 물체가 동일한 위치에 존재하는 것으로 추정되었다.

표 2. 알고리즘별 평균 중심위치 오차 및 평균 반지름 오차(단위: pixels)

Table 2. The estimation error of center position and radius by each algorithms(Unit: pixels)

Image		Centroid	Hough	Landau	NN	Proposed
PCB image	Δ_o	10.14	8.53	5.98	4.10	2.78
	Δ_r	13.22	8.01	6.52	5.76	3.78
Nut image	Δ_o	6	7	4	4	2
	Δ_r	8	7	5	5	3

그림 7과 8에 도시된 영상을 Peckinpaugh이 사용한 방법[13]으로 기준이 되는 중심을 설정하고, 무게중심법, Hough 방법, Landau 방법 및 제안된 방법으로 실험한 결과를 표 2에 정리하였다.

표에서 PCB영상에 대한 오차는 입력 PCB영상에 존재하는 10개 부품에 대한 MSE이며, 나사 영상은 제안한 방법으로 추정한 절대 오차이다.

Ⅳ. 결 론

본 논문에서는 생성 및 병합 학습 기능을 갖는 자기 조직화 신경망을 이용하여 원시 데이터에 대한 분포를 모르거나, 원시 데이타가 잡음에 의해 오염되더라도 양호하게 n-각형 부품의 중심을 추정할 수 있는 방법을 제안하였다. 여러 개의 원들을 인위적으로 발생시켜 가우스 잡음을 첨가한 영상에 대한 실험에서 제안된 방법은 MSE측면에서 중심의 경우 1.39, 반지름의 경우에는 1.81의 추정오차를 나타내어 잡음에 둔감하게 중심 추정이 이루어졌다. 또한 PCB영상 및 나사영상에 대하여 실험하여 본 결과 PCB영상에 대하여 추정된 결과가 무게중심방법 또는 Hough방법과 달리 여러 물체에 대해서도 잡음 등에 둔감하게 중심을 추정할 수 있었다. 실험결과로 볼 때 제안된 방법은 인수 벡터의 분포를 모르거나 입력된 원시 데이터가 잡음에 오염되어 이용 가능한 정보가 적을 경우에도 다각형 부품 또는 물체들의 중심을 쉽게 추출할 것이라 예상할 수 있다. 제안된 자기 조직화 신경회로망은 문자 인식, 얼굴인식 등 패턴 인식 분야에서 배경과 물체의 분리 등에 이용할 경우 보다 좋은 인식 성능을 나타낼 것으로 생각된다.

참 고 문 헌

[1] M. Moganti, and F. Ercal, "Automatic PCB inspection algorithms: A survey," *Computer Vision and Image Understanding*, vol. 63, no. 2, pp. 287-313, Nov. 1995.

[2] V. F. Leavers, "Survey: Which Hough transform?," CVGIP: Image Understanding, vol. 58, no.2 pp. 250-264, Sept. 1993.

[3] V. F. Leavers and J. F. Boyce, "The Radon transform and its application to

shape parameterization in machine vision," *Image and Vision Computing*, vol. 5, no. 2, pp. 161-166, 1987.

[4] M. K. Ibrahim, E. C. Ngau, and M. F. Daemi, "Weighted Hough transform," *Journal of Society of Photo-Optical Instrumentation Engineer*, vol. 1607, pp. 237-241, 1992.

[5] L. Wen-Nung and C. Yung-Chang, "Robust line-drawing extraction for polyhedra using weighted polarized Hough transform," *Pattern Recognition*, vol. 23, no. 3 pp. 261-274, 1990.

[6] A. M. Wallace, "Matching segmented scenes to models using pairwise relationship between features," *Image and Vision Computing*, vol. 5, no. 2, pp. 114-120, 1987.

[7] N. Kiryati, Y. Eldar, and A. M. Bruckstein, "A probabilistic Hough transform," *Pattern Recognition*, vol. 24, no. 4, pp. 303-316, 1991.

[8] D. W. McMichael, "Decision-theoretic approach to visual inspection using neural networks," *IEE Proceeding Vision and Image Signal Processing*, vol. 141, no. 4, pp. 223-229, 1994.

[9] J Kangas, T. Kohonen, and J. Laaksonen, "Variants of self-organizing maps," IEEE Transactions on Neural Networks, vol. 1, no. 1, pp. 93-99, 1990.

[10] C. Ledoux and J. F. Grandin, "Two original weight pruning methods based on statistical tests and rounding techniques," *IEE Proceeding Vision and Image Signal Processing*, vol. 141, no. 4, pp. 230-237, 1994.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan , New York, 1994.

[12] P. Burger and D. Gillies, *Introduction Computer Graphics: functional, procedural and device-level methods*, pp. 75-144, Addison Wesley, England, 1989.

[13] S. H. Peckinpaugh and R. J. Holyer, "Circle detection for extracting eddy size and position from satellite imaginary of the Ocean," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 2, pp. 267-273, 1994.

저 자 소 개

成 孝 慶(正會員) 第 34卷 1號 C編 參照
현재 경북대 전자공학과 박사과정재학 중. 관심분야는 신경회로망 및 그 응용

崔 興 文(正會員) 第 34卷 1號 C編 參照
현재 경북대 전자전기공학부 교수. 관심분야는 병렬처리, 신경회로망 및 그 응용

論文97-34C-12-1

상태천이확률을 이용한 비동기회로의 저전력 상태할당 알고리즘

(A Low Power State Assignment Algorithm for Asynchronous Circuits Using a State Transition Probability)

具 京 會 *, 趙 慶 錄 *

(Kyoung-Hoi Koo and Kyoung-Rok Cho)

요 약

본 논문에서는 비동기회로 설계를 위한 새로운 상태할당 알고리즘을 제안한다. 제안된 알고리즘은 동적 전력소비를 줄이기 위해 switching activity를 줄이는데 중점을 두고 상태 천이시 상태변수의 변화를 최소로 하는 상태코드를 찾는 것으로 이때 정확한 switching activity를 구하기 위해 상태천이확률을 정의하고 이를 이용해서 상태할당을 진행한다. 본 알고리즘은 one-hot코드를 이용하여 초기상태할당을 수행하고 상태천이확률을 근거로 비트 동일화 및 비트 수정 단계를 거쳐 switching activity를 최소로 하도록 상태코드를 결정한다. 벤치마크회로 실험결과 기존의 방식에 비해 약 60%의 switching activity의 감소를 보여 저전력을 기대할 수 있다.

Abstract

In this paper, a new method of state code assignment for reduction of switching activities of state transition in asynchronous circuits is proposed. The algorithm is based on a on-hot code and modifies it to reduce switching activities. To estimate switching activities as a cost functions we introduce state transition probability(STP). As a results, the proposed algorithm has an advantage of 60% over with the conventional code assignment in terms of switching activity and code length of state assignment.

I. 서 론

집적회로 설계는 소자의 미세화에 따라 대규모 집적회로에 고속 클럭을 분배하는 문제와 저 소비전력 회로의 설계에 있으며 특히 소자의 소형화에 따른 배선지연이 소자의 지연보다 크게되어 칩 설계 시에 클럭 주파수를 결정하기 위해 정확한 배선지연 산출이 필요하나 이는 어려운 일이다. 이 문제점을 해결할 수 있는 회로 설계의 기술로 시스템 클럭 신호를 사용하지 않는 비동기 방식의 설계 방법이 있다. 비동기회로는 시스템 클럭을 사용하지 않기 때문에 대규모 집적회로에서 발생하는 클럭 분배 문제와 내부 동기화의 문제를 해결할 수 있을 뿐 아니라 시스템 클럭에 의한 불필요한 전력소비를 줄일 수 있는 설계로 서브미크론 테크놀러지하에서 속도, 소비 전력측면에서 기존의 동기회로 보다 유리하다는 연구가보고 되었다[1].

일반적으로 동기회로는 상태 할당시 log_2N의 최소 코드 상태할당이 가능하며 racing등으로 인한 논리 hazard문제는 발생하지 않는다. 그러나 비동기회로는 클럭이 없는 관계로 회로 동작시 race-free를 보장하는 상태할당이 요구된다[2]. 이에 고전적으로 최소 코

* 正會員, 忠北大學校 情報通信工學科

(Dept. of Computer and Communication Engineering. Chung-Buk National University)

接受日字:1997年9月3日, 수정완료일:1997年11月20日

드 길이로 상태할당을 하는 Tracey 알고리즘[3]에서 최근에는 전력소비측면에서 코드길이 보다 single transition 상태할당을 중요시하고 있다[4]. 본 논문에서 제안한 상태할당 알고리즘은 동적 전력소비를 줄이기 위해 one-hot코드를 이용한 초기상태 할당에서 비트 동일화 및 수정단계를 거쳐 single transition으로 switching activity를 최소로 하는 상태할당을 수행한다. 본 알고리즘은 switching activity를 구하기 위해 모든 상태에서 발생하는 상태천이를 확률 값으로 계산한 뒤 이 값을 근거로 상태할당을 하므로 기존의 방식에 비해 동적 전력 소비의 감소를 기대할 수 있다. 본 논문은 구성은 다음과 같다. 2장에서는 switching activity에 관련된 소비전력과 일반적인 비동기 회로의 상태할당 방법을 소개하고, 3장에서는 제안한 상태할당 알고리즘을 소개하고 4장에서는 벤치마크회로에 적용하여 기존의 상태할당 알고리즘과 비교한 뒤 5장에서 결론을 맺는다.

Ⅱ. 비동기회로의 상태할당

본 장에서는 일반적인 전력소비 모델을 제시하고 switching activity와 소비전력과의 관계를 살펴보고 기존의 비동기회로 상태할당 알고리즘에 대하여 기술한다.

1. 전력소비 모델

일반적으로 CMOS회로에서의 전력소비는 게이트의 동작확률에 비례하며[5], 이때 소비전력은 다음과 같이 표현할 수 있다.

$$\overline{P_{tot}} = kC\ V_{dd}^2 \alpha f = \frac{1}{2}\ V_{dd}^2 / T_{cycle} \sum_{i=1}^{n_g} C_i P_i \qquad (1)$$

여기서 V_{dd}는 공급전압을 나타내고 α는 switching activity를 의미하며, C_i는 회로의 부하 캐패시터이다. n_g는 회로의 게이트의 수이고 P_i는 스위칭 확률이다. 식(1)의 파라메터중 V_{dd}나 $1/T_{cycle}$은 설계 과정에서 결정되는 변수이므로 상수로 취급된다. 따라서 전력소비를 줄이기 위해서는 $\sum_{i=1}^{n_g} C_i P_i$의 항을 감소시키는 방향으로 연구가 진행되어야 함을 알 수 있다. 식(1)에서 알 수 있듯이 switching activity를 줄이는 것은 그 만큼의 전력소비의 감소를 의미한다. 본 논문에서는 상태천이 시 발생하는 switching activity를 줄여 동적 소비전력을 감소시키는 상태할당 알고리즘을 제안한다.

2. 기존의 상태할당 알고리즘

비동기회로의 상태할당에 관한 기존의 연구[3, 6]는 단순히 상태코드의 길이를 최소로 하는데 중점을 두고 상태할당을 수행하였다. 이 방식은 할당된 코드의 길이를 줄일 수 있으나 switching activity가 커지는 단점이 있다. 또 선형코드를 이용하여 간단히 비동기 유한 상태기를 기술할 수 있는 방법도 제시되었다[7]. 그러나 이 방식은 상태코드의 길이가 상태 수에 비례하여 커지는 단점을 갖고 있다. 이들 상태할당 알고리즘은 모두 switching activity를 고려하지 않고 상태할당을 수행하는 것으로 저전력회로 설계에는 부적합한 알고리즘이다. 표1은 5개의 상태를 갖는 회로에 대한 one-hot코드 및 Tracey 알고리즘의 상태할당을 나타낸다.

표 1. one-hot code 및 Tracey 상태할당
Table 1. one-hot code and Tracey state assignment.

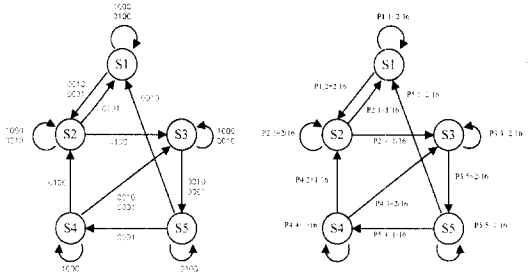
입력 / 현재상태	I_1	I_2	I_3	I_4	상태할당	
	다음상태				one-hot	Tracey
S_1	S_1	S_4	S_2	S_2	10000	001
S_2	S_2	S_3	S_2	S_1	01000	000
S_3	S_3	S_5	S_3	S_5	00100	011
S_4	S_4	S_2	S_3	S_3	00010	100
S_5	S_5	S_5	S_1	S_4	00001	010

Ⅲ. 제안한 비동기회로의 상태할당 알고리즘

본 장에서는 비동기 유한 상태기로 기술된 비동기회로의 내부 상태천이 관계에서 switching activity를 구하기 위해 상태천이확률(STP)를 정의하고 이를 이용해서 switching activity구하는 방법과 본 논문에서 제안한 상태할당 알고리즘의 전개과정을 설명한다.

1. 상태천이 확률(STP)의 계산

비동기회로의 상태할당을 위해서는 먼저 비동기 유한상태기로부터 각 상태사이에서 발생하는 천이관계를 확률로 나타낸다. 비동기 유한상태기는 회로의 입력조건과 입력에 따른 상태천이를 표현한 것으로 이를 기본으로 하여 각 신호 천이의 관계를 확률적으로 표현할 수 있다. 그림 1(a)에 표1의 비동기 유한상태기를 바탕으로 작성한 상태천이도가 나타나 있다.



(a) (b)

그림 1. 상태천이도 : (a) 입력조건이 표시된 상태천이도; (b) 상태천이확률이 표시된 상태천이도

Fig. 1. A state transition diagram(STD) : (a) STD with input condition; (b) STD with a state transition probability.(STP)

상태천이도가 완성되면 이를 바탕으로 해서 각 상태사이의 천이를 구한다. 천이조건 확률은 다음의 식으로 정의된다[8].

$$p_{i,j} = P(S_j \mid S_i) \tag{2}$$

천이조건 확률은 현재상태 S_i에서 다음상태 S_j로 상태천이가 일어날 확률을 나타낸다. 상태천이는 입력이 변화할 때 일어나므로 천이 조건 확률은 입력신호의 변화 확률에 의하여 얻어지며 $p_{i,j} = P(I)$로 표기하고 입력값 I에 따른 상태천이의 확률을 의미한다. 예로 그림1에서 S_2에서 S_1의로 천이 $P_{1,2}$는 4비트의 입력값 즉, 16가지의 경우의수 중에서 입력값이 0010일 때와 0001일 경우에만 천이가 일어나므로 S_2에서 S_1의로 천이조건확률 $p_{1,2}$는 $P(0010)+P(0001)= 2/16$ 의식으로 구할 수 있다. 모든 상태 사이에서 발생하는 천이조건확률은 그림1 (b)에 나타나 있다.

