FPGA에서 FFT(Fast Fourier Transform)를 구현하기 위한 에너지 효율적이고 변수화 된 설계

장 주 욱[†] · 한 우 진[††] · 최 선 일[†††] · Gokul Govindu[†††] · Viktor K. Prasanna[††††]

요 약

이 논문에서 우리는 FPGA에서의 고속 푸리에 변환(FFT)을 함에 있어 에너지를 효율적으로 사용하는 디자인을 제안하고자 한다. FPGA에서의 FFT 구조들은 에너지 손실을 최소화 하기 위해서 설계되어왔다. 가로와 세로의 병렬성 정도와 같은 구조적인 성능 지표들을 정의 했으며, 설계 영역은 설계 디자인들의 조합을 통해서 생성했다. 우리는 에너지를 효율적으로 사용하는 디자인들을 얻기 위해 상위 계층 동작 예측을 사용하여 디자인의 고려사항들을 결정하였다. 우리는 다양한 예측을 위해서 한 무리의 병렬성,radix,저장 형태의 선택등을 갖는 성능 지표화 된 디자인의 집합을 Xilinx Virtex-2 상에서 구현하였다. 우리의 디자인들은 Xilinx 라이브러리에 있는 최적화된 디자인들보다 에너지 손실이 57%에서 78%정도 감소했다. 에너지-영역-시간(EAT)과 같은 이해하기 쉬운 지표를 이용한 결과, 우리의 디자인들이 Xilinx의 디자인보다 3-13배의 성능 개선 효과를 나타냈다.

키워드 : FFT, 에너지 효율, 설계 방법론

Energy-Efficient and Parameterized Designs for Fast Fourier Transform on FPGAs

Ju-wook Jang[†] · Woo-jin Han[††] · Seon-il Choi[†††] · Gokul Govindu[†††] · Viktor K. Prasanna[††††]

ABSTRACT

In this paper, we develop energy efficient designs for the Fast Fourier Transform (FFT) on FPGAs. Architectures for FFT on FPGAs are designed by investigating and applying techniques for minimizing the energy dissipation. Architectural parmeters such as degrees of vertical and horizontal parallelism are identified and a design choices. We determine design trade-offs using high-level performance estimation to obtain energy-efficient designs. We implemented a set storage types as parameters, on Xilinx Vertex-II FPGA to verify the estimates. Our designs dissipate 57% to 78% less energy than the optimized designs from the Xilinx library. In terms of a comprehensive metric such as EAT (Energy-Area-Time), out designs offer performance improvements of 3-13x over the Xilinx designs.

Key Words : FFT, Energy Efficiency, Performance Modeling

1. 서 론

FPGA는 주문제작성과 같은 특징과 높은 처리 능력과 내장된 곱셈기 등과 같은 DSP를 위한 장치, 그리고 RAM과 같은 장치들이 있기 때문에 신호 처리 어플리케이션을 구현하는데 효율적으로 이용되고 있다. 전통적으로 신호처리의 성능 평가 지표는 지연시간과 throughput이다. 하지만 휴대용 장치가 급증함으로 인해서 에너지의 효율성의 중요성이 증가하고 있다. 한가지 에너지를 고려한 어플리케이션이 software -defined-radio (SDR)이다. FPGA기반의 시스템은 적용성과 높은 연산 능력을 요구하는 SDR을 매우 실질적으로 이용할 수 있다.

이 논문에서 우리는 FPGA에서의 고속 푸리에 변환(FFT)을 함에 있어 에너지를 효율적으로 사용하는 디자인을 소개하고자 한다. FFT는 SDR과 센서 네트워크와 같은 광대역 전송 어플리케이션에서 일반적으로 쓰이는 연산 집중적인 방식이다. 우리는 FPGA에 의해서 에너지 손실을 최소화하는 디자인을 조사하고, FFT을 위한 디자인 구조들과 알고리즘들을 적용시켰다. 상위 계층 에너지 동작 모델이 이러한 성능지표들을 이용하여 개발되었다. 이 모델은 디자인의 고려사항들을 결정하고, 에너지 효율의 예측하는데 사용되었다. 성능 지표를 이용한 구조를 설계해서 적절한 성능 지표 값을

