논문 2007-44TC-2-9

고속 FPGA 구현에 적합한 효율적인 정수 나눗셈 알고리즘

(An Efficient Integer Division Algorithm for High Speed FPGA)

홍 승 모*, 김 종 훈*

(Seungmo Hong and chonghoon Kim)

요 약

본 논문에서는 메모리와 곱셈기가 내장된 고속 FPGA(Field Programmable Gate Array)에서 효율적으로 구현할 수 있는 정수 나눗셈 알고리즘을 제안하였다. 제안된 알고리즘은 메모리를 이용한 Look-up Table(LUT)과 곱셈기를 사용하여 반복 계산(Iteration)구조로 FPGA의 자원을 최소화할 수 있으며 반복연산 횟수가 일반적으로 알려진 뺄셈 또는 뺄셈-곱셈에 의한 나눗셈 알고리즘에 비해 매우 적어 Latency를 최소화 할 수 있다. Xilinx사의 Virtex-4 FPGA에 VHDL coding을 통해 Pipeline구조로 구현한 결과 17bit의 정수 나눗셈을 300MSPS(Mega Sample per Second)의 속도로 수행하였다. 또한 일반적으로 사용되고 있는 뺄셈 또는 뺄셈-곱셈 구조에 비해 FPGA의 소요자원인 Slices의 경우 1/6이하, 곱셈기-누산기 수는 1/4이하로 줄일 수 있었으며, 입출력 간의 지연 Latency를 1/3이하로 줄일 수 있어 다른 알고리즘에 비해 매우 효율적인 구조임을 확인하였다.

Abstract

This paper proposes an efficient integer division algorithm for high speed FPGAs' which support built-in RAMs' and multipliers. The integer division algorithm is iterative with RAM-based LUT and multipliers, which minimizes the usage of logic fabric and connection resources. Compared with some popular division algorithms such as division by subtraction or division by multiply-subtraction, the number of iteration is much smaller, so that very low latency can be achieved with pipelined implementations. We have implemented our algorithm in the Xilinx virtex-4 FPGA with VHDL coding and have achieved 300MSPS data rate in 17bit integer division. The algorithm used less than 1/6 of logic slices, 1/4 of the built-in multiply-accumulation units, and 1/3 of the latencies compared with other popular algorithms.

Keywords: integer division, LUT-based division, FPGA, iteration, latency

I. 서 론

최근 DSP 프로세서의 발달로 복잡한 알고리즘을 쉽게 구현할 수 있게 되어 다양한 분야에 활용되어지고 있다. 그러나 수십 MPS(Mega Sample per Second)이상의 속도를 가지는 데이터 처리에 대해서는 DSP 프로세서를 적용하지 못하고 있으며 이러한 환경에서는 FPGA를 사용하여 구현하고 있다. 한편 FPGA는 고속화와 고집적화가 병행되어 고속 데이터 처리에 적합하게 발전되어 있으며 최근에는 다수의 메모리와 곱셈기, 누산기가 내부에 탑재되어 DSP의 여러 알고리즘들을 손쉽게 구현할 수 있는 구조의 FPGA가 일반화 되어있다.

나눗셈은 음성 및 영상 처리, 그래픽 및 멀티미디어, 네트워크 등 응용분야 등에 중요한 기능으로 사용되어지고 있다. 일반적으로 나눗셈 알고리즘은 다음과 같이 크게 두 가지로 나눌 수 있다. 첫 번째 알고리즘은 덧셈/뺄셈과 쉬프트 동작을 이용하여 나눗셈을 수행하는 알고리즘으로 여기에는 복원(restoring)알고리즘, 비복원(non-restoring) 알고리즘, SRT 알고리즘이 속한다. 두 번째 알고리즘은 곱셈을 이용하여 나눗셈을 수행하는 방식으로, 여기에는 수렴 알고리즘과 역수 알고리즘으로 속한다. 일반적으로 성능이 우수하여 현재 널리 쓰이고 있는 SRT 알고리즘은 Redundancy를 두어 radix를 증가하여 계산할 때 발생하는 오차를 수정하여 성능을 향상시키는 방식이나 Iteration 횟수가 많아 고속보