천이조건 확률은 입력값의 변화만을 고려하여 구한 값이기 때문에 이것만으로는 회로의 switching 정도를 정확하게 나타낼 수 없다. 따라서 보다 정확한 switching activity를 얻기 위해서는 회로의 현재상태를 고려하여 이에 따른 천이조건 확률을 계산하여야 한다. 즉 입력변화에 따른 상태천이 뿐만 아니라 회로 내부의 천이를 동시에 고려하여 총 상태천이확률을 계산한다. 총 상태천이확률 $Prob_{i,j}$을 식(3)같이 정의한다.

$$Prob_{i,j} = (p_{i,j}) \times (P_i) \tag{3}$$

여기서 P_i는 회로가 현재 상태 i에 있을 상태확률을 나타낸다. 식(3)에서 알 수 있듯이 천이조건 확률만 가지고 총 천이확률을 구하는 것은 정확한 값을 예측하는 것은 어렵다. 또한 총 상태천이확률이 큰 값을 갖기 위해서는 천이조건확률 $p_{i,j}$와 회로가 현재 상태 i에 있을 상태확률 P_i가 높아야 한다는 것을 알 수 있다. 상태확률을 구하기 위해 먼저 천이조건확률 $p_{i,j}$을 다음의 행렬식A로 나타낸다[9].

$$A = \begin{bmatrix} \frac{2}{16} & \frac{2}{16} & 0 & 0 & 0 \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} & 0 & 0 \\ 0 & 0 & \frac{2}{16} & 0 & \frac{2}{16} \\ 0 & \frac{1}{16} & \frac{2}{16} & \frac{1}{16} & 0 \\ \frac{1}{16} & 0 & 0 & \frac{1}{16} & \frac{1}{16} \end{bmatrix} \tag{4}$$

행렬식 A에서 각 열의 값은 천이의 현재상태를 표시하고 행의 값은 다음상태를 나타낸다. 즉, 1행1열의 값 2/16은 S_1에서 S_2로의 천이조건확률을 나타낸다. P_i의 값은 천이조건 확률에서 얻어진 행렬식을 이용하여 행렬방정식 $P = A^T P$ 을 계산하여 구할 수 있다.

$$\begin{aligned} P_1 &= \frac{2}{16}P_1 + \frac{1}{16}P_2 + \frac{1}{16}P_5 \\ P_2 &= \frac{2}{16}P_1 + \frac{2}{16}P_2 + \frac{1}{16}P_4 \\ P_3 &= \frac{2}{16}P_3 + \frac{2}{16}P_4 + \frac{1}{16}P_2 \\ P_4 &= \frac{1}{16}P_4 + \frac{1}{16}P_5 \\ P_5 &= \frac{2}{16}P_3 + \frac{1}{16}P_5 \end{aligned} \tag{5}$$

식(5)는 일반연립 방정식이므로 간단히 계산할 수 있으며 해는 각각 $P_1=0.059$, $P_2=0.012$, $P_3=0.12$, $P_4=0.054$, $P_5=0.764$이다. 효율적인 계산을 위해 상태확률은 소수점 이하 둘째 자리에서 반올림하였다. 식(5)에서 구한 P_i를 식(3)에 대입하여 총 천이확률을 계산하는 과정이 그림 2에 나타나있다. 상태할당을 시작하기 위해서는 구해진 총 천이확률을 정수화 하여 S_i와 S_j의 weight값 $W_{i,j}$를 식(6)에 따라 계산한다.[10]

$$W_{i,j} = (p_{i,j}) \times (P_i) + (p_{j,i}) \times (P_j) \tag{6}$$

이 과정에서 자기 자신으로의 상태천이는 switching activity계산에서 제외하였으며 에지(edge)에 표시된 weight값이 switching 정도를 나타낸다. 그림3에 완성된 weight 값 그래프가 나타나있다.

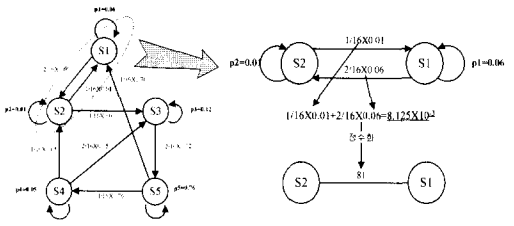


그림 2. Weight값의 계산
Fig. 2. Calculation of weight value for the Fig. 1.

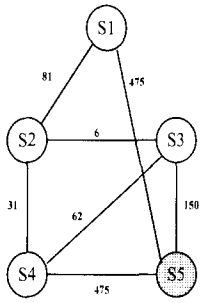


그림 3. Weight값이 표시된 그래프
Fig. 3. Weight value from Eq. (6)

2. 제안한 알고리즘의 진행과정

상태할당의 진행과정은 비동기 유한상태기로 기술된 회로를 상태천이도(STD)로 표현한 뒤 상태천이확률을 계산하고 이를 이용하여 weighted 그래프를 만든다. 만들어진 weight 그래프는 회로의 switching정도를 나타내는 그래프이므로 이 그래프가 상태할당을 수행하는데 기초가 되는 그래프이다. 본 논문에서는 초기 상태할당은 간단히 구할 수 있는 one-hot 코드를 이용하고 초기 상태할당이 결정되면, weight값을 기준으로 할당된 초기 상태코드를 수정해 나간다. 제안한 상태할당 알고리즘을 그림 4에 나타내었다. 그림4의 알고리즘에서 먼저 기준상태를 결정하며 이는 상태천이도에서 weight값이 가장 높은 상태이다. 표1회로의 weighted 그래프는 그림3으로 상태할당은 S_5에서 시작되는 에지의 weight값이 가장 높기 때문에 S_5가 상태할당의 기준이 된다. S_5를 시작점으로 weight값의 크기 순서인 $S_5 \rightarrow S_4 \rightarrow S_1 \rightarrow S_3 \rightarrow S_2$로 초기 상태할당을 수행한다. 여기서 노드 S_i과 S_j사이에서 일어나는 상태천이를 식(7)의 switching activity(SA)로 정의한다.

$$SA_{i \leftrightarrow j} = W_{i,j} \times d_{Hi,j} \qquad (7)$$

식(7)의 $W_{i,j}$는 S_i와 S_j사이의 weight값으로 그림3에 나타난 값이고 d_H는 각 상태에 할당된 코드 사이의 해밍거리를 나타낸다. 식(7)은 노드 사이에서 발생하는 switching activity가 입력조건에 의한 상태천이 뿐만 아니라 할당된 상태코드를 동시에 고려해야함을 알 수 있다.

```
/* switching activity 계산 */
calculate (pi,j × Pi)
make weight graph

/* 기준상태 결정 */
Select_base_state( )
  for each state;
    sum all weight;
Sbase <= highest weight;

/* 초기상태할당 */
sort edge weight decreasing order;
  for each state;
    S1st <= assign one-hot code;

/* 비트동일화 과정*/
S2nd = Sbase XOR S1st;

/* 비트수정과정 */
In 2 transition group;
( Si, Sj ) <= highest weight edge;
Si <= highest weight sum;
  for redundant codes;
    dH (Sj, S(mod) ) = 1;
In S(mod);
Select_code_for_minimum_SA( );

/* 비트 삭제과정 */
  for each state code;
    if ( Si == '0' or '1');
drop Si;
```

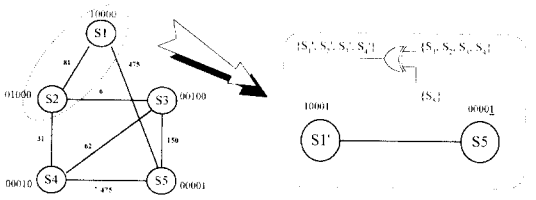
그림 4. 제안한 상태할당 알고리즘의 흐름도
Fig. 4. Flow of proposed state assignment algorithm.

A. 비트 동일화

one-hot 코드를 이용한 초기할당과정이 끝나면 할당된 상태코드는 모두 2비트의 천이를 갖는다. 식(7)에서 변수는 d_H의 값이므로 SA의 값을 줄이기 위해서는 두 상태사이에 할당된 상태코드의 해밍거리를 줄여야한다. 따라서 2비트의 천이를 1비트 천이로 줄이는 것이 상태할당 첫 단계의 목표이다. 즉 weight값에 따라 2비트 천이를 1비트의 천이로 바꾸며 이 과정을 비트 동일화(bit equalization)라 정의한다. 그림 5에 one-hot 코드를 이용한 초기상태할당과 비트 동일화 과정이 나타나있다.

초기 상태할당이 끝나면, 그림 5. (a)에서 알 수 있듯이 weight값이 가장 큰 노드 S_i를 기준으로 비트 동일화 과정을 진행한다. 이 과정은 weight값이 높은 노드를 1비트의 천이로 만드는 과정으로 S_i와 천이가 존재하는 상태들의 코드값은 S_i와 $EXOR$를 해서 얻

을 수 있다. 즉 S_i에서 천이되는 상태의 집합을 S_T라 하면 S'_T은 비트 동일화 후에 얻어지는 상태로 다음 식으로 구할 수 있다.



(a) (b)

그림 5. 1단계 상태할당 과정 : (a)초기상태할당; (b) 비트 동일화 과정;

Fig. 5. Procedure of 1st stage state assignment : (a) initial state assignment; (b) bit equalization;

$$S'_T = S_i \oplus S_T \tag{8}$$

그림 5(a) 예제에서 weight값이 가장 큰 S_5를 중심으로 비트 동일화 과정이 진행되며 이 과정은 그림 5(b)에 나타나있다.

B. 비트 수정

비트 동일화과정을 거치면 weight값이 가장 큰 S_i에서의 모든 상태천이는 2비트 변화에서 1비트 변화로 줄어들어 weight 그래프는 2비트 천이그룹과 1비트 천이 그룹으로 나누어진다. 비트 수정 단계는 2비트 천이 그룹에서 발생하는 switching activity를 줄이는 단계로 다음에 기술된 식(9)을 만족하면 비트 수정 단계를 진행한다.

$$\delta(S_i, I_k) \cap \delta(S_j, I_k) = \varnothing \tag{9}$$

여기서 $\delta(S, I) : S \times I \rightarrow S$이고, 이때 I는 입력, S는 상태이고 δ는 상태천이함수이다. 따라서 식(9)은 하나이상의 상태천이 과정이 동일 입력변수(I_k)에 대하여 동시에 상태를 천이하는 조건이 없을 때 비트 수정단계를 진행한다. 비트 수정은 2비트 변화 그룹에서 각 에지의 weight합이 가장 큰 상태 S_i를 수정대상 노드로 설정하고 수정대상 상태와 가장 큰 weight값을 갖는 S_j를 수정 기준상태로 설정한다. 여기서 상태코드를 2^N개의 코드계열 중 이미 할당된 코드그룹 C_1과, 비사용 코드그룹 C_2를 식 (10)로 정의한다.

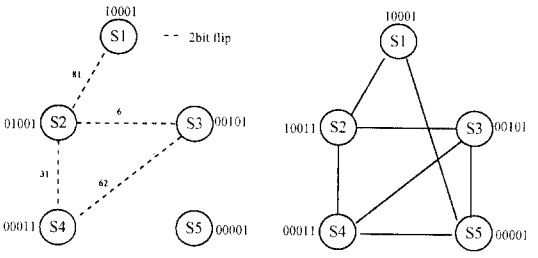
$$C_1 = \bigcup_{k=1}^{n} S_k$$
$$C_2 = \bigcup_{k=n+1}^{N} S_k \tag{10}$$

로 n은 상태수, $N=2^n$이다. 여기서 S_j와 1비트 천이를 갖는 코드 그룹S_{mod}를 C_2그룹에서 찾는다.

$$S_{\mathrm{mod}} = \bigcup_{k=n+1}^{N} \{d_H(S_j,\ C_2{}^k) = 1\} \tag{11}$$

식(11)의 S_{mod} 코드계열 중에서 식(12)로 switching activity가 가장 적은 코드를 S_i에 할당한다.

$$S_i = \min(SA_{k,\mathrm{mod}}),\quad S_j \in S_{\mathrm{mod}} \tag{12}$$



(a) (b)

그림 6. 2단계 상태할당과정 : (a) 비트동일화 과정 후; (b) 비트수정 과정

Fig. 6. Procedure of 2nd stage state assignment : (a) 2bit flip states after bit equalization; (b) bit modification process toward to 1bit change.

그림 6(a)에 나타나는 2비트 천이 그룹 중에서 S_1과 S_2사이의 노드가 가장 큰 weight값을 가지므로 이 들 사이에서 일어나는 천이를 줄이는 것이 수정 단계이다. 즉, 비트 수정단계는 비트 동일화 과정이 끝난 후에 남아있는 에지중에서 weight가 가장 높은 에지의 천이를 줄이는 것이다. 그림 6(a)에서는 S_1과 S_2에 할당된 코드 중 S_2의 상태코드를 수정한다. 여기서 S_1의 코드를 수정하여 S_2와의 천이를 1비트로 만들 경우 weight가 큰 S_1과 S_5사이 에지의 천이가 1비트로 유지되지 않기 때문에 switching activity의 감소를 기대하기 어렵기 때문이다. 비트 수정과정은 코드 S_i(10001)과의 해밍 거리가 '1'이 되는 코드중 이미 할당된 코드를 제외한 S_{mod} = {10000, 11001, 10101, 10011}을 찾는다. 이들 코드 중에서 식(12)을 만족하는 10011을 S_2에 할당한다. 그림 6(b)에 비트수정 과정을 마친 후의 상태할당 코드가 나타나있다. 또 S_k={b_0^k, b_1^k, b_2^k, ···, b_l^k}으로 비트를 표시한다면 one-hot코드를 사용하기 때문에 식(13)로 되는 b_i를 발견할 수가 있다.

$$\sum_{i=0}^{l} b_i^k = \{0,1\} \quad (13)$$

이는 모든 상태에서 비트값이 동일하므로 삭제하여도 상태코드 천이에는 racing이 발생하지 않는다. 비트 수정단계가 끝나면 할당된 상태코드에서 식(13)에 따른 불필요한 비트를 삭제한다. 본 예제에서는 비트 동일화과정에서 사용된 최하위(LSB)비트를 제외하고 상위 두 번째 비트가 모두 '0'이므로 이 두 비트를 삭제하고 최종적으로 4비트로 상태할당을 마쳤다. 그림 7에는 최종적인 상태할당 결과이며 초기상태 할당 과정에서 5비트로 할당된 상태코드 중에서 불필요한 비트를 제거하고 최종적으로 4비트의 상태코드로 회로를 기술할 수 있음을 알 수 있다.

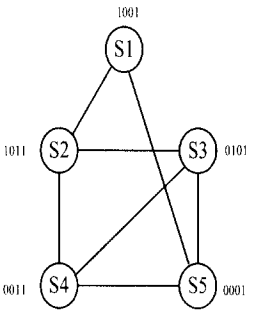


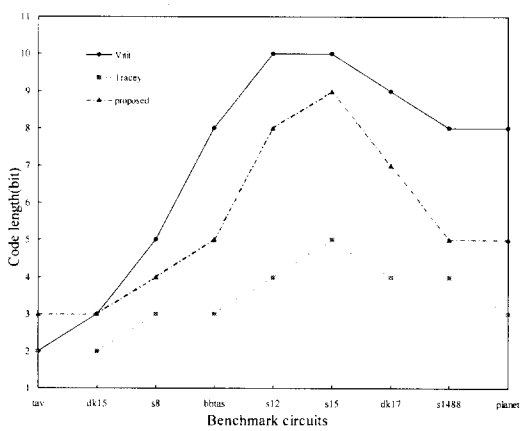
그림 7. 최종적으로 할당된 상태코드
Fig. 7. Optimized state code assignment.

Ⅳ. 모의실험 결과 및 분석

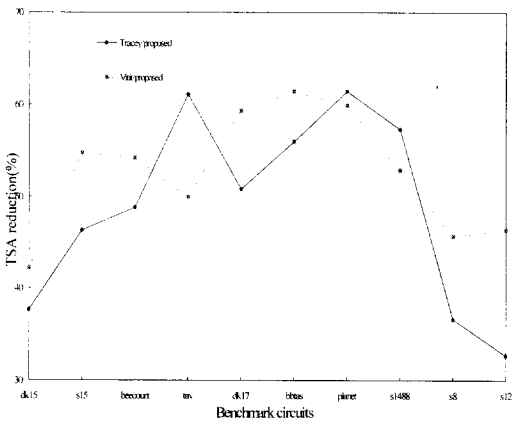
본 장에서는 제안한 상태할당 알고리즘을 기존의 알고리즘[5, 11]과 비교·분석하기 위해 프로그래밍 한 뒤 벤치마크회로 LGSynth89[12]에 적용한 결과를 기술하였다. 기존 알고리즘과의 비교는 switching activity와 상태코드 길이의 두 가지 측면에서 비교한다. 식(3)의 상태천이 확률을 적용하여 총 switching activity를 계산하기 위해서 식(14)을 정의한 뒤 기존의 알고리즘과 비교한다. 총 switching activity (TSA)는 다음과 같이 정의한다.

$$TSA = \sum_{i=1}^{n_{edge}} \{W_i \times d_{Hi}\} \quad (14)$$

n_{edge}는 상태사이에 존재하는 에지(edge)의 수를 나타낸다. 이 식은 모든 에지에서 발생하는 switching activity를 더한 값으로 회로에서 일어나는 모든 switching정도를 나타내는 값이다. 표2에는 벤치마크 회로의 특성이 나타나있고 기존의 알고리즘과의 비교 결과를 그림 8에 나타내었다. Tracey방식[5]에 비해 본 논문에서 제안한 알고리즘이 할당된 코드길이는 늘어난 반면 총 switching activity는 최대 60%가량 낮아졌으며, Vitit방식[11]과 비교해 보면 할당된 코드길이 뿐 아니라 switching activity도 현저히 줄일 수 있음이 확인되었다.