※ 본 논문은 정보통신부의 출연금으로 수행한 IT SoC 핵심 설계인력 양성 사업의 수행결과입니다.
† 정 회 원 : 서강대학교 전자공학과 교수
†† 준 회 원 : 서강대학교 전자공학과 석사과정
††† 비 회 원 : Dep. of Electrical Engineering University of Southern California, Ph.D Student
†††† 비 회 원 : Dep. of Electrical Engineering University of Southern California, Professor
논문접수 : 2005년 3월 4일, 심사완료 : 2006년 1월 27일

설정할수 있고, 완전한 디자인의 구조를 쉽게 종합할 수 있다. 우리의 성능 지표화된 디자인은 병렬성의 정도와 더욱 큰 범위를 이용하기 때문에 soft IP core보다 더욱 유연하다. 우리는 에너지 손실 값을 얻기 위해 Xilinx ISE를 이용하여 Xilinx Virtex-2 FPGA상에서 디자인들을 구현하고 실험했다. 우리는 또한 Xilinx 라이브러리에 기반한 디자인들을 우리의 디자인들과 지연시간, 영역, 에너지 손실에 대해서 비교했다. 우리는 또한 모델을 기초로 한 예상값과 구현된 디자인을 기초로 한 실측값을 비교해 보았다. 이러한 비교들은 우리의 디자인들이 지연시간의 중요한 감소뿐 아니라 에너지 손실의 감소를 제공함을 보여준다. 그러므로 우리는 성능 지표화된 구조와 빠른 예측을 위한 상위 계층 모델, 에너지를 효율적으로 사용할수 있는 FFT 디자인의 구현을 제공한다.

이 논문의 다음과 같이 구성되어 있다. 2.1절은 FPGA에서 에너지 손실을 최소화 하는 디자인 기술을 소개한다. 2.2절에서 우리의 성능 지표화된 FFT 구조들을 소개한다. 3절에서 우리의 동작 예측과 FPAG에서의 선택된 디자인들의 구현을 소개한다. 또한 우리의 디자인과 Xilinx 라이브러리를 기초로한 디자인들을 비교한다. 마지막으로 4절에서 결론을 내린다.

2. FFT를 위한 에너지를 효율적으로 사용하는 디자인

이 절에서는 간략하게 에너지의 효율성을 얻을 수 있는 FPGA 기반의 디자인들에 적용될 수 있는 기술들에 대해 알아본다. 그 후 우리는 FPGA에서의 FFT 구조를 사용함에 있어서 에너지를 효율적으로 사용하고, 성능 지표적인 구조를 갖는 우리의 디자인을 소개한다.