* 정회원, 숭실대학교 정보통신전자공학부
(School of Electronic Engineering, Soongsil Univ.)
접수일자: 2006년11월1일, 수정완료일: 2007년2월14일

다는 정밀도가 중요한 요소인 범용 프로세서에서 많이 쓰인다.[1]-[2][6]

본 논문에서는 정밀도 보다는 고속 신호처리를 목적으로 하는 FPGA를 통한 DSP 구현에 적합한 정수 나눗셈 알고리즘[3]을 제안하였다. 제안된 알고리즘은 FPGA에 내장된 메모리를 LUT로 활용하여 LUT과 곱셈기를 이용한 반복계산 구조를 가지고 있다. 이러한 구조는 메모리와 곱셈기를 탑재한 FPGA에서 소요 Gate수를 최소화할 수 있으며, 또한 고속의 연산을 가능하게 한다. 한편 본 알고리즘은 해를 얻기 위한 반복계산횟수가 일반적으로 알려진 뺄셈 또는 뺄셈-곱셈에 의한 나누기 알고리즘에 비해 매우 적어 Pipeline 구조의 구현에 있어서 소요 자원의 최소화와 함께 Latency를 대폭 줄일 수 있어 고속의 NLMS(Normalized Least Mean Square)적응 필터[4]에서와 같이 Latency가 적은 나눗셈이 필요한 구조에 대단히 적합하다.

본 논문은 Ⅱ장에서 제안한 알고리즘의 기초가 되는 양의 정수 역수 계산 알고리즘을 기술하고 Ⅲ장에서는 이를 이용해 논리회로로 구현 가능한 정수 나눗셈기의 구조를 설명한다. 그리고 Ⅳ장에서는 제안된 구조를 VHDL coding을 통해 실제 FPGA에 구현하여 실험한 결과를 분석하고 Ⅴ장에서는 결론을 맺는다.

Ⅱ. 양의 정수 역수 알고리즘

1. 양의 정수 역수 알고리즘

고속의 디지털 신호처리 알고리즘을 FPGA에 구현하고자 할 때 부동점소수(Floating point)구조로[5] 구현하려면 매우 많은 논리회로 자원이 필요하며, 또한 연산구조의 복잡한 연결로 인해 최대 동작 속도가 떨어진다. 최근의 DSP를 목적으로 상용화된 FPGA들은 다수의 전용 곱셈기를 내장하고 있으나 모두 정수 곱셈기로 아직 부동점 소수 곱셈기를 내장한 FPGA는 출현하지 않고 있다. 신호처리 알고리즘을 정수 연산으로 구현하고자 할 때에는 곱셈기의 최대 bit수가 정해져 있기 때문에 설계에 많은 주의가 필요하다. 예를 들어 NLMS 적응필터의 계수 벡터 W의 Tap-weight 적응 식

$$W_{k+1} = W_k + \frac{\mu e_k X_k}{\varepsilon + X^{T_k} X_k} \tag{1}$$

을 정수 연산으로 구현한다고 가정할 때 FPGA 내부 곱셈기의 최대 bit수가 18이라 가정하면 한 Tap의 계수 적응 스텝의 계산에서 분자는 54bit가 되고 분모는 대략 36bit가 되며, 따라서 각 곱셈마다 적당히 절삭(Truncation)을 하여 18bit이내의 곱셈으로 변환하여야 한다. 나눗셈도 이러한 과정이 필요한데 N-bit 의 양의 정수 Y를 N-bit 의 양의 정수 X로 나눌 때 몫을 Q라 정하면 식(2)와 같은 방법으로 구현할 수 있다.