(a)



(b)

그림 8. 기존의 알고리즘과의 비교도 : (a) 할당된 코드길이; (b) TSA감소비율
Fig. 8. Comparison to conventional algorithm : (a) assigned state code length (b) reduction ratio of TSA

Ⅴ. 결 론

본 논문에서는 저전력 비동기회로 설계를 위해 상태천이확률을 이용하여 switching activity를 계산한 뒤

이를 바탕으로 상태할당을 수행하는 새로운 비동기회로의 상태할당 알고리즘을 제안하였다.

표 2. 기존의 상태할당 알고리즘과의 비교
Table 2. Comparison with conventional and proposed algorithm.

Exam.	# of States	# of State Transition	Code length Tracey/Vitit[11]/ 제안된 방식	TSA Tracey/Vitit[11]/ 제안된 방식	TSA 감소비율(%) [Tracey/proposed] / [Vitit/proposed]
tav	4	49	2/2/3	2618/2034/1017	61.1/50.0
dk15	4	32	2/3/3	11460/12360/7140	37.7/42.2
s8	5	20	3/5/4	5370/6280/3410	36.6/45.7
bbtas	6	24	3/8/5	184/210/81	56.0/61.4
planet	6	58	3/8/5	1480/1426/ 571	61.4/59.9
s1488	6	72	4/8/5	21810/19771/9306	57.3/52.9
beecount	7	28	4/7/6	5790/6470/2960	48.8/54.2
dk17	8	36	4/9/7	2638/3190/1296	50.8/59.3
s12	12	30	4/10/8	9420/11830/6340	32.7/46.4
s15	15	42	5/10/9	1763/2090/947	46.3/54.7

제안한 알고리즘은 회로에서 발생하는 총 switching activity를 계산하여 기존의 알고리즘과 비교하였고 최종적으로 할당된 상태코드의 길이도 비교하였다. 벤치마크회로 실험 결과 제안한 알고리즘으로 상태할당을 수행한 결과 동적 전력소비에 관련된 총 switching activity는 기존의 방식에 비해 최대 약60%의 감소를 확인할 수 있었다. 따라서 제안한 상태할당 알고리즘을 이용하여 상태할당을 수행할 경우 기존의 방식에 비해 동적 전력 소비를 줄일 수 있는 회로의 구현을 기대할 수 있다.

참 고 문 헌

[1] M. Afhahi and C. Svensson, "Performance of synchronous and asynchronous schemes for VLSI systems," IEEE Trans., Compt., vol 41, no. 7, pp. 858-872, July 1992

[2] L. Lavagno, *Algorithms for Synthesis and Testing of Asynchronous Circuits*, 1996, KAP.

[3] J. H. Tracey, "Internal state assignment for asynchronous sequential machine," *IEEE Trans. EC*, vol. 15 pp. 551-560, Aug. 1966.

[4] J. W. Rutten and M. R. Berkelaar, "Improved state assignment for burst mode finite state machines," *Proc. ASYNC97*, pp. 228-239, April 1997

[5] J. Rabaey, *Low Power Design Methodologies*, KAP, USA, 1996.

[6] R. J. Smith, "Generation of internal state assignment for large asynchronous sequential machines," *IEEE Trans. Compt.*, vol 23, pp. 924-932, Sep. 1974.

[7] K. R Cho and Kunihiro Asada, "VLSI oriented design method of asynchronous sequential circuits based on one-hot state code and two-tr AND Logic," *ISCAS*, 1991.

[8] L. Benini and G. De Micheli, "State assignment for low power dissipation," *IEEE Journal of SSC*, vol. 30, no. 3, Mar. 1995

[9] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Sciences Applications*. Prentice Hall, 1982.

[10] Kyoung-Hoi Koo and Kyoung-Rok Cho, "A new state assignment for asynchronous circuits for switching activity reduction," *Proc. 40th MWSCAS*, Aug. 1997.

[11] Vitit Kantabutra and Andreas G. Andreou, "A state assignment approach to asynchronous CMOS circuit design," *IEEE Trans. Compt.*, vol 43, no 4, pp. 460-469, April, 1994

[12] *Proc. International Workshop on Logic Synthesis*, 1989

저 자 소 개

具 京 會(正會員)

1974년 2월 8일생. 1996년 충북대학교 정보통신공학과 졸업(공학사). 1996년 3월 ~ 현재 충북대학교 정보통신공학과 대학원 재학중. 주관심분야는 통신용 ASIC설계, 비동기회로 설계, 논리회로 합성 등

趙 慶 錄(正會員)

1955년 6월 10일생. 1977년 경북대학교 전자공학과 졸업(공학사). 1979년 7월 ~ 1986년 7월 (주)LG전자 TV연구소 선임연구원. 1989년 동경대학교(일본)대학원 전자공학과(공학박사). 1992년 3월 ~ 1992년 8월 (재)산업과학기술연구소 주임연구원. 1992년 8월 ~ 현재 충북대학교 정보통신공학과 부교수. 주관심분야는 고속 VLSI 회로설계, 통신용 LSI개발, 컴퓨터 구조 및 마이크로 프로세서 설계 등

論文97-34C-12-2

다중 프로세서 칩을 위한 시스템 제어 장치의 구조설계 및 FPGA구현

(Architecture Design and FPGA Implementation of a System Control Unit for a Multiprocessor Chip)

朴 性 模 *, 丁 甲 天 **, 鄭 揚 勳 ***, 尹 碩 漢 ****

(Seong Mo Park, Gab Cheon Jung, Yang Hoon Jung, and Suk Han Yoon)

요 약

본 논문은 대규모 병렬 처리 시스템에서 하나의 노드 프로세서로서 사용될 수 있는 다중 프로세서 칩 내부의 시스템 제어 장치의 설계와 FPGA 구현에 대해서 기술하였다. 다중 프로세서 칩은 정수 연산 장치, 명령어 캐시, 데이터 캐시, 메모리 관리 유니트로 구성된 프로세서 4개와 버스 유니트, 그리고 시스템 제어 장치로서 구성되어졌다. 설계된 시스템 제어 장치의 주요 기능은 전역변수들의 잠금 및 해제, 칩 내의 4개 프로세서 명령어의 동기화, 인터럽트들의 제어, 클럭신호들의 발생과 분배, 시스템 에러와 개별 프로세서 에러의 제어, 프로세서의 상태 제어 및 감시 등이다. 이 시스템 제어 장치는 Verilog HDL을 사용하여 최상위 레벨에서 기술하였고, 간단한 트랩처리기와 외부 인터럽트 제어기가 추가된 환경에서 시뮬레이션을 수행하여 그 기능을 검증하였다. 검증이 완료된 시스템 제어 장치의 기능 블록들은 RTL(Register Transfer Level) 모델로 수정되었으며 Xilinx FPGA 셀 라이브러리를 사용하여 Synopsys 툴에서 합성되었다. 합성된 회로는 타이밍 검증 후 Xilinx FPGA 칩(XC4025EPG299)으로 구현되었다.

Abstract

This paper describes the design and FPGA implementation of a system control unit within a multiprocessor chip which can be used as a node processor in a Massively Parallel Processing(MPP) system. The multiprocessor chip is consisted of four integer units, instruction caches and data caches, memory management units, a bus unit and a system control unit. Major functions of the system control unit are locking/unlocking of the shared variables of protected access, synchronization of instruction execution among four integer units, control of interrupts, generation and distribution of clock signals, control of system error and/or individual processor error, watch and control of processor's status, etc. The system control unit was modeled in very high level using Verilog HDL. Then, it was simulated and verified in an environment where trap handler and external interrupt controller were added. Functional blocks of the system control unit were changed into RTL(Register Transfer Level) model and synthesized using Xilinx FPGA cell library in Synopsys tool. The synthesized system control unit was implemented by Xilinx FPGA chip(XC4025EPG299) after timing verification.

* 正會員, 全南大學校 컴퓨터工學科
(Dept. of Computer Eng., Chonnam National Univ.)
** 正會員, 全南大學校 電子工學科
(Dept. of Electronics Eng., Chonnam National Univ.)
*** 正會員, 現代電子 멀티미디어硏究所
(Hyundai Electronics Multimedia R&D Center)
**** 正會員, 韓國電子通信硏究院
(System Reserch Department, ETRI)
接受日字:1997年6月28日, 수정완료일:1997年11月12日

Ⅰ. 서 론

많은 컴퓨터 회사들이 비교적 값이 싸진 프로세서를 가능한 많이 사용하여 가격 대 성능의 비율이 높은 대규모 병렬처리(MPP: Massively Parallel Processing) 시스템을 개발하고 있다. 대형 컴퓨터의 전형적인 사용 용도를 살펴보면 데이터 저장과 검색 등 데이터베이스관리에 약 50% 정도의 성능을 할애하고, 나머지 50%는 클라이언트 응용 부분에 할애하고 있다고 한다. 이런 점에 비추어 볼 때 데이터베이스관리 전용 MPP 시스템의 대두는 필연적이라고 할 수 있다.

병렬 처리 컴퓨터는 계산량이 방대한 문제들을 효율적으로 처리하기 위해 하나의 방대한 문제를 여러 개의 작은 문제로 나누어 이들을 동시에 여러 개의 마이크로프로세서로 처리함으로써 전체 문제의 처리 속도를 높이는 방법을 사용한다.

대규모 병렬 처리 컴퓨터에서 요구되는 전용 마이크로프로세서는 단일 프로세서로서 성능 좋은 프로세서보다는 각 프로세서의 성능은 다소 낮더라도 여러 개의 프로세서를 사용하였을 때 그 성능이 더욱 향상되는 프로세서를 사용하여야 한다. 성능이 우수한 마이크로프로세서를 대규모 병력 처리 시스템에 사용하더라도 연결 망이나 입출력 장치들이 고성능의 마이크로프로세서를 따라 가지 못한다면 값비싼 마이크로프로세서를 제대로 활용 못할 뿐만 아니라, 대규모 병렬처리 시스템이 갖는 장점을 희석시킬 수 있기 때문이다. 그러므로 대규모 병렬 컴퓨터 내에서 사용될 마이크로프로세서는 효과적인 가격 대 성능 비를 갖춘 전용 프로세서이어야 한다[1].

대단위 병렬 처리 컴퓨터들은 이론적으로 대단히 높은 속도를 가능하게 하나, 이를 실제적으로 실현하는 것은 프로세서들을 어떻게 효과적으로 통신하여 함께 동작시키느냐에 달려 있다. 그러므로 큰 업무를 얼마나 작은 단위로 나눌 수 있는지, 마이크로프로세서는 몇 개나 장착할 수 있고, 프로세서들 사이의 인터페이스를 어떻게 효율적으로 할 것인지는 병렬 처리 컴퓨터의 성능을 좌우하는 관건이 된다고 할 수 있다.

최근 마이크로프로세서의 성능을 높이는 방법으로 명령어 수준의 병렬성(ILP: Instruction Level Parallelism)을 높여 CPI(Clocks Per Instruction)를 낮추는 방식을 많이 사용하고있다. 명령어 수준의 병렬성을 높이는 방법으로는 수퍼스칼라 방식, 수퍼파이프라인방식, VLIW(Very Long Instruction Word) 구조, 다중 프로세서 칩 구조가 있다[2].

수퍼스칼라 구조에서는 동시에 issue하는 명령어들 간의 의존성을 하드웨어적으로 점검하여 명령어의 수행 순서를 능동적으로 할당하여야 하므로, 동시에 수행되는 명령어의 개수가 많을수록 제어 회로가 복잡해지는 단점이 있다. 수퍼파이프라인 방식은 하나의 파이프라인을 여러 부분으로 나누어 놓고 클럭 속도를 빠르게 함으로써 파이프라인의 속도를 빠르게 하는 방식인데, 클럭 속도를 개선하는 데에는 한계가 있다. 또한 VLIW 구조는 다른 버전의 동일 구조를 가진 마이크로프로세서에서 프로그램의 이진 호환성이 유지되지 않는다는 단점을 가지고 있으며, 컴파일러의 기술이 아직 VLIW 구조를 충분히 지원할 수 있을 만큼 발달되어 있지 않다는 문제점이 있다[3].

다중 프로세서 칩 구조는 각각 프로그램 카운터와 레지스터 파일을 내장하는 프로세서 여러 개를 한 칩에 집적시켜 두개 이상의 명령어 스트림(stream)들을 동시에 수행시킴으로써 병렬성을 증대시키고 기능 장치들을 효율적으로 사용할 수 있도록 한 구조이다[4].

다중 프로세서 칩 구조는 운영체제나 응용 프로그램이 다중 쓰레드를 지원하는 환경에서 제어 회로를 단순화시켜 수퍼스칼라 구조보다 기능 유니트를 더 적게 사용하면서도 CPI를 낮출 수 있는 구조이다.

본 논문에서는 2장에서 MPP 전용 마이크로프로세서로서 응용 프로그램의 크기에 따라 확장성을 가지고 사용될 수 있는 다중 프로세서 칩 구조에 대하여 기술하고, 3장에서 칩 내 프로세서들 사이의 동기화와 인터페이스를 담당하도록 설계된 시스템 제어 장치의 구조에 대하여 기술하며, 4장에서 시뮬레이션 및 검증, 5장에서 성능분석, 6장에서 FPGA로 구현과정 및 구현결과에 대하여 기술하고, 7장에서 결론을 맺었다.

Ⅱ. 다중 프로세서 칩 구조

다중 프로세서 칩의 하드웨어 구성 요소는 그림 1과 같이 정수 연산 장치, 명령어 캐시, 데이타 캐시, 메모리 관리 장치로 구성된 프로세서 4개와 버스 제어 장치, 그리고 시스템 제어 장치이다.

제안된 다중 프로세서 칩은 하드웨어 구조를 간단히 하기 위해 각 프로세서 내 정수 연산 장치는 단일 파

이프라인을 갖도록 하였다.

각 프로세서는 일반적으로 한 싸이클에 하나의 명령어를 수행시킬 수 있으며, 칩에 4개의 프로세서를 집적시킴으로써 싸이클 당 4개의 명령어를 수행시킬 수 있다. 따라서, 이 다중 프로세서 칩의 동작 주파수를 100MHz로 할 때 400MIPS의 성능을 기대할 수 있다.

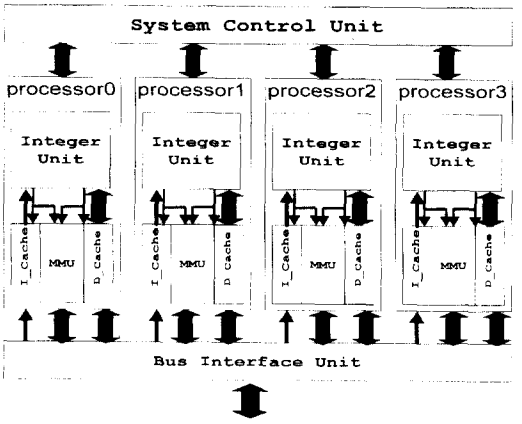


그림 1. 다중 프로세서 칩의 구성도
Fig. 1. Configuration of a multiprocessor chip.

다중 프로세서 칩의 명령어 세트는 32비트 마이크로프로세서의 IEEE 표준으로 채택된 공개구조 SPARC-V8의 정수 연산 장치 관련 명령어 세트[5]에 다중 프로세서 시스템 지원 명령어들을 추가한 구조이다. 추가된 명령어들은 전역 변수 잠금 처리를 위한 sylscc, sylr 명령어와 쓰레드들의 동기적 수행을 위한 sybs, sybrcc 명령어 등이다.