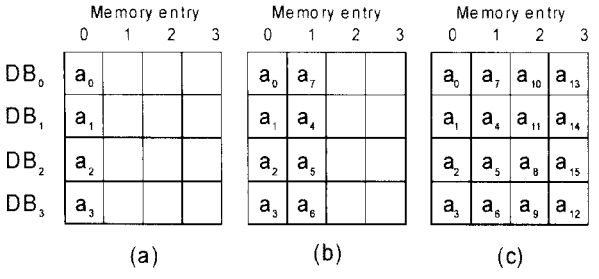
2.1 에너지를 효율적으로 사용하는 디자인 기술들

FPGA 디자인에 적용할 경우에 에너지를 절약할 수 있게 해주는 하위계층 전력 관리 기술들은 많이 있다[7, 9]. 한 가지 방법으로 연산 중에 사용하지 않는 디바이스들을 비활성화 시키는 clock gating이 있다. Xilinx virtex-2 계열의 FPGA들에서 clock gating은 해당되는 연산이 사용될 때에만 clock동작을 활동적으로 수행할 수 있도록 하는 BUFGE와 같은 장치를 사용함으로서 적용할 수 있다. 예를 들어, FFT 연산은 twiddle factor의 연산을 수행하기 위해 많은 복소수 곱셈기를 갖는다.(덧셈/뺄셈에 따른 곱셈). FFT 알고리즘의 성질로 인해서 1, -1, j, -j등과 이들의 연산이 생략 될 수 있다. 그러므로 twiddle factor 연산의 구현은 clock gationg을 이용하여 불필요한 연산 블럭을 비활성화 시킴으로써 가능하다. 에너지를 효율적으로 사용할 수 있는 binding을 고르는 것은 또 다른 기술이다. binding은 연산과 FPGA구성 요소를 짝지어 주는 것이다. 적절한 binding을 고르는 능력은 같은 연산을 위해서 여러 개의 구성이 존재함에 달려있다. 예를 들어, FFT는 큰 데이터 저장 공간을 요구되기 때문에, 저장 공간의 서로 다른 binding들이 에너지 손실에 큰 영향을 미친다. Virtex-2 장치 안에는 레지tm터, SRAM, BRAM의 저장 공간 요소를 위한 세 가지 가능한 binding이 있다. 대용량 저장 공간 요소를 위한 BRAM(48엔트리 이상)은 다른 구현들에 비해서 평균적인 전력 손실을 보여준다. 설계자들은 디자인 요구에 기반한 여러 가지 binding으로부터 발생하는 여러 가지 고려 사항들을 분석할 수 있다. pipeline 구조가 많은 디지털 신호 처리 어플리케이션에서 데이터의 흐름을 처리하기 때문에 선택될 수 있다. 일반적인 데이터 흐름과 관계된 어플리케이션들 때문에 파이프라이닝은 throughput을 증가시킨다. 그러나 파이프라이닝은 디자인 안에 있는 모든 논리들이 계속 활성화되어 있기 때문에 전력의 손실을 증가시킨다. 결국 thoughput은 최대화되기 때문에, 에너지 손실은 감소한다. 게다가 심각한 전력 손실로 인해서 병렬성의 정도와 파이프라이닝의 깊이가 연결을 증가시키고, 이는 곧 에너지 손실로 이어지기 때문에 구현하기 전에 이에 대한 분석이 필요하다. 또 다른 기술은 알고리즘 선택이다. 주어진 어플리케이션들이 서로 다른 알고리즘을 선택해서 FPGA위에 서로 다르게 짝지어질 수 있다.예를 들어 FFT를 구현하는 경우, radix-4에 기반한 알고리즘의 선택은 radix-2에 기반한 알고리즘을 선택했을 경우보다 복소수의 곱셈을 많이 줄여준다. 그러므로 서로 다른 알고리즘과 구조들의 고려 사항들은 에너지를 효율적으로 사용하는 디자인에서 반드시 분석되어져야 한다. 이외의 기술은 [2]에서 소개되었다.

2.2 FFT를 위한 디자인

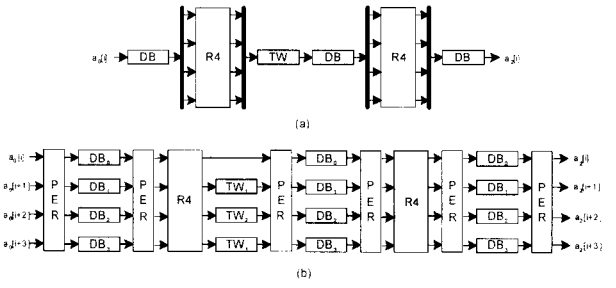
FFT 디자인을 위해서 우리는 잘 알려진 Cooley-Tukey방식을 사용한다. N-point FFT의 계산은 각각의 $log_2(N)$ 단계마다 $O(N)$의 동작을 요구해서, 총 연산은 $O(Nlog_2N)$이다[6].

이러한 사실로 인해서, FFT는 보통 파이프라인 구조가 선택한 데이타의 흐름을 처리한다. N-point FFT디자인은 radix-4 알고리즘에 기반한 것이다. 많은 디자인 성능 지표들이 있지만, 본 논문에서는 FFT 구조를 결정하는 성능 지표들과 결과적으로 에너지 손실에 영향을 주는 성능 지표들을 정의했다. 성능 지표화는 우리가 에너지 효율성에 관한 성능 지표를 기반으로한 디자인 영역을 탐구하기 때문에 우리의 디자인의 중요 요소이다. 다섯가지의 N-point FFT 디자인을 평가하는 성능 지표들이 있다; 1) 문제 크기, (N) 2) 가로 병렬성의 정도 (H_p), 3) 세로 병렬성의 정도(V_p), 4) 저장공간 요소의 binding, 5) 데이터의 정밀함. 가로 병렬성은 얼마나 많은 radix-4단계가 병렬에 쓰이는가를 결정한다. ($1 < H_p < log_4N$). 세로 병렬성은 병렬에 쓰이는 입력값의 갯수를 결정한다. radix-4 알고리즘에서는 최대 4개까지의 입력값이 병렬로 처리될 수 있다. 우리는 [2]에서 소개한 다섯 가지 기본 빌딩 블럭을 고려해왔다; radix-4, 데이터 버퍼, 데이터 경로 치환, 병렬에서 직렬/직렬에서 병렬로의 변환, 그리고 twiddle factor 연산. 각각의 개별적인 블럭은 성능 지표화 되어서 기본 블럭의 조합으로 부터 어떠한 N을 위한 완성된 디자인을 얻을 수 있다:



(그림 1) 16-point FFT를 위한 첫 번째 스테이지에서 DB를 위한 Data permutation (클록 사이클 (a) $t=i$, (b) $t=i+1$, (c) $t=i+3$)

- radix-4 butterfly(R4): 이 블록은 16개의 덧셈기/뺄셈기를 이용하여 덧셈과 뺄셈을 수행한다. 네 개의 입력값을 받아서 병렬로 네 개의 출력값을 생산한다. 각각의 입력 데이터는 실수부와 허수부를 갖는다. 1, -1, j, $-j$를 위한 복소수 곱셈은 입력 데이터 경로를 다시 짝지어주어 덧셈기와 뺄셈기를 통해서 구현했다.
- Twiddle factor 연산(TW) : 이 블록은 twiddle factor들과 함께 데이터의 복소수 곱셈을 수행한다. twiddle factor들은 사인/코사인 테이블로 부터 얻어진다. twiddle factor의 값이 1, -1, j, $-j$인 경우에 곱셈을 우회함으로써 연산과 에너지를 줄일 수 있다.(곱셈기를 비활성화 시킴으로써). 이 블록은 4개의 곱셈기와 2개의 덧셈기/뺄셈기와 두개의 사인 변환기를 포함한다.
- Data buffer(DB): 이 블록은 N/V_p 엔트리를 각각 갖는 두개의 RAM으로 구성되어 있다. 데이터는 한곳으로 기록되고 동시에 다른 한 곳으로부터 읽혀진다. 읽기와 쓰기 동작들은 매 N개의 입력값 후에 바뀐다. 데이터 쓰기와 읽기의 주소들은 구조에 의해서 서로 다르게 진행된다. 예를 들어, N=16이고 하나의 입력값이 있는 경우, 쓰기는 순차적으로 행해지고, 읽기는 네 번 진행된다.
- Data Path Permutation(PER): 병렬 구조에서 쓰인다. (V_p=4). 각 단계의 연산이 끝난 후, 데이터 경로들은 데이터가 병렬로 접근될 수 있고, 다음 단계에서 올바른 순서로 접근될 수 있도록 치환되는 과정을 필요로 한다. 같은 위치 혹은 같은 RAM으로부터 데이터의 접근이 요구되는 경우 의존성이 발생한다. 그림 1은 16-point FFT의 첫 번째 단계의 치환을 보여준다. 첫 번째 클럭 주기에서 네 개의 데이터가 각각의 DB에 병렬로 저장된다. (그림 1)의 (a)를 보라). 두 번째 클럭 주기에서 또 다른 네 개의 데이터가 하나의 위치가 치환되면서 각각의 DB에 저장된다. (그림 1)의 (b)를 보라). 세 번째와 네 번째 클럭 주기에서 같은 방식으로 동작이 수행되고 마지막 결과가 (그림 1) (c)에서 보이는 것과 같다. 네 개의 데이터 a0, a4, a8, a12가 서로 다른 DB에 저장되어서 radix-4 연산이 병렬로 수행되는 것을 볼 수 있다. 치환은 매 단계마다 같은 방식으로 발생한다.
- Parallel-to-serial/serial-to-parallel mux(PS/SP): 이 블럭은 데이터가 radix-4 블럭으로 병렬로 들어오고, 직렬



(그림 2) 16-point FFT의 구조 (a) (H_p, V_p)=(2,1) (b) (H_p, V_p) =(2,4)

구조로 직렬로 나갈 때 사용된다. (V_p<4). radix-4 기기가 병렬로 네 개의 데이터를 처리하는 반면, 나머지 구조는 유연하다. 그러므로 데이터 비율을 맞추기 위해서 직렬에서 병렬로 바꾸는 과정은 radix-4 기기의 이전과 radix-4 기기의 이후에 직렬을 병렬로 바꾸는 변환이 요구된다.

