# FPGA 기반 시스템에서의 열 감지 센서 구현 기법

김선규[1] 김용주[2] 김태환[3]
[1]한국과학영재학교
[2]KAIST, 과학영재교육연구원
[3]서울대학교, 전기컴퓨터공학부
[1]nyroc2@gmail.com
[2]kyjgifted@kaist.ac.kr, [3]tkim@ssl.snu.ac.kr

# Thermal Sensor Design Technique for FPGA Based Systems

Sun Gyu Kim[1] Yongju Kim[2] Taewhan Kim[3]
[1]Korea Science Academy
[2]Institute for Gifted Students, KAIST
[3]School of Electrical Engineering and Computer Science, Seoul National University

**요 약**

주어진 작은 크기의 칩 내부에 많은 기능 (예: 멀티미디어, 음성/영상 등)을 작동시키기 위해서는 고집적(high-integration)의 회로가 구현되게 된다. 이러한 고집적 회로는 작동할 때 상당한 양의 전력 소모를 유발하게 되어 결국 배터리 수명을 단축시키는 상황을 가지게 한다. 더욱 심각한 상황은 고 밀도의 칩 안에서의 많은 전력 소모는 열의 발생을 더욱 가속화 시키게 되며, 결국 칩 작동의 신뢰성(reliability)을 상당히 잃게 만든다. 본 연구에서는 칩의 작동에 따른 열 발생으로 유발되는 칩의 온도 상승을 감지하는 센서 회로 구현에 관한 것이다. FPGA 칩은 주 목적의 기능을 수행하는 회로들을 구현함과 동시에 추가적으로 열 감지 센서 회로를 구현할 자원을 FPGA가 제공을 해 주어야 하는데, 주목적의 회로 공간(즉, 자원) 사용으로 인해 열 센서 회로 구현 자원이 충분하지 않을 경우나 여러 지역에 사용 가능한 자원이 소규모로 흩어진 경우 등 센서 구현을 위한 자원 탐색 및 구현 가능성에 대해 점검하는 알고리즘이 필요하다. 본 연구는 이러한 알고리즘을 개발하여 그 효용성을 실험을 통해 보이고 있다. 제안한 알고리즘의 특징은 Branch-and-Bound에 기반을 두고 있으며, 알고리즘의 수행 시간 단축을 위한 효과적인 search tree pruning 기법을 제안하고 있다.

## 1. 서론

현대 생활에 있어서 전자기기의 공헌은 헤아릴 수 없이 많다. 그 발전 또한 비약적이었다. 특히 그 발전은, 전자기기의 능률을 높이는 방향으로 연구가 되어 왔다. 전자 칩은 진공관에서 시작되어서 현재의 작은 기판 형태로 크기를 줄이면서 효율을 높이는 형태로 발전해 왔다. 하지만 집적 밀도가 늘어남에 따라서 소모 전력의 밀도가 늘어나면서, 칩의 온도가 높아졌다. 또한 이에 따라서 칩의 수명이나 연산 지연 시간 등이 영향을 받았다. 열이 고성능 VLSI에 끼치는 악 영향이 점점 심각해지고 있으며, 이것을 해결하기 위한 많은 연구들이 최근 진행되고 있다 [1, 2].

온도가 높아졌을 때, 일어나는 가장 중요한 일은, 칩의 계산 정확도가 떨어지는 것이다. 10-비트 가산기(adder)의 경우, 온도가 높은 경우, 신호의 도달 시간이 늦춰져서 연산이 실패하는 경우가 생길 수 있다는 연구 결과가 나와 있다 [3]. 따라서 칩 내에서의 온도를 관리하는 것은 매우 중요하다. 온도가 높아지는 경우, 이것을 감지하여, 칩의 가동을 중단시키고 냉각시키는 것이 필요하다. 이를 위해서는 칩에 온도를 측정하는 sensor에 대한 연구의 중요성이 점점 중요하게 부각되고 있다.