Sylscc, sylr 명령어는 전역변수에 대한 쓰기 충돌 방지를 빠르고 간편하게 칩 내에서 수행하기 위해 추가되었으며, sybs, sybrcc 명령어는 칩내의 4개의 프로세서에서 실행되는 쓰레드의 동기적 수행을 하드웨어적으로 지원함으로써 쓰레드의 동기화를 보다 쉽게 수행하도록 하기 위해 추가되었다. 이 명령어들은 시스템 제어 장치에 있는 동기화 정보를 읽거나 변경시키는 작용을 통해 동기화를 지원하는데, 제어 알고리즘을 간단히 하고, 수행 싸이클을 줄일 수 있으며, 전력소비를 줄일 수 있다.

시스템 제어 장치(SCU : System Control Unit)는 다중 프로세서 칩이 다음과 같은 운영 원칙에 따라 동작하도록 지원하게 설계되었다.

첫째, 공유 메모리를 기반으로 하는 다중 프로세서 칩의 각 프로세서에 대한 초기화는 개별 프로세서 상태 감시 및 제어 기능을 이용하여 수행한다.

둘째, 전역 변수 값을 변경시키는 경우에는 오직 한 개의 프로세서만이 실행할 수 있도록 하며, 쓰레드의 동기적 수행을 보장한다.

셋째, 인터럽트 처리는 칩 외부에서 인터럽트의 종류를 알리는 레벨만 정해주고 칩 내부에서 처리할 프로세서를 지정하고 제어하는 일을 담당한다. 칩 내에서 프로세서 상호간 인터럽트와 소프트 인터럽트는 프로세서 ID와 인터럽트 레벨을 지정해서 요청하되 외부 인터럽트 처리기를 통해서 요청하도록 한다. 한 칩에서 다른 칩으로 인터럽트를 요청할 때는 외부 인터럽트 처리기를 통해서 요청하도록 한다.

넷째, 실행 도중 에러가 발생하면 에러의 상황에 따라 칩 전체를 리세트 시키거나, 개별 프로세서에 대해서만 리세트 시킬 수 있다.

Ⅲ. 시스템 제어 장치의 구조

시스템 제어 장치는 칩내 4개의 프로세서에서 수행되는 쓰레드들간의 동기를 수행하며, 전역 변수들의 잠금처리, 명령어들의 동기, 인터럽트의 제어, 클럭 발생, lock 관련 트랩발생, 리세트 처리, 프로세서 상태 감시 및 제어들과 같은 기능을 가지도록 설계되었다.

설계된 시스템 제어 장치는 그림 2와 같이 동기화 모듈, 인터럽트 모듈, 클럭 발생 및 리세트 모듈, 프로세서 상태제어 모듈 등 4개의 모듈로 구성되어 있다.

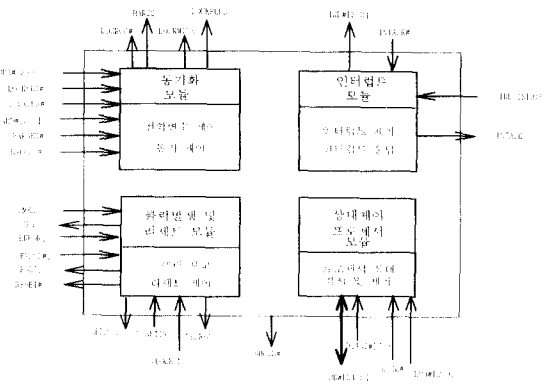


그림 2. 시스템 제어 장치의 전체 구조
Fig. 2. Architecture of the system control unit.

1. 동기화 모듈

동기화 모듈은 전역변수들의 잠금처리와 명령어들의 동기화를 위해 설계된 모듈로서, 그림 3과 같이 4개의 프로세서 사이의 우선 순위를 정해주는 arbiter 회로와 전역변수의 주소를 저장하여 잠금처리를 수행하는 lock array, 각 프로세서의 barrier 상태를 저장하고 제어하는 barrier register들로 구성되어 있으며, 그 세부적 기능은 다음과 같다.

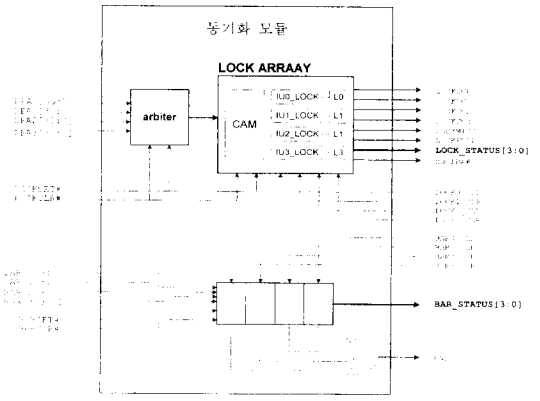


그림 3. 동기화 모듈의 블럭도
Fig. 3. Block diagram of synchronization module.

① 사용을 제한하기 원하는 전역 변수의 주소를 lock array에 저장한다.

② CAM(Content Addressable Memory)을 사용하여 lock 설정 여부를 판별한다.

③ 특정 전역 변수에 대한 잠금 주체를 각 프로세서별로 기록한 레지스터를 두어 프로세서 상태제어 모듈로 잠금 상태 정보를 전달한다.

④ 4개의 프로세서중 2개 이상이 동시에 접근해 올 때 arbitration을 수행한다.

⑤ 프로세서 사이의 동기화를 위해 Barrier 레지스터를 사용하고 그 정보를 프로세서 상태 제어 모듈로 전달한다.

⑥ Lock array에 관련된 트랩을 발생한다.

- Lockfull : lock variable을 32개 초과한 경우에 발생한다.
- Lockmiss : lock array에 해당 전역변수의 주소가 없는 경우나 lock이 설정되기 이전에 sylr 명령어를 수행할 경우에 발생한다.

2. 인터럽트 모듈

인터럽트 모듈은 프로세서간의 인터럽트 요청과 외부에서의 인터럽트 요청 시 인터럽트를 적절히 분배하기 위해 설계된 모듈로서, 프로세서를 선택하고 배분하는 분배기/선택기와 프로세서들의 인터럽트 수행여부를 기록하는 in_service 레지스터로 구성되어 있으며 그 세부적인 기능은 다음과 같다.

① 칩 내부에서 원하는 프로세서를 지정하여 인터럽트를 요청할 경우에는 칩내 해당하는 프로세서에 인터럽트를 전달한다.

② 칩 외부에서 프로세서를 지정하지 않고 인터럽트를 요청할 경우에는 인터럽트 배분기의 역할을 담당한다.

③ 인터럽트 acknowledge 신호를 전달한다.

3. 클럭 발생 및 리세트 모듈

클럭 발생 및 리세트 모듈은 시스템 클럭의 신호를 제공하고 시스템의 초기화 및 에러상태 보고와 프로세서 리세트기능을 제공하기 위해 설계된 모듈로서, 클럭 발생기, 프로세서 리세트기, 시스템 리세트기로 구성되어 있으며, 그 세부적인 기능은 다음과 같다.

① 100MHz 클럭 신호를 발생한다.

② Power on 리세트때나 시스템 리세트때 리세트 신호를 발생하고 시스템 초기화를 수행한다.

③ 각 프로세서에서 명령어 수행중 에러가 발생하여 에러 상태에 들어간 경우에는 해당하는 프로세서만 리세트한다.

- 개별 프로세서 에러가 발생하면 리세트 모듈은 에러 상황을 외부에 보고한다.
- 프로세서 상태 제어 모듈을 통하여 에러가 발생한 프로세서에 리세트 트랩을 요구한다.
- 리세트 트랩을 수행하는 과정에서 개별 프로세서의 문제를 해결하지 못하고 또 다른 에러를 유발시킬 경우에는 시스템 에러를 발생시킨다.

④ 시스템 에러 발생 시에는 칩 전체를 리세트한다.

- 버스 에러나, 메모리 에러 등 시스템에 관련된 에러가 발생하면 리세트 모듈은 시스템 에러상황임을 외부에 보고하고 외부로부터의 리세트 입력을 받아서 전체 칩을 리세트 시킨다.

4. 프로세서 상태 제어 모듈

프로세서 상태 제어 모듈은 다중 프로세서의 적절한

동작을 위해 칩내 각 프로세서의 동작을 원하는 상태로 제어할 수 있도록 설계된 모듈로서, 그림 4와 같이 arbiter 회로와 프로세서 상태/제어 레지스터, 프로세서 ID 레지스터, 인터럽트 레지스터, 전역변수 잠금 상태/제어 레지스터, barrier 상태/제어 레지스터로 구성되어 있으며, 그 세부적인 기능은 다음과 같다.

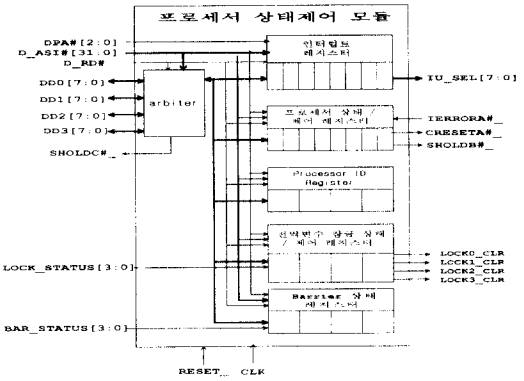


그림 4. 프로세서 상태 제어 모듈의 블럭도
Fig. 4. Block diagram of processor status control module.

① 프로세서 제어 레지스터를 사용하여 각 프로세서의 동작을 제어하는데 외부에서 레지스터 값을 바꾸어 줌으로써 해당하는 프로세서의 상태를 제어한다.

② 프로세서 상태 레지스터에 각 프로세서의 동작상태를 저장하여 두고 감시한다. 칩 외부에서 레지스터의 값을 읽어보면 칩 내 해당하는 프로세서의 상태를 알 수 있다.

③ 인터럽트 레지스터를 사용하여 칩 내의 한 프로세서가 다른 프로세서에게 인터럽트를 요청할 때 특정 프로세서를 지정한다.

④ 프로세서 ID 레지스터를 두어 칩내의 프로세서가 lduba 명령어를 통해 자신의 ID를 물어올 경우 ID를 알려준다.

⑤ 전역 변수 잠금 상태 레지스터에 동기화 모듈에서 보내온 프로세서의 전역 변수 잠금여부를 기록해두어 하나의 프로세서의 에러 상태로 인해 모든 정수 연산 장치가 deadlock에 빠지지 않도록 에러가 발생한 프로세서가 걸어둔 lock 설정을 해제할 수 있도록 한다.

⑥ Barrier 상태 레지스터를 두어 특정 프로세서가 barrier를 해제하지 않은 채로 에러 상태로 들어간 경우 이것을 읽어보고 대처할 수 있도록 한다.

Ⅳ. 시뮬레이션 및 검증

시스템 제어 장치는 Verilog-XL을 사용하여 최상위레벨로 기술되었으며, 시스템 레벨에서 보다 정확한 검증을 위해 외부 인터럽트 처리기와 트랩 처리기를 추가하여 다중 프로세서 칩 환경을 구현하였다. 명령어는 외부 메모리에서 명령어 캐시들로 미리 저장되어 있고 시스템 리세트 신호에 의해 각 시스템들이 초기화되어 동작하도록 되어 있다.

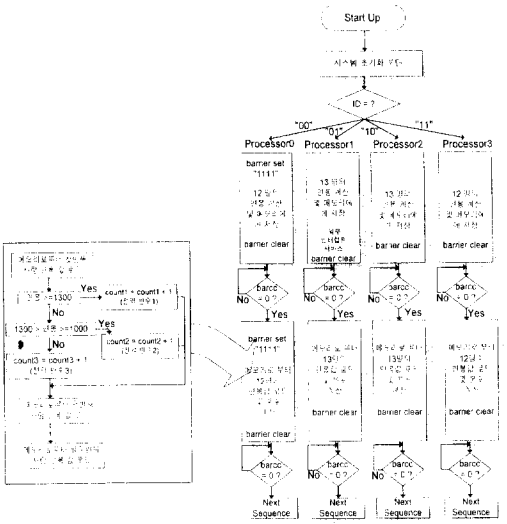


그림 5. 응용 프로그램의 시뮬레이션 흐름도
Fig. 5. Flow chart of simulation for application program.

위와 같은 환경 하에서 검증을 위한 응용 프로그램은 C-언어로 기술되었으며 이를 코드의 병렬화에 중점을 두어 어셈블하고 그 명령어 코드를 네 개의 캐시에 각각 스케쥴하여 시뮬레이션을 수행하였다. 통합 시뮬레이션에 사용한 응용 프로그램은 50명의 1년간의 월수입을 합산하여 각 사람의 연봉을 구하고, 분포(1300이상, 1000-13000, 1000이하)를 화면에 출력하는 프로그램이다. 시뮬레이션은 크게 시스템 초기화, 전역 변수 경쟁접근시 쓰레드 제어상황, 정규적인 프로세서 사이에서 발생하는 비동기 인터럽트 처리상황, 개별 프로세서 에러와 시스템 에러 발생시 대처상황 등의 다중 프로세서 동작 시나리오의 네 가지 영역들

에 대해 순차적으로 검증하였으며 시뮬레이션 흐름도는 그림 5에 나타내었다.

Supervisor mode에서 실행되는 시스템 초기화 루틴의 내용은 각 프로세서 내 정수 연산 장치의 특수 레지스터들인 psr, wim, tbr 레지스터들의 셋팅 및, 시스템 제어 장치 내의 ID 레지스터 값을 로드하여 프로세서 자신의 ID 확인 후 개개의 프로그램 시작 번지로의 분기를 포함하며 시뮬레이션 결과는 그림 6과 같다.

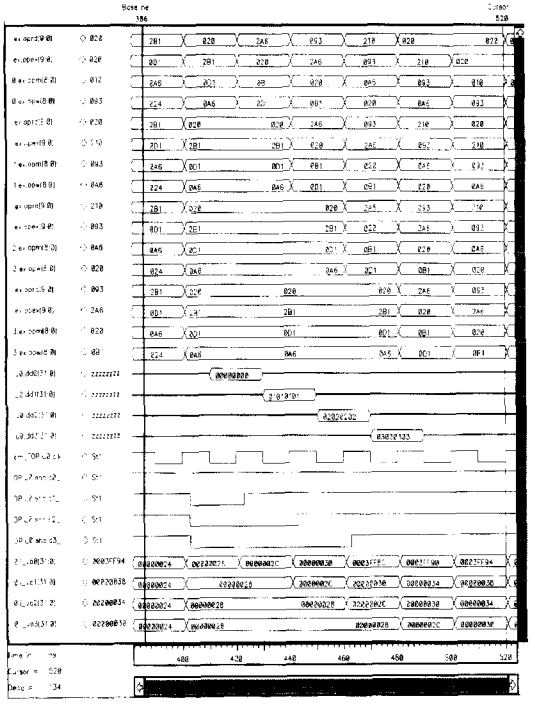
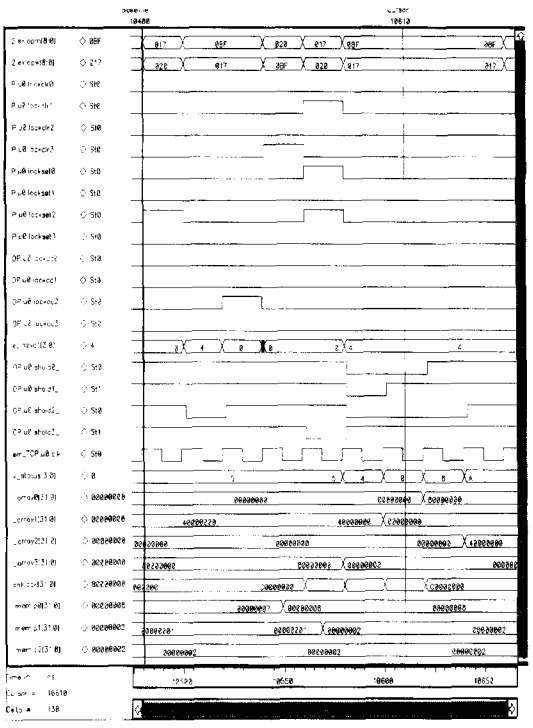
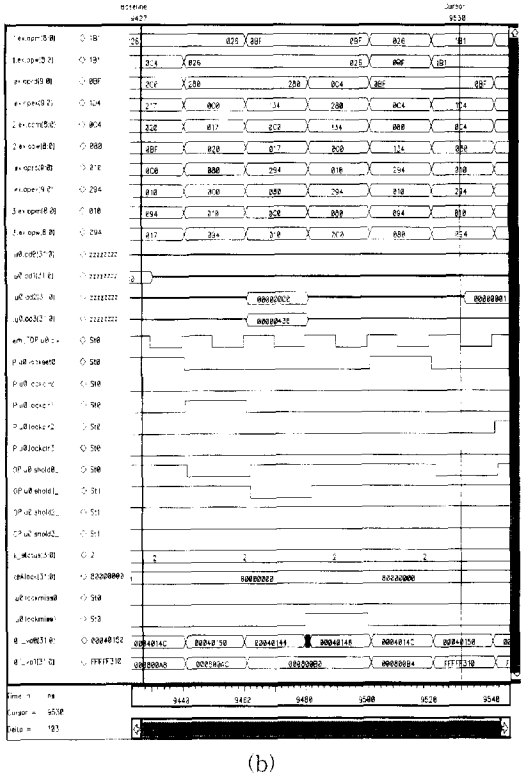


그림 6. 시스템 초기화 루틴
Fig. 6. System initialization routine.