예를 들어, 두개의 radix와 4개의 단계를 갖는 16-point FFT알고리즘의 경우, radix-4 블럭 자원을 공유해서 한 개 혹은 두개의 radix-4 블럭들을 이용할 수 있다 (H_p=1,2). 만약 H_p=1이면, 한 개의 radix-4 블럭이 사용되고 첫 번째와 두 번째 단계에서 공유된다. 그러므로 디자인의 throughput을 감소시키기 위해서 피드백 데이터경로가 필요하다. (그림 2) (a)는 V_p=4, H_p=2일 때 완전한 병렬 구조를 보여준다. 이 디자인은 12개의 데이터 버퍼, 두개의 radix-4 블럭, 3개의 twiddle 연산 블럭을 포함한다. 우리는 또한 다양한 구조들을 위한 관련된 알고리즘을 개발했다. (그림 3)은 (그림 2) (b)에서의 구조를 위한 병렬 알고리즘을 나타낸다. 변수 P는 가로 병렬성에 쓰이고(H_p=log_4N, N=2), 네 개의 펴진 병렬 루프가 있다.(V_p=4). Quad, Dist, P, L, 그리고 R이 병렬로 데이터 버퍼의 구성에 쓰인다.

3. 성능 예측과 설계 합성

구조가 성능 지표화 되었기 때문에, 우리는 성능 지표값을 바꾸어줌으로써 모든 가능한 디자인들을 얻을 수 있다. 그런데, 모든 디자인을 구현하고 실험해보는 것 보다, [1]에서 사용한 상위 계층 모델을 이용해서 동작 예측과 디자인 사항에 대하여 고려하였다. 우리가 사용한 장치는 고성능, Xilinx의 플랫폼 FPGA인 Virtex-II FPGA이다 [8]. 작은 FFT 디자인(N<64)을 위해 XC2V1500을, 큰 FFT 디자인(N>64)을 위해 XC2V3000을 선택했다. 이러한 장치들은 48개와 96개의 18×18 비트 내장 곱셈기를 각각 갖고 있다.

3.1 에너지 성능 예측

데이터의 흐름과 관련해서 FPGA에서는 throughput이 에너지 손실을 고려할 때 중요한 요소이다. 그러므로 우리의 파이프라인 디자인에서는 P가 평균 전력 손실이고 Th가 디

```
Quad = N/4; Dist = N/4;
for P=0 to log4N-1 do parallel. (note: horizontal parallelism, Hp=log4N)
  for K = 0 to 4P-1 do
    L = 4*K*Quad/4P; R = L+Quad/4P-1;
    IdxTw1 = K; IdxTw2 = 2*K; IdxTw3 = 3*K;
    Tw1 = w[IdxTw1]; Tw2 = w[IdxTw2]; Tw3 = w[IdxTw3];
    for J= L to R do
     .do parallel (note: vertical parallelism, Vp=4)
     .   ap+1[J] = ap[J]+ap[J+Dist/4P]+ap[J+2*Dist/4P]+ap[J+3*Dist/4P];
     .   ap+1[J+Dist/4P] = ap[J]-j*ap[J+Dist/4P]-ap[J+2*Dist/4P]+j*ap[J+3*Dist/4P];
     .   ap+1[J+2*Dist/4P] = ap[J]-ap[J+Dist/4P]+ap[J+2*Dist/4P]+j*ap[J+3*Dist/4P];
     .   ap+1[J+3*Dist/4P] = ap[J]+j*ap[J+Dist/4P]-ap*[J+2*Dist/4P]-j*ap[J+3*Dist/4P];
     . end parallel
       do parallel
         ap+1[J] = ap+1[J];
         ap+1[J+Dist/4P] = Tw1*ap+1[J+Dist/4P];
         ap+1[J+2*Dist/4P] = Tw2*ap+1[J+2*Dist/4P];
     .   ap+1[J+3*Dist/4P] = Tw3 *ap+1[J+3*Dist/4P];
       end parallel
    end for
  end for
end parallel for
```