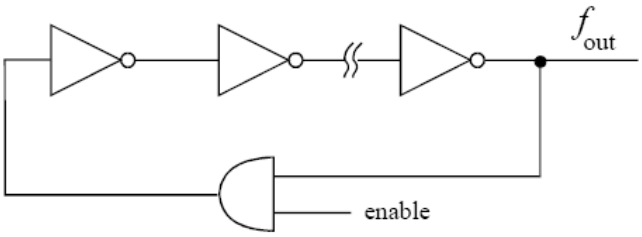
본 연구에서는, 현재 FPGA (field programmable gate array [현장 프로그래밍 가능한 게이트 어레이], 반도체 소자의 일종) 등에서 이용되는 Ring-oscillator sensor를 주어진 자원 제약과 지연시간 조건에서 설계하는 기법을 제안하고 있으며, 특히 sensor를 FPGA의 한정된 여분의 자원(논리 구현을 위한 회로)에 구현하기 위한 자원 탐색에 초점을 맞추고 있으며, 본 연구에서는 Branch-and-Bound 기법에 기반한 방법으로 back-tracking을 효율적으로 수

행하는 방법을 제시 한다.

## 2. FPGA 기반 설계의 열 감지 센서

### 2.1 Ring-oscillator 센서

온도를 감지하는 sensor는 여러 가지 종류가 있다. 그 중에서도 여기서 이용할 sensor 종류는 ring-oscillator sensor (그림 1참조) [4]로 회로 시스템의 설계 구현과 동시에 구현을 하여 FPGA에 넣을 수 있기 때문에 임베디드 시스템 응용 설계처럼 특정 application를 구현하는 시스템 설계에 쉽게 적용, 이용될 수 있다. 우선 ring-oscillator의 원리를 설명하자면 다음과 같다. 홀수 개의 inverter가 연결된 loop 에 전기를 통하면, 그 신호가 계속 공전 하게 된다. 각 inverter와 channel마다 전기 신호가 이동하는 데에 delay가 생긴다. 이 delay의 총합만큼의 시간이 흐르면




[그림 1] Ring-oscillator의 개요도[4] Inverter는 홀수 개가 연결 된다.

0의 신호가 1로 바뀐다. 그러면 delay 총합의 2배에 해당하는 시간이 이 ring oscillator의 주기가 된다.

이 구조가 온도를 측정할 수 있는 이유는, 이 delay가 온도가 커짐에 따라서 바뀌기 전기적 성질이 있기 때문이다. 0과 1 신호의 진동의 진동수(=1/(주기))가 온도가 높아짐에 따라 줄어들고, 이를 이용해서 추측한 온도가 상당히 정확하다는 연구 결과가 있고 실제로 응용되기도 하였다 [4, 5].

### 2.2 Sensor 설계에 대한 중요성

시스템 설계에서는 이러한 회로 작동에 따른 열 분산과 온도 상승을 감지하기 위해서 sensor를 장치해야 한다. FPGA 칩에는 Lookup Table (이하 LUT이라 부름)에 Inverter를 넣을 수 있고(즉,구현할 수 있고), 이들을 channel 내에 있는 연결선 자원을 통해서 연결할 수 있다. 그런데 sensor는 우선적으로 본래의 기능 작동을 하는 회로가 구성된 뒤에야 들어가게 된다. 즉 sensor 구현에 필요한 자원 제약 조건이 있다.

여러 다수의 ring-oscillator가 비슷한 진동수를 가지면, 한개의 processor가 시간차를 두고 모든 sensor 정보를 관리, 제어 할 수 있을 것이다. 즉, 거꾸로 현재 조건에서 원하는 진동수를 가진 ring-oscillator를 구성할 수 있는지 알 수 있다면, 이를 이용, processor를 효율적으로 이용할 수 있을 것이다. 예를 들자면, 4개 sensor를 무작위로 설정해서 processor가 4개 필요한 경우가 있을 때, 각각 sensor의 설치 위치를 고려해서 적합한 진동수 범위를 찾는다면, 이 모든 sensor를 한 processor에서 관리할 수도 있고, 그렇지 않더라도 2개 정도로 줄일 수 있는 경우가 있다. 요약하면, 각 sensor에서 생성되는 진동수가 (온도가 같을 경우) 거의 동일하도록 sensor를 설계하도록 한다.