$$Q' = Y \times (\frac{2^{M-1}}{X})$$
$$Q = (\frac{Q'}{2^{M-1}}) \tag{2}$$

식(2)에서 M=1이고 X가 1보다 크다면 X의 역수는 1보다 작으므로 정수 연산이 불가능하다. 따라서 식(2)에서의 M은 분모에 의한 정밀도 소실을 방지하기 위해 필요하며 값이 클수록 보다 정확한 값을 얻을 수 있으나 구현에 필요한 논리회로의 수가 증가되는 단점이 있다. 식(2)에서 정의한 X,Y의 관계에서 X의 역수 값은 Y값의 관계없이 항상 일정하므로 메모리에 $2^{M-1}/X$값을 넣은 LUT(Look up table)을 만들어 출력 값과 Y 값을 곱하여 나눗셈 결과를 쉽게 구할 수 있다. 그러나 이 때 N-bit 입력에 필요한 LUT의 메모리 용량은 2^N이므로 N이 커지면 커질수록 필요한 메모리의 용량은 많아지는 단점이 있다.

본 논문에서는 N-bit 입력에 대하여 N보다 작고 N/2보다 큰 L-bit (N>L>N/2)의 LUT로 나눗셈을 수행하는 알고리즘을 제안하고 있으며 이러한 구조는 현재 상용으로 널리 사용되고 있는 FPGA들의 내부 메모리로 LUT를 구현하기에 적합하다. 예를 들어 17비트의 양의 정수 역수를 LUT방식으로만 구현할 때 필요한 메모리 용량은 대략 2.2Mbit가 소요되나 제안된 알고리즘에서는 L=10일때 17Kbit로 소요 메모리 용량이 1/127로 줄어들며, 이는 FPGA에서 제공하는 블록 메모리 한 개 정도에 해당된다.

2. 제안된 양의 정수 역수 알고리즘 구조

먼저 어드레스가 L-bit이고 데이터가 M-bit인 LUT 메모리를 가정한다. LUT의 L-bit 어드레스에 대한 M-bit의 데이터 값 $L[X_L]$는 다음과 같다.

$$L[X_L] = \mathrm{INT}[\frac{2^{M-1}}{X_L}] \tag{3}$$

다음으로 N-bit의 입력 값 X에 대하여 상위 L-bit를 X_1, 하위 L-bit를 X_2, 나머지 N-L bit를 각각 X_0, F 로 정의한다.(그림1)

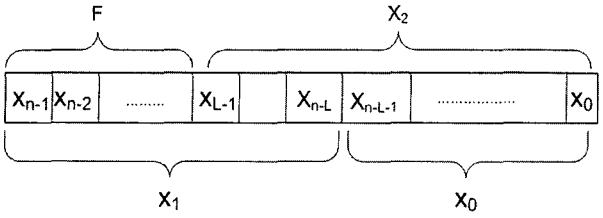

그림 1. . 입력 값 X에 대한 변수 정의.
Fig. 1. A definition of input X variable.

정의에 따라 역수 출력 값 INT(2^{M-1}/X)은 입력 X에 대해 두 가지 경우로 나타낼 수 있다.

(A) 입력 X의 값이 $X \le (2^L - 1)$인 경우 $F = 0$이 되며 역수는 그림 1에서 정의된 변수 X_2를 사용하여 다음과 같은 식으로 나타낼 수 있다.

$$\mathrm{INT}\left(\frac{2^{M-1}}{X}\right) = L[X_2] \tag{4}$$

(B) 입력 X의 값이 $X > (2^L - 1)$이면 $F \neq 0$이며 변수 X_0, X_1을 사용하여 다음과 같은 식을 얻을 수 있다.

$$\begin{aligned}\frac{2^{M-1}}{X} &= \frac{2^{M-1}}{X_1 2^{N-L} + X_0} \\ &= \frac{2^{M-1}}{X_1 2^{N-L}} \frac{1}{[1 + X_0/X_1 2^{N-L}]} \\ &= \frac{2^{M-1}}{X_1 2^{N-L}} \sum_{k=0}^{\infty} \left(-\frac{U}{2^{M-1}}\right)^k \end{aligned} \tag{5}$$

여기서 U는 다음과 같이

$$U \equiv \frac{X_0 2^{M-1}}{X_1 2^{N-L}} = \frac{X_0 L[X_1]}{2^{N-L}} \tag{6}$$

정의한다.

