論文2003-40CI-6-4

십진수 계산을 위한 3초과 부호 가감산기 설계

(An Excess-3 Code Adder/Subtracter Design Decimal Computation)

崔 鐘 化 *, 韓 善 景 *, 劉 泳 甲 *

(Jong-Hwa Choi, Seon-Kyoung Han, and Young-gap You)

요 약

인간 친화적인 10진 계산을 위한 3초과 십진 가·감산기 회로를 제안한다. 십진 회로를 이용한 계산 시속도 문제는 carry lookahead (CLA) 회로를 이용하여 해결할 수 있다. 제안하는 3초과 십진수 가산기 설계에서는 CLA와 함께 보정회로 및 변환회로를 개선함으로서 지연시간을 줄일 수 있다. 3초과 코드를 사용함으로서 감산과정에서 가산기만을 사용하여 계산을 할 수 있다. 이 3초과 십진 가·감산회로는 기존의 설계에 비하여 상당한 속도 개선효과를 얻게 해준다.

Abstract

An excess-3 code adder/subtracter circuit is proposed for human friendly decimal computation. Carry lookahead (CLA) circuitry can be used to enhance decimal computation speed. The proposed excess-3 adder/subtracter employs improved CLA and compensation circuitry recoding computation delay. The circuitry used for addition is used for subtraction without further modification. Substantial speed improvement is obtained compared to conventional designs.

Keywords : carry lookahead, excess-3 code, adder, subtracter

Ⅰ. 서 론

대부분의 사람들은 십진 연산을 사용하여 왔기 때문에 십진수로 바로 계산하는 것이 자연스럽다. 십진수 계산은 두 가지의 장점으로 인하여 실용화의 과정이 꾸준하게 추구되고 있다[1, 2]. 첫째 십진수를 이진수로 변환하는 과정에서의 오차를 없앨 수 있다. 소수점 이하의 숫자에 대한 오차는 변환과정에서 피할 수 없는 것이다. 둘째, 십진수 계산은 인간 친화적인 계산 방식이다. 이진화 십진 회로를 사용하였을 때의 오차 감소 효과에 대한 분석과 동작속도 개선을 위한 회로 설계는 참고문헌[3]에 수록되어 있다.

십진 가산기의 속도개선은 두 가지 방향으로 추구되어 질 수 있다. 첫째, 올림수 전달 방식을 개선하기 위한 캐리 예견(carry lookahead: CLA) 회로의 도입이다. 두 번째는 숫자를 나타내는 부호 형식을 개선하는 것이다. 이 연구에서는 이 두 가지를 모두 도입하여 속도 개선을 추구하였다. 숫자 표현에 대한 부호 형식 중 3초과 코드를 사용할 경우 빠른 올림수의 전달이 계산 속도의 향상에 기여할 것이기 때문이다.

빠른 올림수 전달을 위한 CLA 회로의 도입은 효과적인 속도개선을 가능하도록 한다[4, 6]. 이 연구에서는 3

* 正會員, 忠北大學校 情報通信工學科
(Dept. of Information Communication Engineering, Chungbuk National University.)

接受日字:2003年7月3日, 수정완료일:2003年10月17日

초과 코드 가산기를 위하여 CLA 회로의 설계 문제를 다루고 있다. 이 결과는 십진수를 기반으로 하는 컴퓨터의 설계에 있어서 필요한 산술논리장치의 일부가 된다. 이진화 십진 회로에 비해 3초과 코드 회로에서는 캐리 생성(carry generation), 캐리 전파(carry propagation) 회로를 따로 구성할 필요가 없다. 그것을 토대로 3초과 코드와 고속회로를 사용해 십진 회로의 속도 개선을 추구 하였다. 또한 감산기의 경우 3초과 코드의 특징을 이용함으로서 별도의 감산기를 사용하지 않고, 가산기만으로 감산을 할 수 있으므로 빠른 감산을 할 수 있다.

본 논문은 3초과 코드를 이용해 십진 가·감산회로를 사용하기 위해 가·감산 회로에 입력되는 3초과 코드 특징을 알아보고 이 코드를 이용해 가·감산회로를 제안하였다. 이 논문의 2절에서는 빠른 3초과 가산기 회로설계에 대해서, 3절에서는 감산기 회로설계 및 시뮬레이션 결과, 4절에서는 지연시간 분석, 마지막으로 5절에서 결론을 맺는다.