### 2.3 문제 정의

홀수 개 LUT로 inverter를 구현할 수 있다고 가정한다. (본 연구에서는 한 개의 LUT에 한 개의 inverter를 구현한다는 가정을 한다. 이유는 여러 개의 inverter를 구현한다고 할 경우, LUT 내에서의 진동수를 맞추기 위한 지연 시간에 거의 기여를 못하기 때문이다.) 본 연구에서는, LUT 사이는 channel을 통해 잇는다. 이 문제의 경우는 간단하게 하기 위해서 총 n개의 LUT를 정점(node)으로 하고, channel은 이를 연결하는 edge로 한 그래프(graph) 모델을 사용하였다. 이 때 같은 LUT 쌍 사이에는 모서리가 없거나 하나뿐이라고 가정한다. Delay의 경우, inverter는 각각 $D_i$의 일정한 값을 가진다고 하고, LUT A와 B사이의 channel은 각각 고유한 delay인 E[A,B]를 갖고 있다고 가정한다.

그러면 문제는, *주어진 LUT와 channel을 가지고 원하는 진동수의 범위를 가지는 홀수 개 inverter가 이어진 cycle이 존재하는가 여부를 알아내는 것이다*. 위에서 밝힌 듯이, inverter와 channel의 delay 총합이 진동수와 반비례 관계이기 때문에, 이 문제는 즉 원하는 delay 총합의 홀수 길이 cycle이 존재하는지를 알아보는 것이다. 따라서 "원하는 delay 합 범위를 가진 cycle이 있는가?"를 판별하는 것이 문제이다.

### 2.4 제안한 알고리즘

본 연구에서 제안한 알고리즘은 Branch-and-Bound에 기초한 최적의 해 (즉, 조건을 만족하는 해가 있다면 반드시 찾게 되는)를 찾는 방법이다. 즉, 일단 back-tracking 방법을 사용하여, 구하고자 하는 해가 존재 한다면 반드시 찾을 할 수는 있다. 하지만 back-tracking의 방법은 해당하는 답이 나올 때까지 수형도(search tree)를 계속 그려보는 데에 해당한다. 그래서 비효율적인 경우가 많다. 그래서 이 경우, Branch and Bound가 필요한 것이다. 필요 없는

경우는 가지를 치는 게 (pruning) 필요한 것이다. Branch and Bound는 현재 탐색 상황에서 나올 가능성이 있는 delay의 상한과 하한 값을 계산한다. 이러한 상한을 Upper bound라고 하고, 하한을 Lower bound라고 하자. 이 때 원하는 delay의 값이 이러한 상한과 하한 값을 벗어나면, 가능성이 없기 때문에, 그 경우에 대해서는 탐색을 제외한다.

Upper bound와 Lower bound를 구하는 방법을 설명한다. 이 값들을 각각 $D_U$, $D_L$이라고 하자. 그리고 back tracking 중 탐색과정은 현재 $A_1$ - ... - $A_X$ - $B_I$이라고 하고, 탐색과정 중에 확정된 delay의 합을 $D_S$라고 할 것이다.

Upper bound - 가능한 최대의 Delay

조건에 맞는 최대 delay 합을 가진 cycle이 $A_1$ - ... - $A_X$ - $B_I$ - ... - $B_Y$ - $B_{Y+1}(=A_1)$가 연결되어 있다고 하고, 사용되지 않은 정점들을 $C_1$ ~ $C_Z$라고 하고, $A=\{A_1, ..., A_X\}$, $B=\{B_I, .., B_Y\}$, $C=\{C_I, .., C_Z\}$로 각각 집합으로 묶는다. 이 탐색 상황에서 진행될 경우 일반적인 delay 합을 D라고 하고, 정점 A에 연결된 channel들의 delay 중 최대값을 $E_{max}[B_i]$라고 하고, $R_i \subset R(=B \cup C)$의 원소라 하면