한편 X_0와 X_1의 범위는

$$2^{N-L} > X_0 \ge 0$$
$$2^L > X_1 \ge 2^{2L-N}$$

이므로 X_0의 최대값 $X_{0,\max}$와 X_1의 최소값 $X_{1,\min}$은 각각 $X_{0,\max} = 2^{N-L}$, $X_{1,\min} = 2^{2L-N}$이며, 따라서

$$0 \le \frac{U}{2^{M-1}} < \frac{X_{0,\max}}{X_{1,\min} 2^{N-L}} = \frac{1}{2^{2L-N}} \tag{7}$$

이 성립된다. 식(7)로부터 $L > N/2$ 의 조건을 만족하면 식(5)의 급수는 차수가 증가 할수록 감소하며 다음과 같이 나타낼 수 있다.

$$\frac{2^{M-1}}{X} = \frac{2^{M-1}}{X_1 2^{N-L}} \sum_{k=0}^{K} \left(-\frac{U}{2^{M-1}}\right)^k + O\left(2^{M-N-(K+1)(2L-N)}\right) \tag{8}$$

식(8)에서 K값이

$$K \ge \frac{M - 3L + N}{2L - N} \tag{9}$$

을 만족하는 최소의 정수 값이면 식 (8)의 오차 항이 1/2 이하의 값이 되므로 정수 값은

$$\mathrm{INT}\left(\frac{2^{M-1}}{X}\right) = \mathrm{INT}\left(\frac{L[X_1]}{2^{N-L}} \sum_{k=0}^{K} \left(-\frac{U}{2^{M-1}}\right)\right) \tag{10}$$

이 되며 반복수행(Iteration)의 알고리즘으로 나타낼 수 있다.[6] 식(4)와 (10)으로부터 양의 정수 역수를 산출하는 알고리즘을 나타내면 다음과 같다.

1) 입력 값 X에 대하여 F=0이면 $L[X_2]$ 값을 출력한다.
2) F≠0인 경우, 변수 $L[X_1], U$값을 계산하고 누산기 Q의 초기 값을 $Q[0] = 2^{M-1} - U$로 설정하고 계산한다.
3) $Q[K] = 2^{M-1} - (UQ[K-1])/2^{M-1}$ 연산을 K-1회 반복한다.
4) 최종 값 $\frac{(L[X_1]Q[K-1]/2^{M-1})}{2^{N-L}}$을 계산하여 출력한다.

식(9)에 나타난 바와 같이 최소 반복수행 횟수 K와 LUT의 어드레스 크기 L은 반비례 관계를 가지고 있으며 이들의 크기를 조절함으로써 하드웨어 구현에서의 유연성을 확보할 수 있다. 표 1은 17bit 양의 정수 역수

표 1. M=N=17 일 때 LUT크기와 반복연산 횟수와의 관계
Table 1. M=N=17 Relation between LUT size and iteration.

L	소요 메모리 (Bits)	K (Iterations)
9	8,704	7
10	17,408	2
11	34,816	1
12	69,632	0

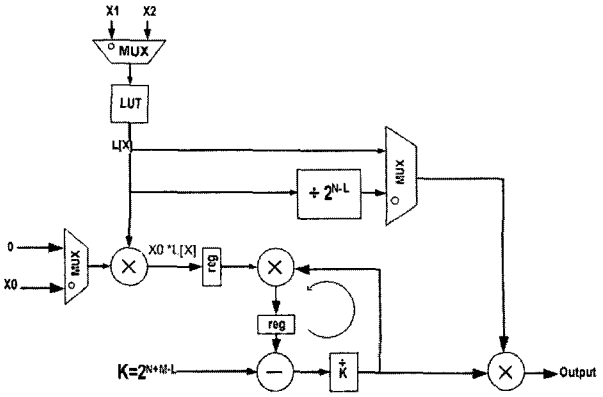

그림 2. 양의 정수 역수 알고리즘 논리구조
Fig. 2. A structure of integer reciprocal algorithms.