Ⅱ. 십진 3초과 가산기

3초과 코드를 사용한 십진 가산기를 살펴 보기로 하자. 먼저 <그림 1>과 같이 3초과 가산기 입력으로 들어오는 3초과 코드는 비가중치 코드로서 4비트 이진수로 된 8-4-2-1 코드에 3을 더해 나타낸 코드이다[5]. 이진 표현관계를 살펴보면,

$$x_{bm}(ex_3) = x_{bm}(8421) + 3$$

여기서,

$$x_{bm}(8421) = \sum_{i=0}^{3} x_i(8421)2^i$$

$$x_{bm}(ex_3) = \sum_{i=0}^{3} x_i(ex_3)2^i$$

이다.

3초과 코드도 마찬가지로 이진 4비트로 표시할 수 있는 코드 조합에서 10개만을 사용한다$(0011\sim1100)_{bin}$. 나머지 값은 don't care 처리를 한다. 8-4-2-1 코드에서 3초과 코드 변환의 경우 카르노맵을 이용해 <표 1>에서와 같이 각 자릿수 별로 최소화 된다. 이제 3초과 덧셈의 경우 결과는 두 가지로 나타낼 수 있는데,

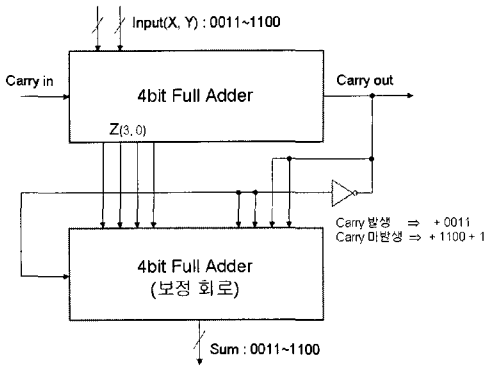

그림 1. 3초과 가산기
Fig. 1. Excess-3 adder.

표 1. 8-4-2-1부호와 3초과 부호간의 변환
Table 1. A conversion between the excess-3 code conversion and the 8-4-2-1 code.

Conversion (BCD → EX-3)		
ABCD (BCD)	WXYZ (EX-3)	
0000	0011	$W = A + B(C+D)$
0001	0100	$X = \overline{B}(C+D) + B\overline{C}\,\overline{D}$
0010	0101	$Y = \overline{C \oplus D}$
0011	0110	$Z = \overline{D}$
0100	0111	
0101	1000	
0110	1001	
0111	1010	
1000	1011	
1001	1100	

Conversion (EX-3 → BCD)		
WXYZ (EX-3)	ABCD (BCD)	
0011	0000	$A = W(X + YZ)$
0100	0001	$B = \overline{X}\overline{Y} + Y(XZ + W\overline{Z})$
0101	0010	$C = Y \oplus Z$
0110	0011	$D = \overline{Z}$
0111	0100	
1000	0101	
1001	0110	
1010	0111	
1011	1000	
1100	1001	

$x_{bm}(ex_3) + y_{bm}(ex_3) + C_m < 16$일 경우,

$Sum(ex_3) = x_{bm}(ex_3) + y_{bm}(ex_3) - 3 + C_{in}$

$C_{out} = 0$

$x_{bm}(ex_3) + y_{bm}(ex_3) + C_m \geq 16$일 경우는

$Sum(ex_3) = x_{bm}(ex_3) + y_{bm}(ex_3) - 3 + C_m - 10$

$C_{out} = 1$

로서 계산 과정을 살펴보면, 3초과 코드로 변환된 이진 4비트 두 입력을 받아 더해진 덧셈 결과는 각 입력이 3초과 형식이므로 6을 더한 결과 값이 나온다. 회로에서 $Sum_{ex3} \geq 16$ 일 경우, 캐리가 발생하고 캐리는 다음 자릿수 결과에 포함된다. 최종 3초과 합을 얻기 위해서는 보정회로가 필요하다. 캐리가 발생했을 경우와 그렇지 않을 경우 합의 결과가 틀리기 때문이다. 캐리가 발생시는 0011을 더해주는 보정회로가, 발생하지 않은 경우는 1100을 더해주는 보정회로가 필요하다.