그림 6에서 oprd, opex, opm, opw는 각 프로세서 내 정수 연산 장치의 5단 파이프라인 중 파이프라인 각 단계(R, E, M, W)별 명령어의 인코드된 값을 나타내며, dd0, dd1, dd2, dd3 값이 410ns, 430ns, 450ns, 470ns에 "00", "01", "10", "11" 값들이 나타나는 것을 볼 수 있는데, 이는 각 프로세서에서 로드 명령어인 lduba를 실행함에 따라 시스템 제어 장치 내의 ID 레지스터 값이 각 프로세서로 load되는 것을 보여준다. 460ns에 프로세서 0가 자신의 ID확인 후 0x00000030에서 자신에게 할당된 응용 프로그램의 시작번지인 0x0003ff8c로 분기하는 것을 알 수 있다.



(a)



(b)

그림 7. 전역변수들의 잠금처리 및 해제
Fig. 7. Locking & unlocking shared variables for protected access.

User mode에서 실행되는 응용 프로그램의 주요 시뮬레이션 결과는 그림 7, 8, 9에 각각 나타내었으며 내용은 다음과 같다.

그림 7은 50명의 연봉 분포를 구하기 위한 전역 변수 갱신과정 중 전역변수 접근시 쓰레드들의 제어 상황 일부를 나타낸다.

그림 7(a)에서 프로세서 2는 10480ns에서 전역 변수의 값을 갱신하기 위해 lock을 요청했지만(lockset2 = '1') 이전에 이 번지는 다른 정수 연산 장치에 의해 잠금처리 되어졌기 때문에 시스템 제어 장치의 동기화 모듈에서는 이미 잠금처리 되어있다는 신호(lockcc = '1')를 프로세서내의 상태 레지스터로 보내어 zero flag값을 '0'으로 셋팅하게된다(10520ns). 그 결과 프로세서 2는 이를 체크하여 다시 10560ns에 다시 lock을 요청하는 것을 볼 수 있다. 10560ns에 전역변수 잠금 처리 신호 lockset0, 2와 전역변수 잠금 해제 신호 lockclr1이 동시에 '1'이 되었을 경우 시스템 제어 장치는 lockclr, 프로세서 0에 우선 순위를 둔 arbitration을 수행하여 lockclr1, lockset0, lockset2 순으로 매클럭 마다 순차적으로 전역변수 접근을 처리하게 된다.

우선 프로세서들의 파이프라인을 일시적으로 정지하기 위한 shold0_, shold1_, shold2_는 '0'이 되고(10580ns), shold신호들을 프로세서 1, 0, 2 순으로 순차적으로 '1'로 풀어주어 프로세서 0, 1, 2는 정상적인 파이프라인으로 복귀하게 된다(10600ns, 10620ns, 10640ns). 또한 파이프라인이 복귀되는 시점에서 각각의 프로세서에 대한 lock 관련 정보를 지니는 레지스터들(_array0 [31:0], _array1 [31:0], _array2 [31:0], _array3 [31:0])은 전역 변수의 물리적인 번지를 저장하는 CAM에 대응되는 위치에 셋팅 및 클리어 된다(10600ns, 10620ns, 10640ns). 그림에서 chklock [31:0] 은 동기화 모듈내 CAM의 잠금처리 설정여부를 나타내는 최상위 비트이고, p0 [31:0], p1 [31:0], p2 [31:0] 는 전역변수들로서 각각 연봉이 1300 이상, 1300-1000, 1000이하인 분포결과를 나타내며 전역 변수 잠금처리를 해제하기 이전에 update된다(10550ns, 10570ns).

그림 7(b)는 동기화 모듈내의 CAM내에 없는 번지를 가진 전역 변수를 프로세서 1이 클리어 할 경우를 나타낸다. 9440ns 때 이는 shold1_이 '1'로 바뀌는 시점인 9480ns에서 lockmiss1신호가 '1'로 셋팅되어진다. 이에 따라 lockmiss 트랩이 발생하여 프로세서 1은 9520ns에서 트랩을 처리하기 위해 0xfffff310번지로 분기하게 된다.

그림 8은 응용 프로그램 실행 도중 칩 외부에서 비동기적인 소프트 인터럽트가 외부 인터럽트 제어기를 통해서 걸어 왔을 경우를 나타낸다.

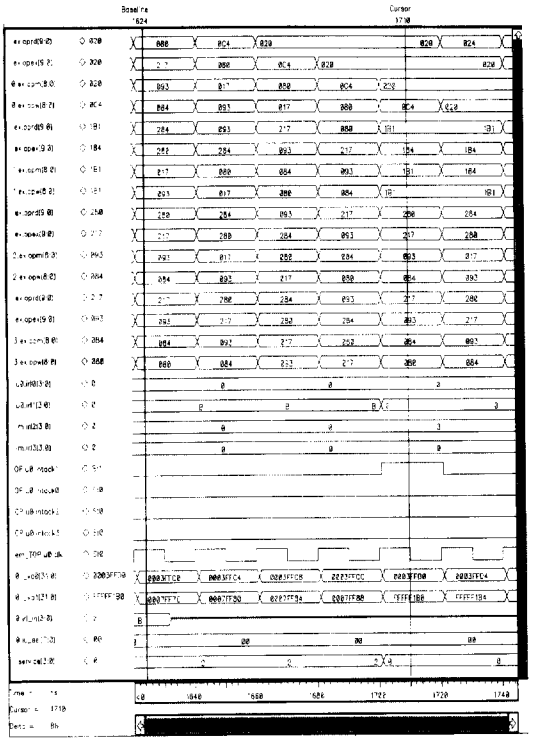


그림 8. 칩 외부에서 비동기적인 인터럽트 요청
Fig. 8. Asynchronous interrupt request that occurs in external chip.

1620ns에서 irl_in [3:0] 은 외부 인터럽트 제어기로부터 칩으로 요청되는 인터럽트 레벨을 의미하며, 레벨 11(B)임을 알 수 있다. 이 때 시스템 제어 장치내 인터럽트 모듈은 이 인터럽트 레벨을 인터럽트 서비스를 수행하고 있지 않는 임의의 프로세서로 분배하여 1700ns때 프로세서 1로부터 인터럽트를 인식했다는 신호 intack1이 발생하는 것을 볼 수 있으며, 이에 따라 프로세서 1은 인터럽트를 처리하기 위해 명령어 번지인 i_val [31:0] 이 0xfffff1b0값으로 분기하는 것을 확인할 수 있다.

그림 9는 개별 프로세서가 응용 프로그램 실행 도중에 에러를 발생했을 경우 제어 상황에 대한 시뮬레이션 결과를 나타낸다.

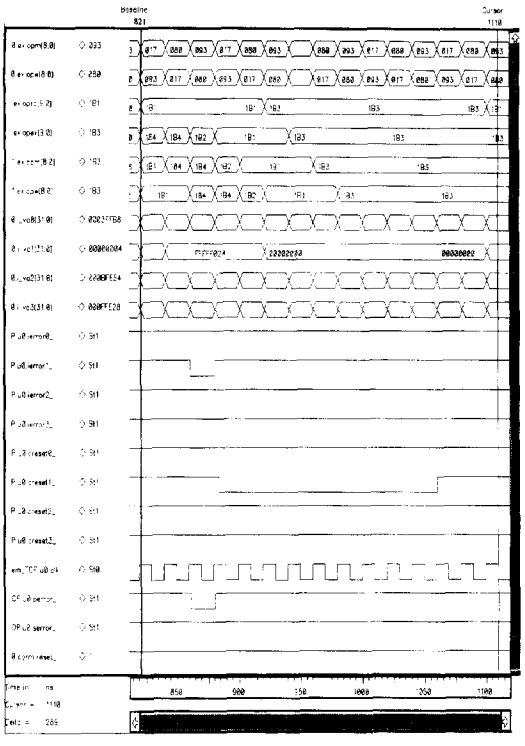


그림 9. 개별 프로세서 에러발생
Fig. 9. The occurrence of individual processor error.

860ns때 프로세서 1이 프로그램 수행 도중에 에러가 발생하는 것(ierror1_= '1')을 볼 수 있는데, 시스템 제어 장치 내 클럭 발생 및 리세트 모듈에서는 개별 프로세서 에러라는 것을 판별하여 외부에 보고하고(perror_ = '0'), 880ns 때 creset1_신호를 8클럭 동안 '0'으로 셋팅함으로써 프로세서 1의 명령어 번지(i_va [31:0])는 920ns때 시스템 초기화 루틴의 시작번지 0x00000000으로 변하게 되고, 이에 따라 프로세서 1은 다시 초기화 루틴을 수행하여 정상 상태로 복귀하는 것을 볼 수 있다.

위의 통합 시뮬레이션을 통하여 시스템 제어 장치의 도움으로 4개의 프로세서들이 충돌 없이 원활하게 자신에게 할당된 쓰레드들을 처리하는 것을 확인하였다.

V. 성능 분석

공유메모리 다중 프로세서 환경에서는 프로세서간의 통신이 데이터 구조를 공유함으로써 이루어진다. 따라서 공유된 데이터의 일관성을 유지하고 쓰기 충돌 방지 및 쓰레드의 동기화를 맞추는 것이 중요하다. 이를 위해 공유 메모리 다중 프로세서 환경에서는 번망-대기(Busy-wait) 동기화 기술을 사용하는 것이 일반적이다. 가장 널리 사용되는 번망-대기 동기화 기술은 스핀 락(spin locks)과 베리어(barriers)이다. 스핀 락은 공유된 데이터에 대한 상호 배타적 접근 수단을 제공한다. 보통 이진 세마포아(binary semaphore)를 사용하여 구현하며, atomic load-store와 같은 메모리 접근 명령어로 처리할 수 있다[6] [7]. 동기화 작용이 지역 캐시에서 실행되어진다면 원시(atomic) 메모리 접근 명령어들은 본래의 수행 사이클에 캐시에 있는 전역 변수에 대한 일관성 점검과 배타적 접근 보장을 위한 추가 사이클이 요구된다. 또한 실제적인 동기화 동작 전체를 따져 보면, 페치한 상태 코드 값에 따라 프로세서 상태 레지스터의 값을 변경시켜 프로세서의 다음 동작을 결정하도록 상태코드 변경 명령어를 추가해야 하므로 한 싸이클이 추가된다. 캐시 상에서 실행되는 위와 같은 잠금 작용과 본 논문에서 정의한 칩내의 제어 장치 상에서 실행되는 잠금 작용을 비교하면 그림 10과 같으며, 왼쪽은 명령어의 번지를 나타낸다.

원시 메모리 접근 명령어 사용	시스템 제어 장치상에서 상태코드 변경 명령어 사용
00003fc0 : sethi 80000,%5 00003fc4 : ldst %l3 00003fc8 : nop 00003fcc : andcc %l3,%l3, %l5 00003fd0 : bne {00003fc4}	00003fc0 : sylscc 00003fc4 : nop 00003fc8 : bne {00003ffc0}

그림 10. 전역 변수 잠금처리시 어셈블리 라인 비교
Fig. 10. Comparison of assembly line when shared variable is locked.

다중 프로세서 구조에서 메모리의 최상위 비트는 잠금처리 수행여부를 나타내기 때문에 원시 메모리 접근 명령어(ldst)를 사용할 때는 그값을 비교하여 잠금처리 수행여부를 알 수 있고, 칩 내의 시스템 제어 장치 상에서 상태코드 변경 명령어(sylscc)를 사용할 때에는 sylscc명령어를 수행 후 전역변수 잠금 여부가 상태코드에 바로 나타나기 때문에 그 상태코드를 가지고서 전역변수 잠금처리 여부를 알 수 있다. 그림에서 보듯이 칩 내의 시스템 제어 장치 상에서 전역변수 잠금처리를 할 경우 어셈블리 라인의 수는 2싸이클 이상 감소하고, 수행 싸이클이 많은 메모리 접근을 하지 않기 때문에 잠금처리 수행 싸이클을 감소시키게 된다. 전

역 변수 잠금 작용의 구체적인 수행 싸이클 비교는 표 1에 나타내었다.

표 1. 전역 변수 잠금 작용의 수행 싸이클 비교

Table 1. Comparision of execution cycle for shared variable locking.

(단위 : 싸이클)

전역변수동시 접근 상태	① 캐시에서 실행			② 설계된 장치에서 실행	
	캐시 히트		캐시 미스	잠금된 개수 32 이하	잠금된 개수 32 초과
	비동일 변수	동일 변수			
프로세서 1개	5	5	13	2	33개째부터는 캐시에서 실행해야 하므로 ①의 경우와 동일함
프로세서 2개	5	6	13	3	
프로세서 3개	5	7	13	4	
프로세서 4개	5	8	13	5	

표 1에서 보는 바와 같이 시스템 제어 장치를 사용하는 경우 하나의 프로세서가 접근하는 경우에는 2싸이클에 처리가 되고 여러개의 프로세서가 동시에 접근하는 경우에는 arbitration 때문에 싸이클이 하나씩 추가된다. 4개의 프로세서가 동시에 접근하는 경우에만 캐시에서 실행할 때와 동일한 수행 사이클이 걸리고, 그 외의 경우에는 설계한 제어 장치가 적은 사이클 수를 보임으로써 성능이 향상됨을 알 수 있다. 이러한 성능 향상은 잠금 처리된 전역 변수에 대한 접근 시도 횟수가 늘어날수록 증대될 것이다. 만약 캐시 미스가 발생할 경우를 고려한다면 시스템 제어 장치를 사용한 경우의 이득은 더욱 클 것이다.

본 논문에서 설계한 시스템 제어 장치 내의 CAM이 동시에 최대 32개까지 락을 걸 수 있으므로 32개를 초과하면 지역 캐시를 사용해서 처리해야한다. CAM에 저장될 수 있는 전역변수의 주소를 32개로 제한한 것은 칩 크기와 성능사이의 trade-off 때문이었다.

쓰레드의 동기화를 보장하는 베리어 기능의 구현에 대한 연구는 전역 변수 잠금 처리에 대한 연구보다 오히려 증대되고 있으며 구현되는 시스템의 사양에 따라 매우 다양하게 여러 가지 알고리즘이 개발되고 있다. 따라서 본 논문에서 설계한 방식과 기존의 방식을 비교하는 일은 간단하지 않다. 하지만 분명한 차이점은 본 논문에서는 칩 내 4개의 프로세서간의 동기화를 위한 barrier register를 시스템 제어 장치 내에 제공하여 하나의 명령어에 의해서 barrier 해지 및 조건코드 변경이 이루어지게 한 것이다. 메모리 접근 명령어를 사용하는 방식처럼 barrier가 설정된 후 계속해서 barrier가 설정되는 nesting을 지원하지는 않지만, 다중 프로세서 칩내 프로세서 4개에서 수행되는 쓰레드들의 barrier 설정은 메모리 접근 없이 처리함으로써 수행싸이클을 크게 줄이고, 제어 알고리즘을 간단하게 할 수 있다. 그림 11은 원시 메모리 명령어를 사용하는 메커니즘과의 어셈블리 라인을 비교한 것이다.