를 계산하는 알고리즘에 필요한 LUT의 크기와 반복연산 횟수와의 관계를 나타낸 것이다.

그림 2는 제안된 양의 정수 나눗셈 알고리즘을 논리회로 구조로 표현한 것이다.

입력 값에 대한 변수 정의를 그림 1에서 정의한 변수 F의 상태에 따라 그림 2에서 나타난 곱셈기들의 방향을 제어하며 레지스터들은 알고리즘 단계별 값을 저장하는 역할을 한다. 논리 구조 내의 나눗셈들은 $\div 2^N$ 형태로 레지스터 값들을 쉬프트해서 연결하여 구성하였다.

3. Pipeline 구조의 역수 알고리즘

FPGA의 구동 속도가 신호처리 데이터 속도보다 충분히 빠르지 않을 경우 2절에서 제안된 알고리즘은 Pipeline 구조로 구현되어야 한다. 제안된 알고리즘의 경우 FPGA의 클럭 속도를 f_{ck}, 입력 데이터 속도를 f_{data}라 할 때 식(9)에서 나타낸 알고리즘의 최소 반복횟수 K로부터 $f_{ck} < K f_{data}$일 때 Pipeline 구조의 구현이 필요하다. 2절에서 제안된 알고리즘은 입력 수의 크기에 따라 연산 횟수가 달라지므로 Latency의 차이로 인해 그대로 Pipeline 구조로 변환될 수 없으며, 모든 입력에 대해 같은 Latency를 가지도록 수정된 알고리즘은 다음과 같다.

1) 입력 값 X에 대한 변수 $L_2.U$를 F=0 이면 U=0이고 $L_2 = L[X_2]$로하고 $F \neq 0$이면 $U = \dfrac{X_0 L[X_1]}{2^{N-L}}$, $L_2 = \dfrac{L[X_1]}{2^{N-L}}$로 한다.

2) 누산기의 초기 값을 $Q[0] = 2^{M-1} - U$로 설정한다.

3) $Q[K] = 2^{M-1} - (UQ[K-1])/2^{M-1}$연산을 K-1회 반복한다.

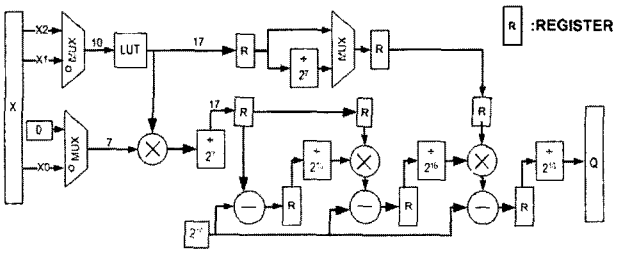

그림 3. Pipeline 구조의 역수 알고리즘
Fig. 3. A reciprocal algorithms of pipeline structure.

4) 최종 값 $\dfrac{(L[X_1]Q[K-1]/2^{M-1})}{2^{N-L}}$을 계산하여 출력한다.

일반적으로 반복연산 구조를 Pipeline 구조로 변환할 때에는 반복연산 구조를 펴서 반복연산 횟수만큼의 연산 구조를 연결한다. 본 논문에서 제안한 알고리즘은 식(9)통해 연결되는 연산구조의 길이를 산출할 수 있으며 이를 통해 소요되는 곱셈기 및 논리회로 자원의 양을 알 수 있다.

그림 3은 17bit 양의 정수 역수 알고리즘을 Pipeline 논리 구조로 표현한 것이다.