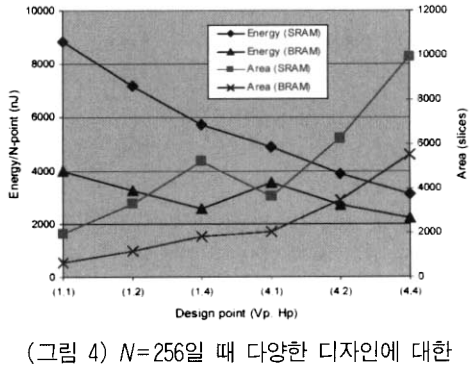
(그림 3) (그림 2) (b)에서 아키텍처에 사용된 알고리즘

자인의 throughput이라고 할 때, 에너지 방정식이 $E=P/Th$ 이다. $1/Th=L$이 디자인의 효율적인 지연시간으로 고려될 수 있다. 효율적인 지연 시간은 파이프라이닝에서의 중복된 연산으로 얻는 이익이다. 2.2절의 구조와 알고리즘에 기초해서, N-point, radix-4 FFT로 이루어진 지연시간(L)을 계산하는 방정식은 다음과 같다; $L=Nlog_4N/(V_p\times H_p)$, 이 때 L은 주기 안에 있다. 이 지연시간을 초로 바꾸려면 클럭 주파수로 나누어주면 된다. 우리는 또한 FPGA 요소들(곱셈기, 레지스터, 등등)의 종류를 알고, 다섯 개의 기본 빌딩 블럭을 이용하는 각각 종류의 요소들의 양을 알고 있다. 각각 기본 빌딩 플록에 쓰인 전력 함수는 [1]의 기술에서 얻었다. 전체 전력 손실을 예상하기 위해 각 블록의 평균 전력 손실을 합했다. 전력은 에너지를 지연시간으로 나누어준 것이기 때문에, 그리고 전에 지연 시간을 계산했기 때문에 전력을 지연시간과 곱해서 알고리즘을 수행하는 동안의 에너지를 예측할 수 있다. 데이터 버퍼에 쓰이는 전력 함수는 radix-4 블럭, 데이터 경로 치환, 직렬에서 병렬로/병렬에서 직렬로의 변환, 그리고 twiddle 연산 블럭인 P_{DB}, P_{R4}, P_{PER}, $P_{PS/SP}$, 그리고 P_{rw}로 이루어져 있다. 여기서 SRAM을 사용하면 P_{DB}=1.23N+35.44(mW)이고, BRAM을 사용하면 P_{DB}=0.0156N+79.65(mW)이고, P_{R4}=142.84(mW), P_{PER}=54.09(mW), $P_{PS/SP}$ =13.52(mW), Block SRAM을 사용하면 P_{TW}=0.0054N+183.57(mW), SRAM을 사용하면, P_{TW}=0.4879N+157.74(mW), 그리고 P_{IO}=44(mW)이다. 따라서 에너지는 다음과 같이 예측된다.

$$\begin{aligned} E = L\{ & V_p(H_p+1)P_{DB} + 2mH_pP_{PER} \\ & + (H_p-1)P_{R4} + 2s(H_p-1)P_{PS/SP} \\ & + t_vt_hP_{TW} + 2V_pP_{IO}\} \end{aligned} \quad (1)$$

m은 데이터 경로 치환 블럭의 개수이다(V_p=4일 때 m=1, 그 밖에는 m=0). s는 병렬에서 직렬/직렬에서 병렬로의 변환이다 (H_p=1일 때 s=1이고, 그 밖에는 s=0). t_v, t_h는 twiddle 연산 블럭의 개수이다(V_p=4일 때 $t_v=V_p-1$, 그 밖에는 $t_v=V_p$; $H_p=log_4N$일 때는 $t_h=H_p-1$, 그 밖에는 $t_h=H_p$).

우리는 식 (1)의 성능 지표값들을 다양한 문제 크기의 에너지 손실과 비교하기 위해 적용했다. FPGA기반의 디자인의 클럭 속도는 장소와 경로 결과에서 얻어질 수 있는 속도이다. (그림 4)는 256-point FFT를 연산할 때 다양한 디자인 포인트들을 예측한 것을 보여준다. BRAM 디자인이 SRAM에 기반한 디자인 보다 좋음을 명확하게 알 수 있다. BRAM을 기반으로 한 디자인은 데이터 버퍼와 페이즈 테이블이 N>64인 환경에서 에너지를 효율적으로 쓸 수 있기 때문에 BRAM을 사용했다. 각 데이터의 정확한 양은 16비트 이다. 우리는 구현된 디자인이 100MHz의 클럭 주파수에서 동작하므로, 클럭 주파수를 100MHz로 가정했다. 서로 다른 성능 지표값들을 사용함으로써 에너지 손실을 최소로 하도록 설정했다.(H_p=4, $V_p=log_4N$, BRAM 기반). 이것은 병렬성이 더 많은 영역을 요구하지만, 에너지를 효율적으로 사용하는 FFT디자인임을 보여준다.