원시 메모리 접근 명령어를 사용	시스템 제어 장치상에서 상태코드 변경명령어 사용
00458a00 : sethi 0001, %l5 00458a04 : ld %l3, count 00458a08 : sub %l3, 1 00458a0c : sethi 0001 00458a10 : andcc %l3, %l3, %l5 00458a14 : bne {00458a00} 00458a18 : stb 'global_sense, 0 next sequence	00458a00 : sybrcc 00458a04 : nop 00458a08 : bne {000458a00} next sequence

그림 11. 베리어 설정시 어셈블리 라인 비교

Fig. 11. Comparison of assembly line when barrier is built.

그림 11에서 보는 바와 같이 본 논문에서 설계된 구조는 메모리 상에서 실행되는 베리어 작용에 비해 두 싸이클 이상의 실행 시간을 단축시켰다.

VI. 구현 및 결과

검증이 완료된 시스템 제어 장치는 합성이 가능하도록 각 모듈별로 더욱 세분화된 RTL(Register Transfer Level) 레벨로 수정하였으며, 이 변환된 모델을 Xilinx FPGA 셀 라이브러리로 targeting하여 Synopsys 툴을 사용하여 자동 합성을 수행하였다[8].

FPGA 구현을 위해 합성된 네트리스트(netlist) 파일을 Xilinx사의 XACT 툴로 옮겨 처음에는 회로의 네트리스트 파일로부터 배치, 배선과 칩에 다운로드할 파일 (.bit)까지 자동으로 생성해 주는 xmake를 사용하였다. 그리고 상세한 post-placement와 post-routing 타이밍 정보를 얻기 위해 XACT 툴에서 제공하는 XDelay 명령어를 사용하여 성능 측정결과 시간지연이 생각보다 많이 나와 회로의 critical path 부분

을 수정하고, 배선 결과의 congestion이 많이 발생하는 부분을 재배치시킨 결과 시간 지연을 개선할 수 있었다. 구현된 FPGA 칩은 시스템 제어 장치의 총 입출력 핀 수가 160핀이고, 게이트 수가 약 15,000게이트임을 감안해 XC4025epg299로서 구현하였다.

다중 프로세서 칩 내의 주처리 장치인 정수 연산 장치는 기존의 SPARC-8 정수 연산 장치에 시스템 제어장치와 관련된 여러 동기화 명령어들이 추가된 구조이고 Verilog HDL로 모델링하여 기능을 검증하였으나 아직 칩으로 구현되지 않아 현재는 구현된 시스템 제어 장치와 직접 연결하여 칩테스트를 할 수는 없는 실정이다. 그러므로 칩 검증환경은 Verilog HDL을 사용한 시스템 통합시뮬레이션을 통해 얻어진 행위 수준 동작 결과들로부터 시스템 제어장치로 들어오는 비트 패턴들을 분석하여 16k×8 EPROM 15개에 테스트 패턴을 저장하고 8086 CPU와 8255등으로 구성된 테스트 키트와 시스템 제어 장치를 연결하여 구성하였다. EPROM으로의 주소 발생은 키트내 8255 (PPI) 2개의 입력을 참조하여 발생하도록 어셈블리어로 작성하여 부분적으로 테스트 키트와 시스템 제어 장치가 상호 동작하도록 환경을 구성하였으며, 칩 테스트 환경은 그림 12에 나타내었다.

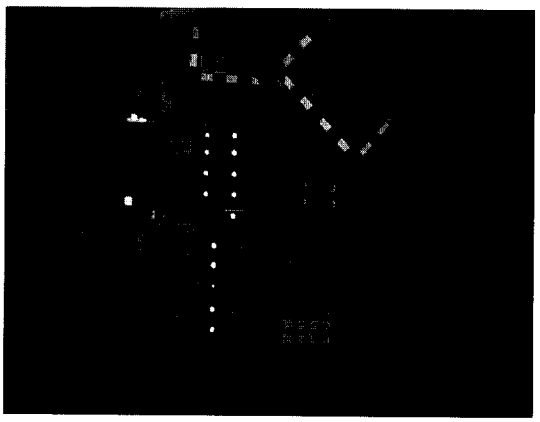


그림 12. FPGA 칩 테스트 환경
Fig. 12. FPGA chip test environment.

어드레스 발생기인 테스트 키트에서 up/down 카운터 방식으로 처음 0번지를 EPROM에 내보내고, 그에 대응하는 EPROM에 저장된 비트 패턴이 시스템 제어 장치에 들어가게 되며, 이 비트 패턴에 해당되는 시스템 제어 장치의 출력 신호를 logic analyzer를 사용하여 측정하였다. 또한 시스템 제어 장치의 출력 결과에 따라 어드레스가 1씩 증가하든지 또는 지정된 값으로 분기하여 계속 비트 패턴을 내보내 원하는 출력 신호들이 나오는지 측정함으로써 칩이 원하는 데로 동작하는 지를 확인하였다.

Ⅶ. 결 론

본 논문은 대규모 병렬 처리 시스템에서 효과적으로 사용될 수 있는 전용 마이크로프로세서인 다중 프로세서 칩의 시스템 제어 장치의 구조에 대한 연구 결과이다. 다중 프로세서 칩은 프로세서 4개를 메모리 공유 방식으로 하나의 칩에 집적시킨 구조인데, 정수 연산 장치, 명령 캐시, 데이터 캐시, 메모리 관리 장치, 버스 제어 장치 그리고 시스템 제어 장치로 구성되었다.

시스템 제어 장치는 4개의 프로세서와 함께 한 칩에 집적되어 쓰레드들간의 동기적 수행을 보장하며 동기화 기능을 담당하도록 설계되었지만, 연구수행과정에서 시스템 제어 장치만을 별도로 제작하게 되었다. 먼저 각 모듈을 Verilog HDL로 상위 수준에서 기술한 후 명령어 수준에서 모듈별로 검증하였다. 다음에 시스템 제어 장치 전체 모듈에 대한 검증을 하였는데, 다중 프로세서 칩 내에서 관련된 다른 장치들과 상호 인터페이스 하는 부분을 중점적으로 검증하였다. 상위 수준에서 검증된 모델은 게이트 수준에서 Xilinx FPGA 셀 라이브러리를 사용하여 Synopsys tool로 자동 합성하고 타이밍 검증 후 Xilinx FPGA XC4025epg299로서 구현하였다.

본 논문에서 소개한 시스템 제어 장치가 다중 프로세서 칩 내에 구현되면 외부 제어 핀 수를 크게 줄일 수 있고 전체적인 성능 향상을 가져올 것이다.

참 고 문 헌

[1] Kai Hwang, Advanced computer Architecture : Parallelism Scalability Programmability, McGraw-Hill, 1993.

[2] J. Hennessy et al., "Panel: beyond superscalar," Records of Sixth Annual Microprocessor Forum, MicroDesign Resources, Oct. 1993.

[3] J. A. Fisher and B. R. Rau "Instruction-level parallel processing," Science, vol. 253, no. 5025, pp. 1233-1242, Sept.

1991.

[4] L. G wennap, "Microprocessors head toward MP on a chip," Microprocessor Report, vol. 8, no. 6, pp. 18-21, May 1994.

[5] Weaver, L. David, editor, The SPARC Architecture Manual, Version 8, Prentice-Hall, Inc., 1992.

[6] Michael Dubois and Christoph Scheurich, "Synchronization, Coherence, and Event Ordering in Multiprocessor", IEEE Trans. Computer, vol. 21, no. 2, pp. 9-21, Feb. 1988.

[7] J. M. Meller-crummey and M. L. Scott, "Algorithms for Scalable Synchronization on Shared-Memory Multiprocessor", ACM Trans. Computer Systems, vol. 9, no. 1, pp. 21-65, Feb. 1991.

[8] Xilinx, Inc., Synopsys for FPGAs Interface /Tutorial Guide, 1995.

저 자 소 개

朴 性 模(正會員)

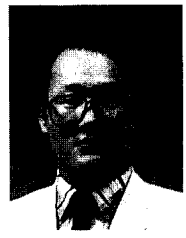
1977년 서울대학교 전자공학과 학사. 1979년 한국과학기술원 전기 및 전자공학과 석사. 1988년 노스캐롤라이나 주립대학 전기 및 컴퓨터공학과 공학박사. 1979년 ~ 1984년 한국전자통신연구소 반도체단 설계개발부 선임연구원. 1988년 ~ 1992년 올드도미니언 대학교 전기 및 컴퓨터공학과 조교수. 1992년 ~ 현재 전남대학교 컴퓨터공학과 부교수. 1992년 ~ 1994년 전남대학교 전자계산소 연구개발부장 역임. 현재 전남대학교 정보통신 특성화추진센터 소장 겸임. 주관심분야는 마이크로프로세서, 멀티미디어 프로세서 구조, VLSI 시스템 설계, 신호처리용 ASIC 설계 등

鄭 揚 勳(正會員)

1995년 전남대학교 전자공학과 학사. 1997년 전남대학교 전자공학과 석사. 현재 현대전자 멀티미디어 연구소 연구 1실 재직 중. 주관심분야는 영상/음성 데이터의 부호/복호 알고리즘, 저전력 칩 설계, 고수준 설계자동화 도구 등임

丁 甲 天(正會員)

1996년 전남대학교 컴퓨터공학과 학사. 현재 전남대학교 전자공학과 석사 과정. 주관심분야는 저전력 프로세서 구조, 병렬 및 분산처리 시스템, VLSI 설계 및 CAD 등임



尹 碩 漢(正會員)

1977년 고려대학교 전자공학과 학사. 1986년 한국과학기술원 전산학과 석사. 1995년 고려대학교 전자공학과 박사(컴퓨터공학 전공). 1977년 2월 ~ 1985년 11월 한국전자기술연구소 선임연구원. 1985년 11월 ~ 현재 한국전자통신연구원 책임연구원(시스템연구부 부장)으로 재직. 주관심분야는 고성능 마이크로프로세서 구조, 다중처리 컴퓨터 구조, 병렬 처리 컴퓨터 구조 분야, 멀티미디어 처리 시스템 등임

論文97-34C-12-3

10 비트 CMOS Algorithmic A/D 변환기를 위한 저전력 MDAC 회로 설계

(A Low-Power Multiplying D/A Converter Design for 10-bit CMOS Algorithmic A/D Converters)

李 宰 燁 *, 李 承 勳 *

(Jae-Yup Lee and Seung-Hoon Lee)

요 약

본 논문에서는 algorithmic 구조를 가진 저전력 10 비트 CMOS A/D 변환기 (analog-to-digital converter : ADC)를 위한 multiplying digital-to-analog converter (MDAC) 회로를 제안하였다. 전체 회로는 전형적인 5 V 전원 전압에서 뿐만 아니라, 3 V 수준의 저전원 전압에서도 동작이 가능하도록 설계하였으며, 바이어스 스위칭 회로를 응용한 전력 최소화 기법 (power reduction technique)을 적용하여 전체 전력 소모를 극소화 하였다. 또한, 제안된 회로는 온-칩 보상 회로 (tuning circuit)를 내장하여 공정상에서 큰 변화폭을 갖는 metal-to-metal 캐패시터로 인해 가능한 연산 증폭기의 전반적인 성능 변화를 보상 가능하게 하였다. 제안된 저전력 MDAC을 응용한 10 비트 200kHz algorithmic A/D 변환기는 0.6 um single-poly double-metal n-well CMOS 공정을 사용하여 제작되었으며, 측정 결과 전력 최소화 기법 응용시 3.3 V 전원 전압에서 전체 전력 소모는 7 mW로 기존 구조의 11 mW에 비해 36 %의 전력 감소 효과가 얻어졌다.

Abstract

In this paper, a multiplying digital-to-analog converter (MDAC) circuit for low-power high-resolution CMOS algorithmic A/D converters (ADC's) is proposed. The proposed MDAC is designed to operate properly at a supply voltage between 3 V and 5 V and employs an analog-domain power reduction technique based on a bias switching circuit so that the total power consumption can be optimized. As metal-to-metal capacitors are implemented as frequency compensation capacitors, opamps' performance can be varied by imperfect process control. The MDAC minimizes the effects by the circuit performance variations with on-chip tuning circuits. The proposed low-power MDAC is implemented as a sub-block of a 10-bit 200kHz algorithmic ADC using a 0.6 um single-poly double-metal n-well CMOS technology. With the power-reduction technique enabled, the power consumption of the experimental ADC is reduced from 11 mW to 7 mW at a 3.3 V supply voltage and the power reduction ratio of 36 % is achieved.

I. 서 론

이동 통신용 단말기, 캠코더, 각종 멀티미디어 기기 등에 이르기까지 휴대용 장비의 사용은 점차로 증가 추세에 있으며, 그 응용 범위 또한 확대되고 있다. 이 장비들은 보다 긴 배터리 수명, 적은 전력 소모, 그리고 소형화 및 단소화를 우선적으로 요구하고 있으며, 최근 오디오 응용 분야에서는 이러한 추세에 부응하여 Σ-Δ ADC, algorithmic ADC 등과 같이 여러 형태의 데이타 변환기 설계가 이루어지고 있다. 이중 Σ

* 正會員, 西江大學校 電子工學科
(Dept. of Electronic Engineering, Sogang University)

接受日字:1996年12月4日, 수정완료일:1997年11月24日

-⊿ ADC의 경우 신호 처리를 위해 복잡한 디지탈 회로 및 내부 클락을 필요로 하고, 반면에, algorithmic ADC는 동일한 회로를 반복적으로 사용하는 구조를 가지므로 전력 소모 및 요구되는 회로 면적 면에서 보다 효율적인 형태로의 구현이 가능하다[1]-[7].

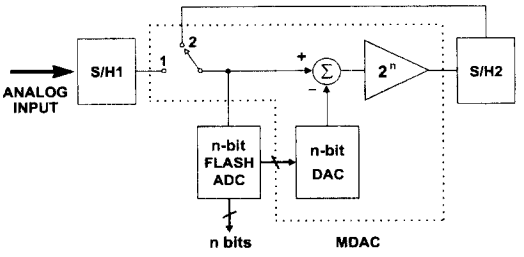


그림 1. 전형적인 algorithmic A/D 변환기의 블럭도
Fig. 1. Block diagram of a conventional algorithmic A/D converter.

그림 1은 전형적인 algorithmic A/D 변환기의 블록도이다. 전체 구조는 입력 신호를 선택하기 위한 스위칭 회로를 포함하여 두 개의 Sample-and-Hold 회로, 플래시 변환기 (FLASH ADC), 및 증폭기 (amplifier)로 구성된다. 신호 변환 과정은 모두 세 단계로 이루어지며, 첫번째 단계에서는 아날로그 입력 신호가 외부로부터 입력단 Sample-and-Hold 회로 (S/H1)에 샘플되어 저장된다. 두번째 단계로 S/H1에 저장된 신호는 단자 1을 통해 플래시 변환기에 전달되며, n-bit 디지탈 출력이 얻어진다. 이와 동시에, 전달된 아날로그 신호와 DAC (digital-to-analog converter)를 통해 재생된 양자화된 아날로그 신호의 차가 증폭기를 통해 2^n배 만큼 증폭되어 다음단의 Sample-and-Hold 회로 (S/H2)에 저장된다. 세번째 단계로 S/H2에 저장된 신호는 그림 1의 단자 2를 통해 다시 입력단으로 전달되며, 이후의 동작은 위의 두번째 및 세번째 과정의 순차적인 반복 과정으로 이루어진다. 이처럼 algorithmic A/D 변환기는 간단한 구조의 회로를 주기적으로 반복 동작시킴으로써 매 주기 (cycle)마다 순차적으로 n 비트의 디지탈 출력을 얻을 수 있으므로, 전력 소모 및 면적에서 가장 최적화된 형태 중의 하나로 알려져 있다. 그러나, 아날로그 입력 신호와 플래시 변환기 출력에 상응하는 양자화된 아날로그 신호와의 차이, 즉, 잔류 전압을 재 샘플링하는 과정이 여러 주기동안 반복되는 순환 구조를 가지므로, 다른 구조에 비해 소자 부정합 등 공정상에서 생기는 소자 크기 변화로 인한 영향이 크다[8]. 또한, 신호 변환을 위한 코딩 기법의 종류에 따라 첫번째로 얻어지는 디지탈 출력과 이후로 얻어지는 디지탈 출력이 상이할 수도 있으므로, 이들 각각에 대한 아날로그 회로의 상당 부분을 다르게 설계하여야 하는 제약이 따른다.