4. 반올림을 통한 오차 개선

2,3절에서 제안된 역수 나눗셈은 양의 정수 반복 연산과정에서 Truncation으로 인해 오차가 발생한다. 그림4는 M=N=17 일 때의 본 알고리즘의 출력과 부동점 소수로 계산한 실제 역수와의 오차를 나타낸 것이다. 입력 값은 1~65535까지의 정수이며 그림4의 점선이 출력의 오차를 나타내고 있으며 오차는 [0,1]사이에 있음을 알 수 있다. 이러한 오차는 신호처리에 사용될 경우 출력 신호에 DC 오프셋을 생성할 수 있다. 본 알고리즘에서는 구조상의 큰 변화 없이 LUT 값의 조정과 알고리즘 마지막 단의 한 번의 반올림 연산으로 오차의 분포를 [-0.5,0.5] 사이로 옮길 수 있었다. 반올림을 포함한 LUT값 $L[X_L]$는

$$L[X_L] = \mathrm{INT}[\frac{2^{M-1}}{X_L} + 0.5] \quad (11)$$

이며 최종 값의 출력 과정에서의 반올림은 알고리즘 과정(4) 대신 다음과 같이 계산한다.

$$\mathrm{Output} = \left(\frac{(L[X_1]Q[K-1]/2^{M-1})}{2^{N-L-1}} + 1 \right) / 2 \quad (12)$$

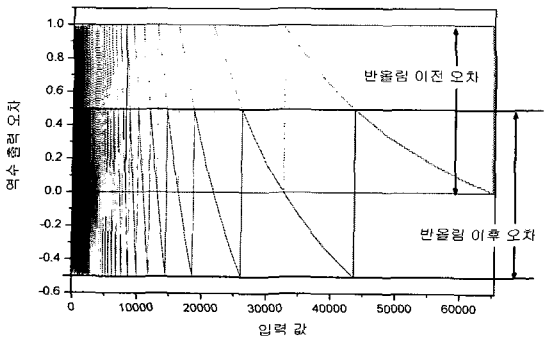

그림 4. 역수계산 알고리즘의 오차 분석
Fig. 4. Error analysis of calculated reciprocal algorithms.

반올림을 포함한 알고리즘의 출력에 대한 오차는 그림 4의 실선으로 나타낸 그래프이며 오차의 범위가 [-0.5,0.5]로 이동한 것을 확인할 수 있다

Ⅲ. 정수 나눗셈

정수 나눗셈은 양의 정수 역수 알고리즘을 기반으로 이루어지고 있다. LUT에 반올림을 한 정수 나눗셈 연산 계산을 식(11)에 따라 미리 값을 정하여 저장한 후 알고리즘 연산을 통한 Iteration을 하여 최종적으로 반올림 연산 과정을 통하여 원하는 나눗셈 결과 값을 얻을 수 있다. 따라서 위에서 설명한 반올림 양의 정수 알고리즘 토대로 개선된 정수 나눗셈 알고리즘을 제안한다. 기본 논리구조는 다음과 같다. 입력 값 X, Y 값이 정수인 경우 즉 MSB 1 일 때 2의 보수 과정을 통해 양의 정수로 바꾸어 위에서 제안한 양의 정수 나눗셈 알고리즘을 수행한 후 최종 결과 값에서 다시 한 번 2의 보수 과정을 거쳐 원하는 최종 값을 얻는다. 입력 값 X,Y 값의 부호가 같을 경우는 2의 보수 과정 없이 나눗셈 연산과정을 통해 값을 얻는다. 한편 입력 값 X,Y 값의 부호가 서로 다른 경우 2의 보수 과정을 통해 양의 정수로 만들어 나눗셈 연산과정을 수행한다. 정수 나눗셈 알고리즘을 이용해 보다 정확한 나눗셈 결과 값을 얻을 수 있다.

다음 그림 5는 정수 나눗셈 연산과정을 Pipeline 논리구조로 표현한 그림이다.