$D = D_S + (Y-1)D_I + \sum E[B_i,B_{i+1}]$ $(i = 1 .. Y)$

$\leq D_S + (n-1-X)D_I + \sum E_{max}[B_i]$

$\leq D_S + (n-1-X)D_I + \sum E_{max}[R_i] = D_U$ : Upper bound 값으로 정의

와 같이 Upper bound값, 즉 가능한 상한 delay 합인 $D_U$를 알 수 있다. 여기서 중간 과정의 값을 이용하지 않은 이유는, 그 안에 들어간 집합 B의 경우, 탐색 과정 중에는 정확히 알 수 없기 때문이다. $E_{max}[B_i]$ 값은 $O(n^2)$만에 미리 모두 계산해 놓을 수 있다.

Lower bound 계산

현재 탐색 과정이 $A_1$ - ... - $A_X$ - $B_I$ 이다. 이 때, cycle을 존재한다고 하면, 현재 있는 것을 제외하면 $A_1$과 $B_1$을 잇는 경로가 남게 된다. 이 경로에서 $A_1$과 $B_1$에 들어가는 inverter를 제외한 delay를 $D_P$라고 하자. 임의의 정점 C와 정점 D를 이을 때 가능한 최소의 delay (단 C와 D에 들어가는 inverter는 계산하지 않는다.)를 $D_{min}[C,D]$라고 하면 이 값은 항상 $D_P$보다 작거나 같다는 것을 보장할 수 있다. 그러면 다음과 같은 Lower Bound 값을 정할 수 있다.

$D = D_S + D_P \geq D_S + D_{min}[B_1,A_1] = D_L$ : Lower bound #1

이전의 경우, sensor cycle의 inverter가 홀수라는 것을 고려하면, 더 줄일 수도 있을 것이다. $D_{min}[i,C,D]$을 정점 C와 정점 D를 있는 최소 delay (이번에도 C와 D의 inverter는 제외)하되 조건이 하나 더 붙는다. 연결하는데 사용되는 edge 개수가 i가 0이면 짝수, i가 1이면 홀수 개라는 제한이 붙는다. $D_{min}[C,D]$가 $D_{min}[0,C,D]$와 $D_{min}[1,C,D]$ 중에서 더 작은 값이다. 그래서 더 큰 bound값을 얻을 가능성이 있다. Cycle을 이루는 정점의 수가 (X+Y)개 (각각 위의 집합 A와 B의 원소의 수이다.) 이고, 이는 홀수 개여야 한다. 현재까지 (X+1)개의 정점을 고려했다. 따라서 남은 경로의 edge 개수는 Y개이다. X+Y가 홀수여야 하기 때문에, X가 짝수이면 남은 모서리가 홀수, X가 홀수이면 남은 모서리가 짝수 개가 되어야 한다. 이를 사용하면 새로운 Lower bound $D_L'$를 구할 수 있다.

$D = D_S + D_P \geq D_S + D_{min}[(X+1)\%2,B_1,A_1] = D_L$ : Lower bound #2

$D_{min}[C,D]$나 $D_{min}[i,C,D]$의 경우는 그래프에서의 최단경로 찾는 알고리즘 적용함으로서 계산해 낼 수 있다.

## 3. 실험 결과

제안된 기법의 소스 코드는 Microsoft Visual C++ 6.0 언어로 작성되고, Microsoft Windows에서 구동되었다. 개발된 알고리즘은 다음과 같다.