빠른 연산을 하기 위해서는 고속연산회로가 필요하다. 고속연산회로를 사용하기 위해 3초과 가산기 특징을 살펴보면, 내부 계산에서 올림수 발생이 $Sum_{ex3} \geq 16$ 일 때 이므로 이진 가산시 올림수 발생 형식과 같다.

이러한 특징으로 이진 가산회로에서 빠른 가산을 하기 위해 사용되는 고속회로인 CLA 회로를 3초과 가산기에서도 이용할 수 있다. 3초과 가산기에서 CLA 회로를 사용하기 위해 특별한 G, P회로를 만들 필요가 없다. 일반적으로 사용되는 CLA의 생성, 전파 신호 형태와 같다. 캐리 생성(carry generation), 캐리 전파(carry propagation)는 3초과 가산기 회로에서 덧셈 시 사용되는 전가산기 게이트를 사용한다. <그림 2>는 CLA 회로가 들어간 3초과 가산기 회로이다.

$$Carry\ generation\ : G = X_{ex3} \cdot Y_{ex3}$$

$$Carry\ propagation\ : P = X_{ex3} \oplus Y_{ex3}$$

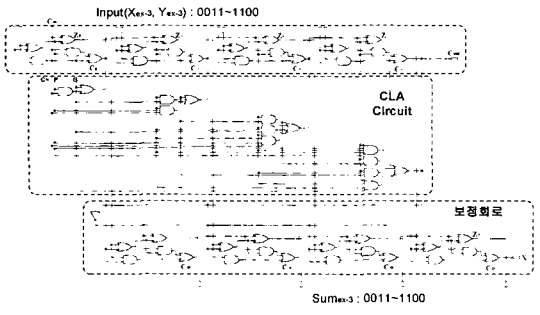

그림 2. Carry lookahead 회로를 사용한 3초과 가산기
Fig. 2. An Excess-3 adder by using the Carry lookahead.

3초과 가산기에서 CLA 회로를 사용해 속도개선이 어느 정도 이루어졌다고 하지만 문제가 되는 부분이 있다. 올림수 전달을 빨리 해도 보정회로에서 sum 출력을 위해 보정회로의 올림수 값이 리플 형태로 전파되므로, 그만큼 sum 출력이 늦어진다. 또한 3초과 값에서 십진 값으로 변환해주는 회로가 추가되기 때문에 출력 값은 지연 된다. 이러한 지연시간을 줄이기 위해서 보정회로부분에 CLA를 사용을 생각할 수도 있겠지만, 회로 사이즈 및 팬인(fan-in), 팬아웃(fan-out)을 고려했을 때 바람직하지 못할 것이다.

이렇게 많은 지연시간을 차지하는 보정회로 부분을 수정하여 지연시간을 줄일 수 있는 방법을 제안한다. 보정회로를 살펴보면 3초과 값인 두 입력의 더한 값을 보정해 3초과 결과값으로 나타내주는 회로인데, 여기서 계산 결과값을 보면 $Sum_{ex3} \geq 16$ 일 경우는 올림수를 발생하고 3초과 결과 값이 십진 형식의 숫자로 나타난다. 이것은 보정회로와 십진 변환회로를 사용하지 않아도 된다. 그러나 계산결과가 $Sum_{ex3} < 16$ 일 때 문제가 된다. 넘어온 계산 값에 대하여 십진 값으로 나타내줄 회로가 필요하다. 회로를 만들기 위해 카르노맵을 이용해 각 자릿수에 대해 최소화 했을 때 <표 2>와 같은 식을 얻을 수 있다.

결과식을 이용해 회로를 만들고, 올림수가 발생하는 경우와 발생하지 않는 두 가지 계산 결과에 대해서 선택해 줄 수 있는 MUX회로가 필요하다.

표 2. 변환 식
Table 2. Conversion expression.