본 논문에서는 불안정한 공정상의 소자, 즉, metal-to-metal 캐패시터의 크기 변화로 인한 연산 증폭기의 성능 변화 범위를 최소화하기 위해 온-칩 보상 회로를 제안하고, 전력 소모를 최소화하기 위한 아날로그 영역에서의 전력 최소화 기법을 제시한다. II 장에서는 algorithmic A/D 변환기에의 응용을 위한 MDAC의 전체 구조 및 동작을 살펴보고, 제안된 보상 회로 및 전력 최소화 기법에 대해 논의한다. III 장에서는 제안된 회로 및 설계 기법들이 실제 10 비트 algorithmic A/D 변환기에 사용된 예를 보여주고, IV 장에서는 모의 실험을 통해 그 성능을 비교하며, V 장에서는 제안된 MDAC을 응용하여 제작된 10 비트 algorithmic A/D 변환기 측정을 통해 전력 최소화 기법의 효용성 및 성능을 검증한다.

II. 제안된 MDAC 구조

1. 전체 구조 및 동작 설명

그림 2는 제안된 MDAC의 블록도이다. 신호의 샘플링 및 오차 증폭 동작을 위해서 두 개의 중첩되지 않은 클럭 (nonoverlapping clock)을 사용하는 것은 기존의 MDAC과 동일하다[9]. 그러나, 컨트롤 신호 QC1 및 QC2를 추가로 사용함으로써, 입력 샘플링 (input sampling) 및 순환 (recycling)의 두 가지 모드로 동작할 수 있으며, 사용되는 연산 증폭기 (OP)에 성능 향상을 위해 제안된 보상 회로 및 전력 최소화 기법이 사용된다는 것이 기존의 MDAC 구조와 다르다. 최초 입력 신호를 샘플링할 때, 신호 입력은 컨트롤 신호 QC1에 의해 외부와 연결된 경로 1 (PATH 1)로 선택되어 입력 샘플링 모드로 동작한다.

그림 3은 입력 샘플링 및 순환 모드에서의 MDAC 동작을 보인다. 그림 3 (a)는 샘플링 동작으로 아날로그 입력 전압은 캐패시터의 아래 부분 (bottom plate)에 샘플되어 저장된다. 이때 증폭기의 입력단이 증폭기의 출력단과 스위치를 통해 연결되므로 캐패시

터의 윗 부분 (top plate)은 입력 신호의 공통 모드로 잡힘과 동시에 증폭기의 옵셋 전압이 샘플되어 저장된다.

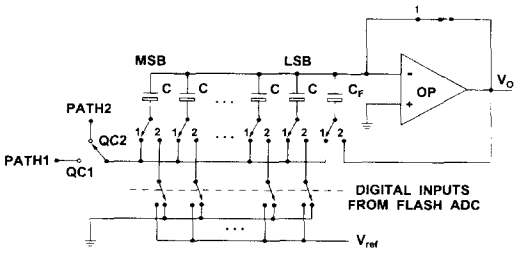


그림 2. 제안된 MDAC의 구성도
Fig. 2. Proposed MDAC configuration.

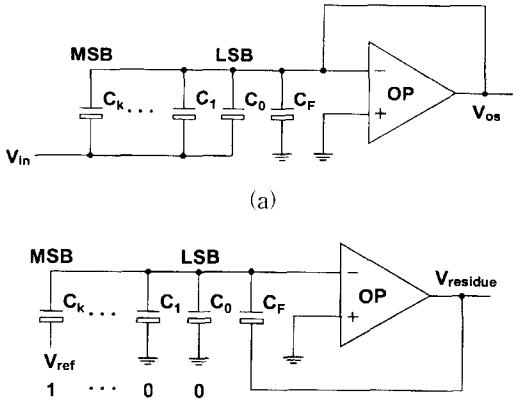


그림 3. 입력 샘플링 및 순환 모드에서의 MDAC 동작: (a) 샘플링 및 (b) 오차 증폭
Fig. 3. MDAC operation during input sampling and recycling mode: (a) sampling and (b) error amplification.

그림 3 (b)는 잔류 전압 (residue) 증폭 동작으로 캐패시터 아래 부분은 아날로그 입력 전압에 상응하는 플래시 변환기의 디지탈 출력 코드가 1 또는 0의 여부에 따라 기준 전압 (V_{ref})이나 접지 (GND)로 연결된다. 그림3 (b)는 그중 한 예로 플래시 변환기로부터 1…00의 디지탈 코드가 인가된 경우를 나타내며, 이 경우 앞서 샘플링 동작시 저장된 연산 증폭기 옵셋 전압은 자동적으로 상쇄되어 없어지고, 증폭기 출력단에는 재생된 아날로그 전압과 샘플된 입력 전압과의 차이인 잔류 전압이 (k+1)배로 증폭되어 나타난다. 이 증폭된 잔류 전압은 다음단의 Sample-and-Hold 증폭기에 저장되었다가 순환 모드 동작시 MDAC의 입력 신호로 다시 샘플된다. 그림 3 (a) 및 (b)의 모든 캐패시터는 단위 캐패시터이며, 다음의 식 (1)과 같은 관계를 갖는다.

$$C_0 = C_1 = \cdots = C_k = C_F = C \quad (1)$$

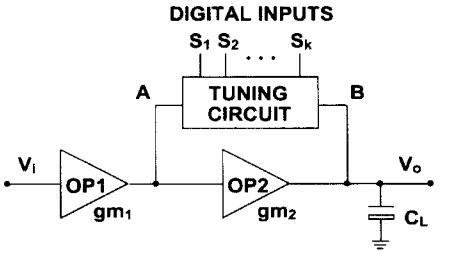
식 (1)에서 C_0, C_1,…, C_k 는 입력 신호를 저장하고, 접지 또는 기준 전압을 받아들이기 위한 샘플링 캐패시터 (sampling capacitor)이며, C_F는 증폭기를 폐루프 (closed-loop) 구조로 동작시키기 위한 피드백 캐패시터 (feedback capacitor)이다. 한편, 순환 모드에서는 샘플링 동작시 MDAC 출력 신호를 입력 신호로 재샘플링한다는 점을 제외하고는 입력 샘플링 모드에서와 동일한 샘플링 및 오차 증폭 과정을 거친다.

2. 제안된 보상 회로 및 전력 최소화 기법

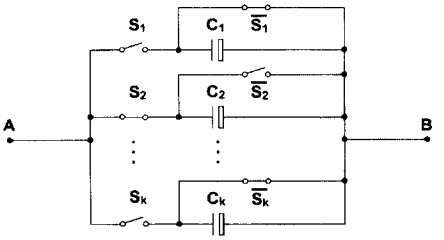
전체 회로는 0.6 um single-poly double-metal 구조의 standard CMOS 공정으로 설계되었고, 회로내 캐패시터는 모두 metal-to-metal 캐패시터가 사용되었다. 이 캐패시터는 두 metal 사이의 두꺼운 oxide 층으로 인해 공정상에서 그 크기를 정확하게 제어하기 어려우므로, 캐패시턴스는 최대 ± 30 %의 부정확성을 갖는다. 이러한 공정상의 변화는 실제 회로 동작을 의도된 설계 목표에서 벗어나게 하므로, 이러한 오류 범위를 최소화 하기 위해 그림 4 (a)와 같이 MDAC에 사용되는 연산 증폭기는 제안된 보상 회로를 사용하는 2단 증폭기로 구성되었다. 보상 회로에 적절한 디지탈 신호를 S_1, S_2,…, S_k에 인가하면, C_1, C_2,…, C_k 중 입력된 디지탈 코드에 상응하는 어느 하나의 캐패시터만 주파수 보상 캐패시터로서 작용하며, 그림 4 (b)는 보상 회로의 내부 구조로 캐패시터 C_2 만이 선택되는 경우를 보여준다. 따라서, 이처럼 외부로부터 간단하게 보상 캐패시터 크기의 변화 및 재조정이 가능하므로, 불안정한 공정상의 오차로 인한 소자 성능 변화에 따르는 증폭기의 성능 보상이 용이해지며, 특히 새로운 시스템 구조의 개발시 재설계로 인해 수반되는 공정 비용을 절감할 수 있다.

증폭기의 정적 전력 소모 (static power consumption)는 MDAC 전체 전력 소모의 대부분을 차지한다. 본 논문에서는 이러한 전력 소모를 최소화하기 위한 아날로그 영역 (analog domain)에서의 전력 최소화 기법을 적용하였다. MDAC이 algorithmic 구조와 같이 주기당 적은 비트 수가 얻어지는 ADC에

응용될 경우, 오프셋 샘플링을 통해 증폭기 입력단의 오프셋을 제거하지 않아도 ADC 전체 선형성에는 영향을 미치지 않는다. 따라서, 이 경우 클럭의 반주기인 샘플링 기간 동안 MDAC 내 증폭기를 사용하지 않을 수 있으므로, 증폭기 바이어스 회로에 의해 증폭기의 전력 공급을 차단하고, 나머지 반주기인 잔류 전압 증폭 기간 동안에는 다시 증폭기에 전류를 공급함으로써, 전체 전력 소모를 최소화할 수 있다. 그러나, 전형적인 방법으로 전력 소모만을 최소화할 경우, 연산 증폭기의 출력단에 순간적인 오버슈트 (overshoot) 및 위상 변화로 인한 불안정 동작을 수반하게 되어 높은 속도로 동작하는 경우 필요로 하는 최종 출력을 얻을 때까지 걸리는 settling 시간에 나쁜 영향을 끼치게 된다. 본 논문에서는 전력 소모를 최소화 하면서도 settling 시간에 영향을 주지 않도록 다음과 같이 증폭기의 바이어스 전류를 적절히 조절한다.



(a)



(b)

그림 4. 제안된 보상 회로의 블록도: (a) 보상 회로를 사용하는 2단 연산 증폭기 및 (b) 보상 회로 구조

Fig. 4. Block diagram of a proposed tuning circuit: (a) two-stage amplifier with the tuning circuit and (b) tuning circuit structure.

그림 4 (a)와 같은 2단 연산 증폭기에서 zero가 높은 주파수 영역에 위치하도록 설계되는 경우, 단위 주파수 이득 폭 (unit-gain bandwidth : ω_{unity})에서의 위상 여유 (phase margin : Φ_{PM})는 식 (2)와 같이 근사될 수 있다.

$$\Phi_{PM} \cong 90^\circ - \tan^{-1}\left(\frac{\omega_{unity}}{\omega_{P2}}\right) \quad (2)$$

이때 non-dominant pole, ω_{P2}는 g_{m2}/C_L이고, ω_{unity}는 g_{m1}/C_C인 관계로부터 식 (2)는 다음의 식 (3)과 같이 정리할 수 있다[10].

$$\Phi_{PM} \cong 90^\circ - \tan^{-1}\left(\frac{g_{m1}}{g_{m2}} \cdot \frac{C_L}{C_C}\right) \quad (3)$$

식 (3)에서 g_{m1} 및 g_{m2}는 2단 증폭기의 첫번째 및 두번째 이득단의 트랜스컨덕턴스를 나타내며, C_C는 주파수 보상을 위한 보상 캐패시터 (compensation capacitor), C_L은 출력단의 부하 캐패시터 (load capacitor)를 나타낸다. 출력 신호의 settling 시간을 줄이기 위한 스위칭 방법으로는 일단 전류를 차단한 후, 다시 전류를 인가해줄 때, 두번째 증폭기에 먼저 전류를 공급하여 g_{m2}를 키워주고 이에 종속적으로 g_{m1}을 키워줌으로써, 두 증폭기는 정상 상태 (steady state)에 이르기까지 상대적으로 $g_{m2} \gg g_{m1}$의 관계를 유지하도록 하는 것이 중요하다. 따라서, 상대적으로 ω_{P2}와 ω_{unity}가 멀어지는 효과를 봄으로써, 실제 계산된 정상 상태의 위상 여유 보다 바이어스 전류가 조정되는 동안 향상된 위상 여유를 얻게 되고, 동작 모드의 변화시 생기는 출력 전압의 오버슈트를 방지할 수 있다.

Ⅲ. 10 비트 algorithmic A/D 변환기에의 응용예

그림 5는 설계된 완전 차동 (fully-differential) 구조를 가진 이단 증폭기를 사용하는 MDAC의 전체 회로도이다. 스위치 M3과 M11은 전력 최소화 기법 응용시 샘플링 기간 동안 바이어스 공급이 차단된 각 증폭기 AMP1 및 AMP2의 출력단을 공통 전위로 고정시켜, 증폭 동작으로 전환시 출력 전압이 빠르게 settling 하도록 작용한다. 그러나, 샘플링 동작에서 잔류 전압 증폭 동작으로 전환할 때, 스위치 M11이 턴-오프 (turn-off)되면 virtual ground가 순간적으로 흔들리고, 이로인해 출력단에 글리치 (glitch)가 생긴다. 특히, 음성 대역폭 (audio-band) 응용에서 증폭

기의 대역폭 (bandwidth)이 낮을 경우 이러한 글리치가 오래 지속되고, 또한 불규칙적인 성향을 보이므로 전반적으로 settling 시간이 불규칙적이게 된다. 점선 안의 회로는 이러한 현상을 줄이기 위해 추가된 디글리치 (deglitch) 회로로 스위치 M11을 지연된 시간후에 턴-오프 시킴으로써, 글리치가 settling 시간에 미치는 영향을 감소시켜, 전력 최소화 기법 응용시 성능 저하를 방지한다.

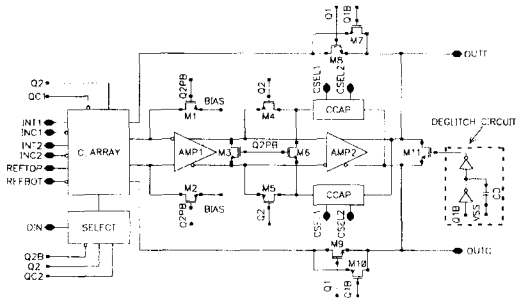


그림 5. 제안된 MDAC의 전체 회로도
Fig. 5. Top schematic of the proposed MDAC.

입력 부분의 C_ARRAY는 입력을 선택하기 위한 MOS 스위치 단과 단위 캐패시터 열을 포함하고 있다. 입력 선택은 컨트롤 신호 QC1에 의해 이루어지며, 클럭 한 주기마다 2 비트의 디지탈 출력 코드가 얻어지는 algorithmic 구조이므로 캐패시터 열은 총 6개의 단위 캐패시터로 구성된다. SELECT는 C_ARRAY에서의 스위칭을 컨트롤하기 위한 디지탈 회로이다. 이 회로는 입력 샘플링 모드 및 순환 모드에서 잔류 전압 증폭 동작시 각 캐패시터를 선택적으로 $+V_{ref}$ 또는 $-V_{ref}$로 연결한다. CCAP는 제안된 보상 회로를 나타낸다.

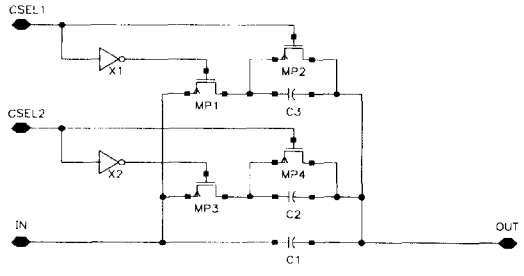


그림 6. 보상 회로의 구조
Fig. 6. Schematic of the tuning circuit.