Ⅳ. 나눗셈 알고리즘의 FPGA 구현

제안된 알고리즘의 성능을 평가하고 기존의 알고리즘들[7]과 비교하기위해 알고리즘들을 VHDL로 코딩한 후 코드 레벨에서의 기능을 Mentor Graphics사의 ModelSim을 사용하여 시뮬레이션 및 디버깅을 수행하였고, FPGA에 구현하여 논리 분석기로 결과를 측정하였다. 구현된 알고리즘들은 모두 Pipeline구조의 17bit 양의 정수 나눗셈을 대상으로 하였으며, FPGA는 Xilinx사의 xc4vsx35-10ff668 (Virtex4)를[8]사용하였고, 코드의 합성(Synthesis), Map 및 PAR(Place and Route)을 위한 소프트웨어는 Xilinx사의 ISE 8.1을 사용하였다. 코드들은 모두 300HMz의 클럭에서 동작하도록 설계되었으며 동작 속도를

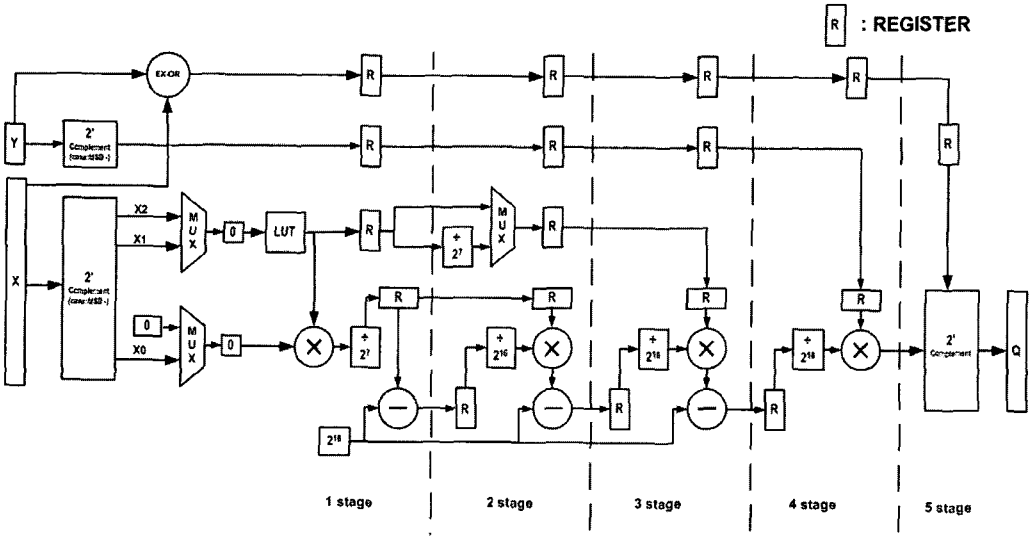

그림 5. Pipeline으로 구성된 정수 나눗셈 알고리즘 논리구조
Fig. 5. An integer division algorithms structure of composed pipeline.

표 2. 각 알고리즘의 FPGA자원 소요 비교표
Table 2. A comparative table of FPGA resources with compared division algorithms.

정수 나눗셈 알고리즘	Slices	Block RAM	곱셈기/누산기(DSP48)	Latency (clocks)
Division by subtraction	1872	0	17	53
Multiply-Subtraction	697	0	16	51
제안된 알고리즘 (LUT-Multiply)	106	1	4	17

실현하기 위해 버퍼 레지스터들이 추가 되었다.

표 2는 알고리즘을 구현하기 위해 소요된 FPGA의 자원과 동작 시 입출력 간의 Latency들을 비교한 것이다. 세 가지 알고리즘 모두 ISE8.1 도구의 최적화 기능을 사용하여 300MHz 클럭에서 동작하기 위한 최소한의 자원을 가지도록 설계 되었으며, 논리 분석기를 통해 정상 동작이 확인되었다. 제안된 알고리즘은 LUT를 위해 한 개의 Block RAM 자원을 추가로 소요한 대신 Slices소요량은 1/6 이하, 곱셈기-누산기(DSP48)의 소요량은 1/4 이하로 자원을 절약 했을 뿐만 아니라 Latency를 1/3이하로 줄일 수 있음을 보여 주었다.