Sum conversion		
$Z_3Z_2Z_1Z_0$(Z)	$S_3S_2S_1S_0$ (Sum)	
0110	0000	
0111	0001	$Sum(3) = Z_3 \cdot Z_2 \cdot Z_1$
1000	0010	$Sum(2) = Z_2 \oplus Z_1$
1001	0011	
1010	0100	$Sum(1) = \overline{Z_1}$
1011	0101	$Sum(0) = Z_0$
1100	0110	
1101	0111	
1110	1000	
1111	1001	

올림수가 발생하면 그대로 결과 값을 출력하고, 발생하지 않을 경우 보정회로를 거쳐 출력 값이 나가는 것이다. <그림 3>은 보정출력 회로를 나타낸다.

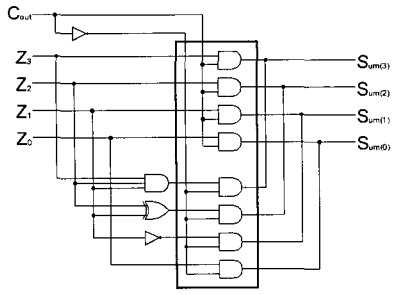

그림 3. 보정출력회로
Fig. 3. Output correction circuit.

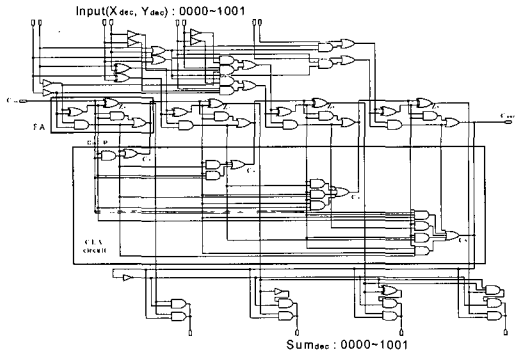

그림 4. 제안된 1디지트 3초과 CLA 가산기
Fig. 4. Proposed 1-digit excess-3 CLA adder.

보정회로와 십진 변환회로를 제거하고 보정출력회로로 바꾼 1디지트 3초과 CLA 가산기가 <그림 4>에 보여지고 있다. 기존의 3초과 가산기 회로에서 보정회로 및 변환회로를 제거하고 각 출력 비트에 나올 수 있는 값을 카르노맵으로 최소화 한 보정회로를 사용함으로서 기존에 걸리는 지연시간 및 게이트수를 줄일 수 있다.

Ⅲ. 십진 3초과 감산기

감산은 피감수에서 감수를 빼는 형식인데, 계산 과정에서는 감수의 보수를 취하고 이것을 피감수에 더하는 방식으로 계산된다. 대부분의 감산회로가 보수회로와 가산기를 이용하고 있다. 십진 회로에서 감산을 보수 형태로 바꾸어 가산을 하는 방식을 사용한다면 회로 측면에서 많은 이점이 있을 것이다.

입력 데이터의 크기에 관한 두 가지 경우에 대해 생각해 보기로 하자. 첫째는 감수가 피감수보다 작을 경우이고, 두 번째는 감수가 피감수보다 큰 경우이다. 감수가 피감수보다 작을 때 감산 결과는 보수를 취한 감수가 피감수보다 크기 때문에 더한 결과는 항상 올림수를 발생시키고, 9의 보수를 취하기 때문에 덧셈결과는 원하는 결과에서 1이 모자란 값이 나온다. 피감수가 감수보다 작을 경우 감산 결과는 보수를 취한 감수가 피감수보다 작기 때문에 더한 결과는 올림수를 발생시키지 않고 결과값은 원하는 결과를 9의 보수 형태로 나타난다.

이 두 가지 경우에 대해서 정확한 결과를 얻기 위해서는 보정이 필요하다. 보정은 첫 번째 경우 올림수는 빼고 결과 값에 1을 더해주면 되고, 두 번째 경우에는 다시 9의 보수를 취해주면 원하는 결과 값을 얻을 수 있다. 피감수와 감수의 크기에 따른 예를 아래에서 보여주고 있다.