Algorithm-1 : Branch and Bound with Upper Bound + Lower Bound #1

Algorithm-2 : Branch and Bound with Upper Bound + Lower Bound #2

Algorithm-3 : No Branch and Bound, Normal Back-Tracking

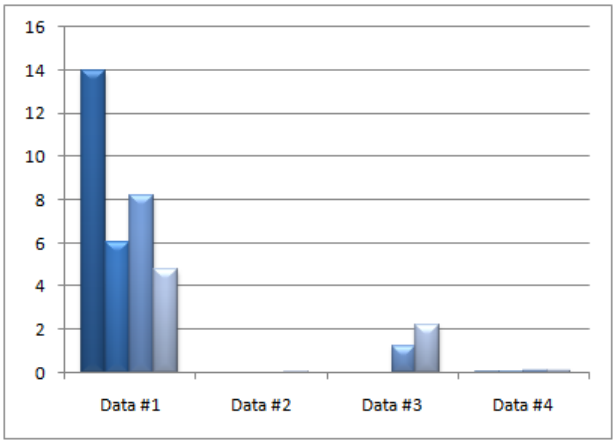
각각 알고리즘에서 효율성의 척도로서 탐색하는 과정에서 탐색하는 경우의 수를 측정하였으며 (표 1 참조), 알고리즘 1과 2의 차이, 그리고 2와 3의 차이를 각각 %로 나타내었다. 즉, (#1-#2)/#1, (#3-#2)/#2 의 값을 구하여 표 2에 비교하였다. 실험에 사용된 입력 데이터는 크기를 골고루 한, 다양한 방법으로 랜덤 생성하여 사용하였다.

표 1. Algorithm-1, 2, 3에서의 탐색 경우의 수

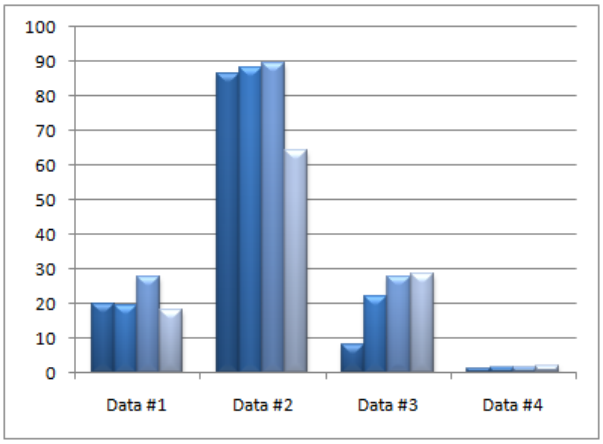
| # | $D_{min}$ | $D_{max}$ | Algo-1 | Algo-2 | Algo-3 |
|---|---|---|---|---|---|
| 1 | 1000 | 1100 | 211812 | 185861 | 232463 |
| | 1000 | 1050 | 197028 | 185861 | 230914 |
| | 1000 | 1010 | 178669 | 165112 | 228157 |
| | 1000 | 1004 | 366881 | 350108 | 428310 |
| 2 | 3900 | 4000 | 174675 | 174675 | 1281497 |
| | 3950 | 4000 | 168315 | 168312 | 1427301 |
| | 3990 | 4000 | 153721 | 153708 | 1489547 |
| | 3995 | 4000 | 22793131 | 22786874 | 63407715 |
| 3 | 4000 | 4050 | 56 | 56 | 61 |
| | 4000 | 4020 | 153 | 153 | 196 |
| | 4000 | 4010 | 327 | 323 | 448 |
| | 4000 | 4005 | 322 | 315 | 441 |
| 4 | 6000 | 6100 | 11151 | 11148 | 11304 |
| | 6000 | 6050 | 11146 | 11140 | 11304 |
| | 6000 | 6020 | 11444 | 11433 | 11603 |
| | 6000 | 6010 | 14595 | 14581 | 14872 |

표 2. Algorithm들 사이의 차이를 %로 나타낸 표

| # | $D_{min}$ | $D_{max}$ | Dif%(#1-#2) | Dif%(#2-#3) |
|---|---|---|---|---|
| 1 | 1000 | 1100 | 13.962585 | 20.0470613 |
| | 1000 | 1050 | 6.0082535 | 19.5107269 |
| | 1000 | 1010 | 8.2107903 | 27.6322883 |
| | 1000 | 1004 | 4.7908074 | 18.2582709 |
| 2 | 3900 | 4000 | 0 | 86.369457 |
| | 3950 | 4000 | 0.0017824 | 88.2076731 |
| | 3990 | 4000 | 0.0084576 | 89.6808896 |
| | 3995 | 4000 | 0.0274588 | 64.0629315 |
| 3 | 4000 | 4050 | 0 | 8.19672131 |
| | 4000 | 4020 | 0 | 21.9387755 |
| | 4000 | 4010 | 1.2383901 | 27.9017857 |
| | 4000 | 4005 | 2.2222222 | 28.5714286 |
| 4 | 6000 | 6100 | 0.0269107 | 1.38004246 |
| | 6000 | 6050 | 0.05386 | 1.45081387 |
| | 6000 | 6020 | 0.0962127 | 1.46513833 |
| | 6000 | 6010 | 0.0960154 | 1.95669715 |