그림 6은 제안된 보상 회로 (CCAP)의 실제 구조로 세 개의 캐패시터 C1, C2 및 C3을 사용하였으며, C2와 C3은 실제 공정상에서의 변화폭을 감안하여, 각각 제일 큰 캐패시터인 C1의 약 20 %의 크기로 설계하였다. MP1 (MP3)과 MP2 (MP4)는 서로 반대로 동작하며, MP2와 MP4는 C3 및 C2가 사용되지 않을 때, 윗 부분과 아래 부분을 묶어줌으로써 전체 회로 동작에 영향을 거의 주지 않으면서 캐패시터가 연결되지 않은 상태로 남아있는 것을 방지한다. 보상 회로의 디지탈 입력 CSEL1 및 CSEL2에 따른 보상 캐패시터의 크기 변화는 표 1과 같다.

표 1. CSEL1 및 CSEL2에 따른 보상 캐패시터
Table 1. Compensation capacitor by CSEL1 and CSEL2.

CSEL1	CSEL2	Cc (보상 개패시터)
Low	Low	C_1
Low	High	$C_1 + C_2$
High	Low	$C_1 + C_3$
High	High	$C_1 + C_2 + C_3$

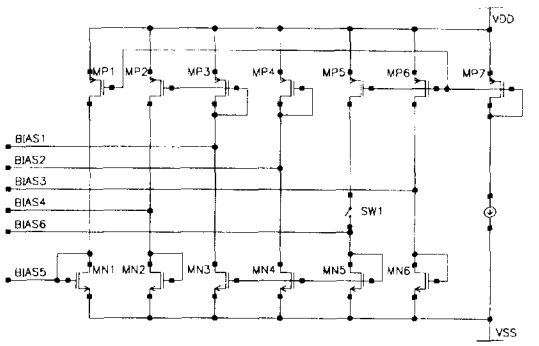


그림 7. 전력 최소화를 위한 바이어스 회로 설계 기법
Fig. 7. Biasing technique for power reduction.

그림 7은 3 V 수준의 저전원 전압에서 증폭기의 정상적인 동작을 위해 설계된 증폭기 바이어스 회로이다. 아날로그 영역에서의 전력 최소화 회로 설계 기법의 일환으로 스위치 SW1의 기능을 추가하였다. MDAC의 샘플링 동작에서는 스위치를 열어 바이어스 전류의 흐름을 차단함으로써 증폭기내 바이어스 공급을 막아 AMP2와 AMP1을 동시에 꺼준다. 반면에, 잔류 전압 증폭 동작에서는 스위치 SW1을 닫아줌으

로써 증폭기 AMP2와 AMP1이 순차적으로 정상 상태로 동작하게 한다. BIAS1, BIAS2, BIAS3, BIAS5는 두 증폭기의 출력단에 공통으로 사용되고, BIAS4는 첫째단 증폭기의 입력단에, BIAS6은 두째단 증폭기의 입력단에 각각 독립적으로 사용된다. 이는 바이어스의 순차적인 공급을 위한 것으로 SW1이 턴-온 되면, BIAS6-BIAS1-BIAS4 순으로 바이어스 공급이 재개된다. 그리고, BIAS1 및 BIAS2는 동시에 켜지며, BIAS3과 BIAS5는 바이어스가 공급되지 않고 있는 상태에서 전력 소모와는 관계 없으므로, MDAC의 출력 공통 모드를 위해 항상 일정하게 공급한다.

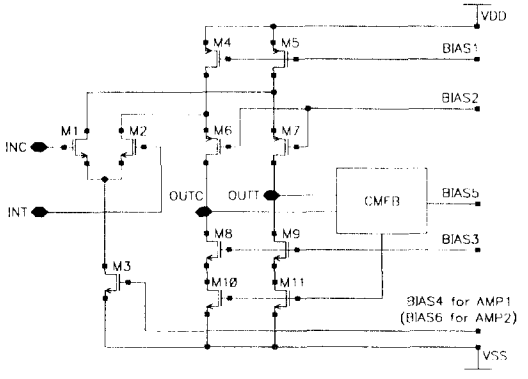


그림 8. AMP1 및 AMP2의 회로도
Fig. 8. Schematic of AMP1 and AMP2.

그림 8은 사용된 연산 증폭기 AMP1 및 AMP2의 회로도이다. 모두 완전 차동 증폭기로 전형적인 folded cascode 구조를 가지며, 전력 최소화 기법을 적용하기 위해 입력단 바이어스로 첫째단 증폭기에는 BIAS4를 두째단 증폭기에는 BIAS6을 각각 분리하여 공급한다. 출력단의 CMFB 블럭은 공통 모드 궤환(common-mode feedback) 회로로 동작 전력 소모를 줄이기 위해 동적 (dynamic) 구조를 사용하였으며, 출력 전압의 공통 모드를 잡아준다.

Ⅳ. 모의 실험 및 측정 결과

그림 9는 전력 최소화 기법 응용시 AMP2의 출력단에 디글리치 회로를 이용하지 않는 경우와 이용하는 경우를 비교한 출력 파형으로, 단일 전원 전압이 3.3 V인 경우에 MDAC 출력단에 차동 신호 -1 V가 나타날 경우의 모의 실험 결과이다. 디글리치 회로를 이용하지 않는 경우 잔류 전압 증폭으로의 동작 변환시 출력 파형의 시작 부분에 출력 신호와는 반대의 극성으로 나타나는 글리치로 인해 출력 신호가 원하는 신호 수준까지 settling하는데 걸리는 시간이 지연된다. 반면에, 디글리치 회로를 이용한 경우 전반적으로 글리치로 인한 영향이 감소하였고, 이로인해 출력 파형이 일정하게 settling함을 볼 수 있다.

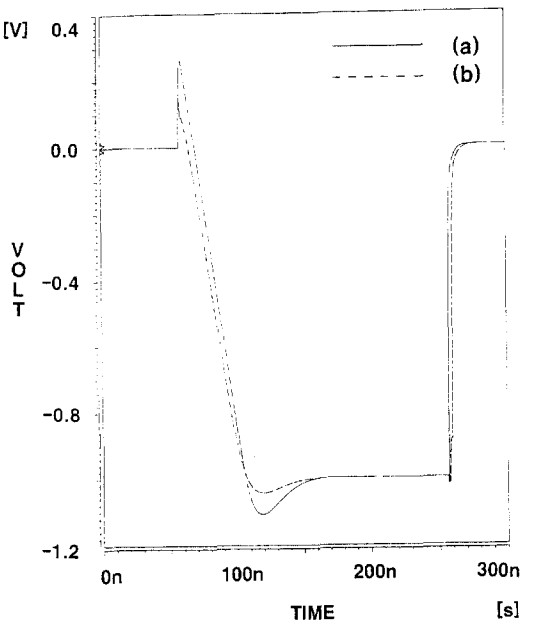


그림 9. MDAC 출력단 신호 파형: (a) 디글리치 회로를 이용하지 않는 경우 및 (b) 이용하는 경우
Fig. 9. MDAC output waveforms: (a) without and (b) with the deglitch circuit.

그림 10은 증폭기에 바이어스를 항상 공급하는 기존의 전형적인 경우와 제안된 전력 최소화 기법을 응용한 경우의 MDAC 출력단 신호 파형을 비교한 것으로, 3.3 V 단일 전원 전압에서 MDAC 차동 신호 출력 +1 V를 얻을 경우의 모의 실험 결과이다. 바이어스 조건이 동일할 경우 제안된 전력 최소화 기법을 응용하여도 기존의 전형적인 방식과 비교하여 증폭기의 성능 저하와 같은 문제가 없음을 확인할 수 있다.

그림 11은 전원 전압을 가변하며 기존의 전형적인 경우와 제안된 전력 최소화 기법을 응용한 경우를 각각 모의 실험한 결과로, 바이어스 회로를 포함한 MDAC의 전체 전력 소모 비교이다. 각 전력 소모량

은 가장 큰 전력이 사용되는 5 V 전원 전압에서 전형적인 경우의 전력 소모를 기준인 1로 정규화 (normalization)하여 도시하였으며, 전력 최소화 기법을 응용한 경우 3.3 V에서 5 V 사이의 모든 전원 전압에서 42%로 일정한 전력 감소 효과를 얻을 수 있었다.

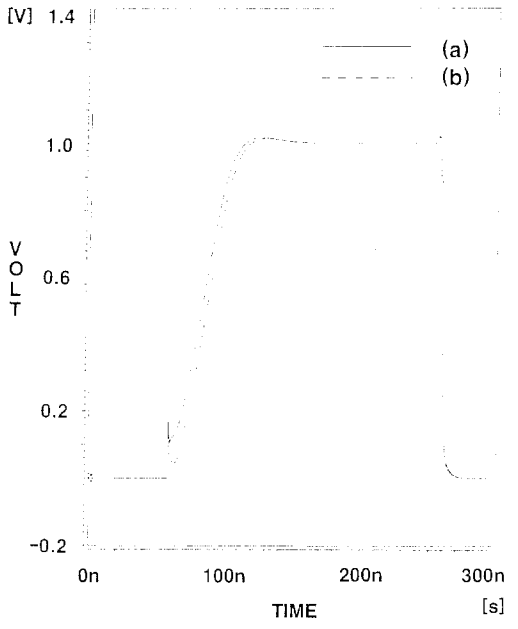


그림 10. MDAC 출력단 신호 파형: (a) 전력 최소화 기법을 응용하지 않는 경우 및 (b) 응용한 경우
Fig. 10. MDAC output waveforms: (a) without and (b) with power reduction.

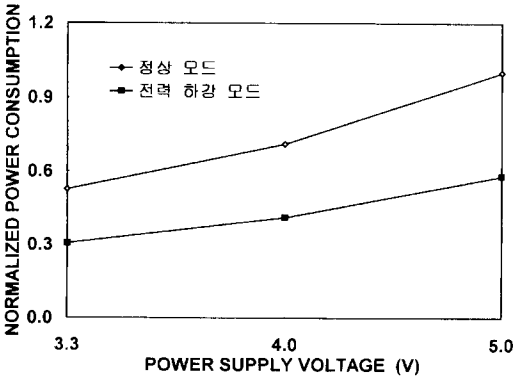


그림 11. 정규화된 전력 소모와 전원 전압의 관계
Fig. 11. Normalized MDAC power consumption vs. power supply voltage.

5 V 전원 전압에서 전력 최소화 기법을 적용한 경우 증폭기에 바이어스를 항상 공급하는 기존의 구조를 3.3 V의 낮은 전원 전압에서 동작시킨 경우와 유사한 전력 소모를 보인다.

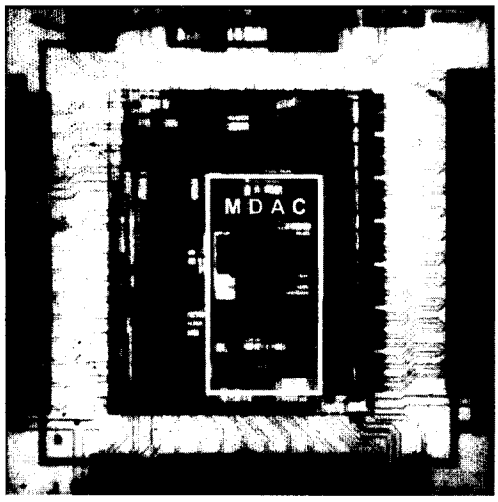


그림 12. 칩 사진
Fig. 12. Chip photograph.

제안된 MDAC은 그 효용성 및 성능을 검증하기 위해 10 비트 200kHz algorithmic ADC에 응용되었으며, 전체 회로는 0.6 um n-well single-poly double-metal CMOS 공정을 사용하여 제작되었다. 그림 12는 응용된 ADC의 전체 칩 사진으로 흰 테두리 안은 제안된 저전력 MDAC을 나타낸다. 표 2는 ADC의 전체 전력 소모로 증폭기에 항상 바이어스를 공급하는 기존의 전형적인 MDAC을 응용한 경우와 제안된 저전력 MDAC을 응용한 경우의 측정 결과이다. 전력 최소화 기법을 응용한 경우 기존의 구조에 비해 3.3 V 전원 전압에서는 11 mW에서 7 mW로, 5 V 전원 전압에서는 19 mW에서 14 mW로 소모 전력이 줄어듦을 확인할 수 있으며, 최대 36 %의 전력 감소 효과를 얻을 수 있었다. 그림 13은 제안된 MDAC을 응용한 10 비트 ADC의 DNL (differential nonlinearity) 측정 결과로 ±0.8 LSB가 얻어졌으며, 전력 최소화 기법 응용시에도 전체 시스템 성능에는 영향을 끼치지 않음을 확인할 수 있었다.

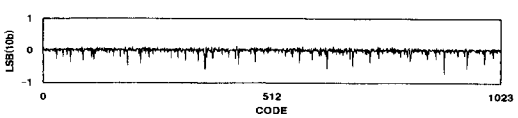


그림 13. DNL 측정 결과
Fig. 13. DNL plot.

표 2. 측정된 ADC 전체 전력 소모
Table 2. Total power consumption of the experimental ADC.

전원 전압	기존의 MDAC 응용	제안된 MDAC 응용
3.3 V	11 mW	7 mW
5 V	19 mW	14 mW

V. 결 론

본 논문에서는 algorithmic A/D 변환기 구조를 위한 저전력 CMOS MDAC을 제안하였으며, 실제 10 비트 200kHz algorithmic A/D 변환기에 응용하여 집적 회로로 구현되었다. 제안된 MDAC 회로는 칩 제작 후 공정상의 문제로 인한 증폭기의 성능 변화도 간단한 보상 회로를 미리 회로에 내장함으로써 외부에서 쉽게 보상이 가능하며, 공정 개발 초기에 흔히 발생하는 재설계에 따른 시간 및 비용절감 효과를 얻을 수 있다. 또한, 제안된 회로는 아날로그 부분에서의 전력 최소화 기법을 적용하여 동작 속도 및 정확도에는 거의 영향을 주지 않으면서 응용된 ADC 전체 전력 소모를 36 % 감소 시킴을 측정 결과로부터 확인할 수 있었다. 이러한 기법들은 음성 신호 처리 영역 외에도 영상 신호 처리를 위한 파이프라인 (pipeline) 구조의 고속 A/D 변환기에 직접적인 응용이 가능하며, 이는 추후로 진행되어야 할 사항이다.

참 고 문 헌

[1] P. W. Li and M. J. Chin, "A ratio-independent algorithmic analog-to-digital conversion technique," *IEEE J. Solid-State Circuits*, pp. 828-836, Dec. 1984.

[2] A. M. Mallinson and P. Spitalny, "A 17b algorithmic ADC," *ISSCC Dig. Tech. Papers*, pp. 40-41, Feb. 1992.

[3] D. G. Nairn and C. A. T. Salama, "A ratio-independent algorithmic analog-to-digital converter combining current mode and dynamic techniques ," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 319-325, Mar. 1990.

[4] K. Nagaraj, "Efficient circuit configurations for algorithmic analog-to-digital converters," *IEEE Trans. Circuits Syst.*, vol. 40, no. 6, pp. 777-783, Dec. 1993.

[5] Akihiro Kitagawa, et al., "A 10b 3M sample/s CMOS cyclic ADC," *ISSCC Dig. Tech. Papers*, pp. 280-281, Feb. 1995.

[6] J. Candy and G. Temes, "Oversampling methods for A/D and D/A conversion," in Oversampling delta-sigma data converters. Theory, design and simulation, IEEE press, 1992.

[7] I. Fujimori, K. Koyama, D. Targer, F. Tam, and L. Longo, "A 5V single-chip delta-sigma audio A/D converter with 111dB dynamic-range," *Custom Integrated Circuits Conference*, pp. 415-418, May 1996.

[8] Gert Cauwenberghs, "A micropower CMOS algorithmic A/D/A converter," *IEEE Trans. Circuits Syst.*, vol. 42, pp. 913-919, Nov. 1995.

[9] B. S. Song, S. H. Lee, and M. F. Tompsett, "A 10-bit 15-MHz CMOS recycling two-step A/D converter," *IEEE J. Solid-State Circuits*, vol. 25, no. 6, pp. 1328-1338, Dec. 1990.

[10] P. R. Gray and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, John Wily & Sons, New York, 1993.

저 자 소 개

李 宰 煒(正會員) 第 33권 A編 第 8號 參照
현재 삼성전자 연구원

李 承 勳(正會員) 第 32권 A編 第 12號 參照
현재 서강대학교 전자공학과 부교수