피감수 > 감수	피감수 < 감수
7259 - 2481	2481 - 7259
7259 2481 → + 7518(9의 보수)	2481 7259 → + 2740(9의 보수)
① 4777 보정 ⇒ 올림수 제거 , +1 + 1	5221 보정 ⇒ 보수 취함(9), 음수 표시 5221 → 4778
4778	- 4778

위와 같이 감산방식을 사용할 때 입력 값을 이진화 십진 형식(BCD)으로 사용할 경우 감수에 대해 보수형태로 나타내기가 힘들다. 만약 $0100(4)_{BCD}$에 대해 보수를 취할 경우 $0101(5)_{BCD}$로 나타나야 한다. 그러나 이렇게 사용할 경우 보수 변환 방식이 복잡해지게 된다. 이러한 문제를 해결하기 위해 이진화 십진 코드가 아닌 3초과 코드를 이용하여 보수변환을 쉽게 할 수 있다. 3초과 코드 사용시 $0111(4)_{ex3}$에 대한 보수는 $1000(5)_{ex3}$으로 역수로 취해주면 되는 것이다. 3초과 코드를 이용한 감산 회로는 <그림 5>와 같다.

회로의 구성은 피감수부분 입력으로 3초과 값을 받고, 감수부분은 보수가 취해진(Inverting) 3초과 값이 입력된다. 두 입력 값은 제안된 3초과 CLA 가산기에

의해 더해진 결과 중 올림수가 발생하면 4비트 전가산기 회로로 들어가 보정된 결과가 출력되고, 올림수가 발생하지 않는 경우 에는 논리 반전(Inverter)을 취하면 결과 값이 출력난다.

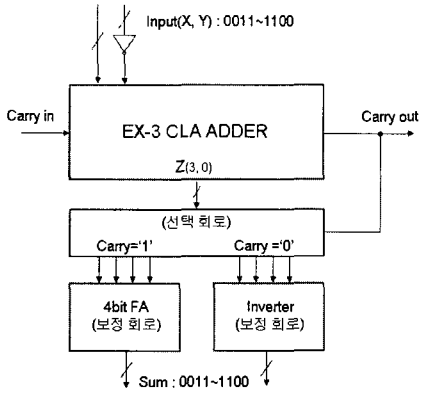

그림 5. 3초과 감산기
Fig. 5. Excess-3 subtracter.

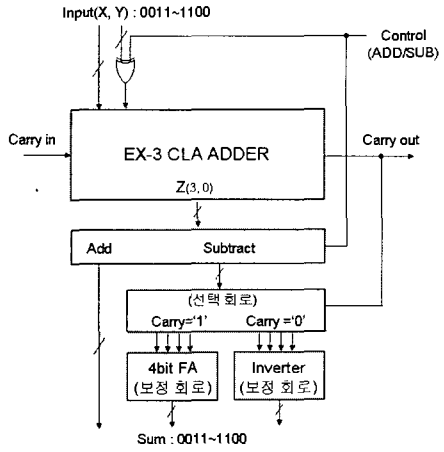

그림 6. 제안된 회로의 3초과 가·감산기
Fig. 6. Proposed excess-3 adder/subtracter.

감산기 회로를 살펴보면, 가산기와 반전(Inverter) 회로만으로 구성이 가능하므로 가·감산이 가능한 회로로 구현할 수 있다. 가산 신호와 감산 신호를 선택할 수 있는 제어 신호만 추가하면, <그림 6>에서와 같이 제안된 회로의 3초과 가·감산기 회로를 구성할 수 있다.

<그림 7>은 Altera사의 Quartus II simulation tool에서 검증한 가산 결과이다. 선택(sel) 신호는 가·감산 선택신호이며, 0일 경우는 가산, 1일 경우는 감산을 수행한다. 가·감산을 위한 자릿수는 16자리이고, 부호 비

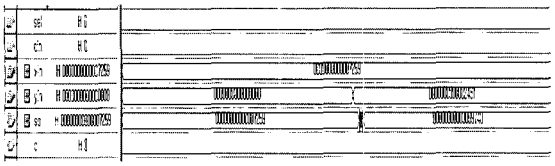

그림 7. 가산 결과
Fig. 7. Result in addition.