[그림 2] Algorithm 1과 2사이 감소 비교




[그림 3] Algorithm 2와 3사이의 감소 비교

그림 3에서 보듯이, 알고리즘 2와 3을 비교했을 때의 차이는 엄청났다. data-4 의 경우, 차이가 2%내였지만, 나머지 1~3번 data의 경우 거의 20% 이상의 감소를 보였다. data-4 의 경우, 특수한 경우일 수 있기 때문에, 추가적인 시뮬레이션이 필요할 것이다.

Algorithm-1과 algorithm-2의 경우는 Lower-bound 값에 조금 차이가 있었다. 하지만 #1의 경우도 최대 14%의 차이 나타냈고, 2번 data부터는 대부분이 감소율이 1%에도 미치지 못하는 등, 크게 효율의 차이가 없었다.

## 5. 결론 및 논의

본 연구에서는 ring-oscillator 기반 sensor 구현을 찾기 위한 Branch-and-Bound 알고리즘을 제시하였다. 특히 tight한 upper, lower bound 값을 찾는 데에 그 중점을 두었다. 또한 실험을 통해 일반적인 back tracking과 비교했을 때, 대부분의 경우 그 효율성이 크다는 것을 알 수 있었다.

추후로, 이번 시뮬레이션에서 나온 특수한 경우들에 대한 추가 실험을 수행하고, 탐색 방법으로 breadth-first search, 혹은 best-first search 등의 다양한 탐색 방법을 이용해서 비교하는 등의 연구를 진행해서, 더 효율적인 방법을 찾을 수 있을 것이다. 또한 이 연구 결과를 이용, ring-oscillator sensor를 효율적으로 설계하는 데에 상당히 유용하게 사용될 수 있을 것으로 전망된다.

**감사의 글**: 본 논문은 한국과학재단의 RE 영재 프로그램과 지식경제부가 지원하는 국가 반도체 연구 개발 사업인 “시스템 집적 반도체 기반 기술 개발 사업(시스템 IC 2010)”을 통해 개발된 결과이며, 또한, “서울시 산학연 협력사업”으로 수행한 것이며, 2007년도 정부의 재원으로 한국과학재단의 지원 (과제번호:

R01-2007-000-20891-0)을 받아 수행된 연구임.

## 4. 참고 문헌

[1] K. Banerjee, A. Mehrotra, A. Sangiovanni-Vincentelli, and C. Hu, "On thermal effects in deep sub-micron VLSI interconnects," *Proc. ACM/IEEE Design Automation Conference,* pp. 885-891, 1999.

[2] K. Banerjee, M. Pedram, and A. H. Ajami, "Analysis and optimization of thermal issues in high-performance VLSI," *Proc. ACM/SIGDA International Symposium on Physical Design*, pp. 230-237, 2001.

[3] Y. K. Cheng, P. Raha, C. C. Teng, E. Rosenbaum, and S. M. Kang, "Illiads- t: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of cmos VLSI chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, pp. 668-681, August 1998.

[4] E. I. Boemo, and S. López-Buedo, "Thermal monitoring on FPGAs using ring-oscillators", *Proc. International Conference on Field-programmable Systems,* pp. 69-78, 1997.

[5] S. López-Buedo, J. Garrido, and E. I. Boemo, “Dynamically Inserting, Operating, and Eliminating Thermal Sensors of FPGA-Based Systems", *IEEE Transactions on Components and Packaging Technologies*, Vol. 25, No. 4, pp. 561-566, December 2002.