V. 결 론

본 논문에서는 곱셈기와 RAM이 내장된 FPGA에 적합한 정수 나눗셈 알고리즘을 제안하고 이를 VHDL코드 로 변환하여 Xilinx Virtex4 FPGA에 실제로 구현하였으며, 일반적으로 사용되는 정수 나눗셈 알고리즘과 비교하여 그 효율성을 입증하였다. 먼저 FPGA의 소요자원 면에서 17bit 양의 정수 나눗셈을 구현할 경우 Slices자원은 1/6 이하, 곱셈기-누산기 자원은 1/4이하로 소요량을 줄일 수 있었으며, 연산 횟수에서 다른 알고리즘의 연산횟수의 1/3이하로 줄일 수 있었다. 소요자원과 연산횟수의 감소는 서로 상승적으로 구현된 알고리즘의 효율성을 높여주게 되는데, 먼저 적은 연산횟수로 인해 입출력간의 Latency가 다른 알고리즘보다 상대적으로 매우 적었다. 따라서 NLMS 적응 필터의 계수 적응 알고리즘과 같이 지연시간이 적은 Feed-back이 필요한 구조에 적용되기에 적합하다. 또한 소요자원의 감소는 FPGA에 구현할 수 있는 시스템의 집적도를 높일 수 있게 하며, 알고리즘을 구현하는 하드웨어의 소모 전력을 절약할 수 있음을 나타낸다.

참 고 문 헌

[1] 권호경, 문상국, 문병인, 이용식, "Talory 전개식을 이용한 나눗셈 연산기의 VLSI설계", 한국통신학회 논문지, 제26권 C편 pp.6-9, 2000년 6월

[2] 권순열, 최종화, 김용대, 한선경, "고속 십진 나눗셈을 위한 혼합 알고리즘" 대한전자공학회논문지, 제41권 제5호 pp.82-84, 2004년9월

[3] 홍승모, 김종훈, "NLMS 적응 필터의 FPGA 구현에 적합한 효율적인 정수 나눗셈 알고리즘", 하계종합 학술발표회, 제33권 pp.319-320, 2006년7월

[4] Ali H. Sayed, "Fundamentals of Adaptive Filtering", John Wiley & Sons, pp.214-218, 2002.

[5] S. F.Oberman and MJ.Flynn,"Design Issue Dvision and Other Floating-Point operation" IEEE Trans.Computer, vol.46. no2, pp.154-161, 1997.

[6] S. D. Conte C. de Boor, "Elementary Numerical Analysis", 3rd Ed, McGRAW-HILL, pp.33

[7] Stuart F. Oberman and Michael J. Flynn, "Division algorithms and implementation ", IEEE Transactionon Computers, vol.46. no8, August 1997.

[8] Xtreme DSP design consideration user guide "divide" pp.62-65, "Pipeline", pp.44-51.

저 자 소 개

홍 승 모(정회원)

1999년 숭실대학교 정보통신전자공학부 학사
2001년 숭실대학교 정보통신전자공학부 석사
2003년 숭실대학교 정보통신전자공학부 박사 수료

<주관심분야 : 통신 신호처리, 무선통신 시스템설계, Interference cancellation system (ICS) >

김 종 훈(정회원)

1984년 서울대학교 전자공학과 학사
1986년 서울대학교 전자공학과 석사
1993년 Northwestern Univ. Evanston, Ilinois US 공학박사
1995년 6월~1996년 2월 한국통신연구개발원 선임연구원
1996년 3월~ 현재 숭실대학교 IT대학 정보통신전자공학부 부교수

<주관심분야 : 통신 신호처리, 무선통신 시스템, 설계. Interference cancellation system(ICS) >