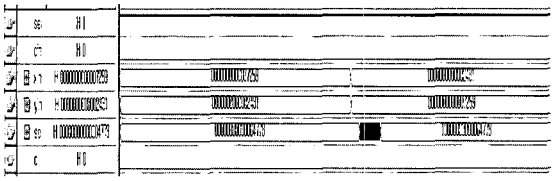

그림 8. 감산 결과
Fig. 8. Result in subtraction.

트를 위해 상위 1비트가 추가 되었다. 결과에서는 가산된 양수 출력 결과가 정확히 출력됨을 볼 수 있다. <그림 8>에서 감산결과를 보면 좌측의 경우 감수보다 피감수가 크므로 상위 부호 비트에는 0이 출력되고, 우측의 경우는 감수가 커서 음수 표현이 되므로 상위 부호 비트에는 음수 표현인 1이 출력된 것을 볼 수 있다.

Ⅳ. 지연시간 분석

이제 3초과 십진 가·감산회로의 지연시간을 분석해 보기로 하자. 이 회로에서는 가산이나 감산의 경우 가산기만을 사용하기 때문에 가산기의 지연시간에 대해 보기로 한다. 3초과 십진 가산기의 지연시간 성분을 결정하는 회로 요소는 입력된 값을 더해 주는 가산회로와 출력된 값을 3초과 코드에 맞게 보정해 주는 보정회로가 있다. 그리고 BCD를 3초과 값으로, 3초과 값을 BCD로 변환해 주는 회로가 필요하다. 여기서 게이트를 통과했을 때 발생하는 지연시간을 ΔT라고 할 때, sum을 결정하는 지연시간을 보면 가산회로와 보정회로는 4비트 전가산기이므로 지연시간은 $(2\cdot(N-1)+1)\triangle T$의 2배 만큼 지연시간을 가진다. 또한 sum에 추가되는 지연시간이 있는데 바로 코드 변환시 사용되는 게이트 통과 지연시간이다.

여기에 CLA 회로를 사용 할 경우 3초과 십진 가산기의 전체 지연시간 성분은 다음과 같다. τ_{becon}은 BCD 코드를 3초과 코드로 변환하는데 걸리는 시간, τ_{pg}는 각

비트 위치에서 G와 P 신호를 생성하는데 걸리는 시간, τ_{digit}은 4비트 그룹 내에서 내부 올림수를 결정하는데 걸리는 시간, τ_{sum}은 1디지트 3초과 sum을 결정하는데 걸리는 시간, τ_{ebcon}은 3초과 sum을 BCD sum으로 변환하는데 걸리는 시간이다.

$$\tau_{becon} = 3\Delta T,\ \tau_{pg} = 1\Delta T,\ \tau_{digit} = 2\Delta T,$$
$$\tau_{sum} = 9\Delta T,\ \tau_{ebcon} = 3\Delta T$$

과 같이 나타내며, 각 디지트에 나오는 총 sum 지연시간은 $\tau_{becon} + \tau_{pg} + \tau_{digit} + \tau_{sum} + \tau_{ebcon} = 18\Delta T$ 이다.

이것은 CLA회로를 한 단 사용한 것으로서 디지트가 증가할수록 $2\triangle T$ 만큼의 지연시간을 갖는다. N디지트의 경우 $(16+2N)\triangle T$ 만큼의 지연시간을 갖는다. CLA 회로를 여러 단으로 구성 했을 경우, 회로의 총 지연시간은

$$\tau_{total} = (18+4(log_{b}N-1))\triangle T$$

로 나타난다.

여기서 지연시간을 더 줄이기 위해 보정회로를 수정한 제안된 3초과 십진 가산기의 경우 기존의각 자릿수마다 보정회로에 사용되는 논리 게이트의 개수는 21개인데 비하여 12개로 줄임으로서 9개 만큼의 논리 게이트를 감소시키고, 지연시간은 $10\triangle T$ 만큼 줄일 수 있었다. <그림 9>에서는 자릿수 증가에 따른 방식별 지연시간을 비교하였다. 결과를 보면 CLA를 사용한 회로들은 리플 캐리 십진 가산기 보다 지연시간이 작음을 알 수 있다. 제안된 3초과 CLA(3-stage) 십진 가산기의 경우 3초과 CLA(3-stage) 십진 가산기 보다 각 디지트 지연시간이 $10\triangle T$만큼 작은 것을 볼 수 있다.

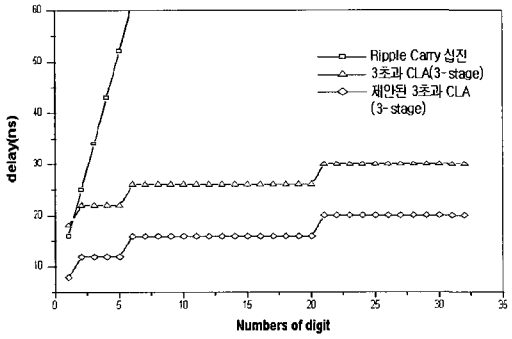

그림 9. 디지트 증가에 따른 방식별 지연시간 비교
Fig. 9. Comparison of delays due to digit increases.

V. 결 론

십진수 기반 컴퓨터 설계에 필요한 가산기의 설계에서 가산속도를 향상시키기 위한 3초과 부호의 적용회로를 제안하였다. 고속 연산회로인 CLA방식을 3초과 부호에 도입함으로서 빠른 계산이 가능하도록 하였다. 또한 변환회로나 보정회로에서의 지연시간을 줄일 수 있는 회로를 제안하여 각 출력은 기존의 3초과 가산기보다 회로의 논리 게이트 갯수를 9개 감소시키고, 10ΔT의 지연시간이 감소되는 결과를 얻었다. 감산기를 구현하는데 있어 3초과 코드의 보수 특성을 이용해 가산기만으로 감산기 회로를 구현하였다. 이 회로는 십진 부동 소수점 회로 등의 기반 회로로서 사용 되어지며, 십진 계산기의 구성에서 중요한 구성요소가 된다.

참 고 문 헌

[1] F. Busaba, C. Krygowski, E. Schwarz, W. Li, and S. Carlough, "IBM z900 decimal arithmetic unit," Proc. the 35th Asilomar Conf. on Sig-nals, Systems, and Computers, pp. 1335-1339, Nov. 2001.

[2] M. S. Schmookler and A. Weinberger, "High speed decimal addition," IEEE Trans. Computers, vol. C-20, no. 8, pp. 862-866, Aug. 1971.

[3] 최종화, 유영갑, "Carry lookahead를 이용한 BCD 가산기", 대한전자공학회 2002년도 컴퓨터/반도체 소사이어티 추계학술대회 논문 집, 제25권 제2호, 525-528쪽, 2002. 11

[4] B. Parhami, Computer Arithmetic Algorithms and Hardware Designs, Oxford University Press, 2000.

[5] R. J. Tocci and N. S. Widmer, Digital Systems Principles and Applications, Prentice-Hall International, 1998.

[6] M. J. Flynn, S. F. Oberman, and M. J. Flynn, Advanced Computer Arithmetic Design, John Wiley & Sons, 2001.

저 자 소 개

崔 鐘 化(正會員)

2001년 2월 : 충북대학교 반도체공학과(학사). 2001년~현재 : 충북대학교 정보통신 석사과정. <주관심분야 : VLSI설계, 연산회로 설계>

韓 善 景(正會員)

1991년 2월 : 충북대학교 정보통신공학과 공학사. 1993년 2월 : 충북대학교 정보통신공학과 공학석사. 1995년 3월~현재 : 충북대학교 정보통신공학과 박사과정. <주관심분야 : Computer arithmetic, Cryptographic system, ASIC 설계>

劉 泳 甲(正會員)

1975년 8월 : 서강대학교 전자공학과 공학사. 1975년~1979년 : 국방과학연구소 연구원. 1981년 8월 : Univ.of Michigan, Ann Arbor 전기전산학과 공학석사. 1986년 4월 : Univ.of Michigan, Ann Arbor 전기전산학과 공학박사. 1986년~1988년 : 금성반도체(주) 책임 연구원. 1993년~1994년 : 아리조나 대학교 객원교수. 1998년~2000년 : 오레곤 주립대학교 교환교수. 1988년~현재 : 충북대학교 정보통신공학과 교수. <주관심분야 : VLSI 설계 및 Test, 고속 인쇄 회로 설계, Cryptography>