(그림 4) N=256일 때 다양한 디자인에 대한 에너지와 면적의 추정치

〈표 1〉 Xilinx 라이브러리와 성능 비교

Problem	Xilinx (100MHz)					Our designs (100MHz)									Area	Time	E	EAT
size(n)	A(slice)	T(usec)	Em(nJ)	EAT	E/AT	Vp	Hp	Binding	A(slice)	T(usec)	Est(nJ)	Em(nJ)	Error	EAT	E/AT	(inc.)	(dec.)	(dec.)	(dec.)
16	1362	0.16	179.68	0.04	0.83	1	2	SRAM	1171	0.16	65.40	76.99	15%	0.014	0.41	0.86x	1.0x	57%	2.71x
						4	2	SRAM	2390	0.04	63.55	75.18	15%	0.007	0.79	1.75x	4.0x	58%	5.45x
64	1079	1.92	1785.60	3.70	0.87	1	3	SRAM	2266	0.64	552.39	493.32	12%	0.72	0.79	2.10x	3.0x	72%	5.17x
						1	3	BRAM	1613	0.64	464.24	390.40	19%	0.40	0.38	1.49x	3.0x	78%	9.18x
						4	3	SRAM	5690	0.16	393.86	418.68	6%	0.38	0.46	5.27x	12.0x	77%	9.70x
						4	3	BRAM	4193	0.16	403.17	400.41	1%	0.27	0.60	3.89x	12.0x	78%	13.77x
256	1303	7.68	6927.36	69.32	0.69	1	4	BRAM	2050	2.56	2582.17	2223.1	16%	11.67	0.42	1.57x	3.0x	68%	5.94x
						4	4	BRAM	5624	0.64	2203.22	1971.3	12%	7.10	0.55	4.32x	12.0x	72%	9.77x
1024	1557	30.72	34283.5	1639.8	0.72	1	5	BRAM	2741	10.24	14963.5	13739.4	9%	386.06	0.50	1.76x	3.0x	60%	4.25x
						4	5	BRAM	6673	2.56	11424.7	9204.2	20%	157.23	0.54	4.29x	12.0x	73%	10.43

Est is the estimated energy. Em is the measured energy from the synthesized designs. The unit of EAT is E-9. Inc. stands for increment. Dec. stands for decrement.

3.2 합성한 설계의 성능

모든 디자인들은 N, H_p, V_p, binding, 그리고 2.2절에서 언급한 정밀도를 기반으로 성능 지표화하고, VHDL로 코딩한 후에 구현하였다. 동작 예측을 기반으로, 우리는 다양한 문제 크기의 여러 가지 디자인들을 정의했다. 정밀도는 16비트이다. 이러한 디자인들은 Xilinx ISE 4.1i[8]안에 있는 XST (Xilinx Synthesis Technology)를 이용하여 조합하였다. 위치와 경로 파일(.ncd file)은 Virtex-2 XC2V1500과 XC2V3000에서 얻었다. 시뮬레이션을 위한 입력 시험 벡터들은 무작위로 생성했고, 평균 변환 활동성은 50%이다. Mentor Graphics ModelSim 5.5e는 디자인을 실험하고, 실험 결과를 생성하기 위해 사용했다. (.vcd file) 이러한 두 파일들은 평균 전력 손실을 얻기 위해 Xilinx XPower tool을 사용하는데 쓰였다. 에너지 손실값은 평균 전력에 지연시간을 곱해서 얻었다. 에너지 예측 에러(table1 을 보라)는 에너지 손실의 20% 아래였다. 에러의 원인은 대부분 예측이 이것을 포함하지 않았기 때문에 제어 논리에 의해서 발생했다. 만약 디자인의 예측이 에러 범위의 20%내에 있으면, 조합된 값과 실험한 값의 비교가 필요하다. N>64인 경우 BRAM이 SRAM이 에너지를 적게 쓰기 때문에 BRAM을 선택했다.

본 연구는 또한 Xilinx 라이브러리에서 가져온 FFT디자인을 우리의 디자인들과 비교했다. <표 1>의 결과에서 보듯이, 우리의 디자인이 다양한 문제 크기들에서 57%에서 78%적게 손실이 발생했다. 이것은 우리의 디자인이 Xilinx기반의 디자인보다 연산을 빠르고, 에너지를 효율적으로 쓴다는 것을 알려준다. 이해하기 쉬운 단위를 쓰면, EAT(Energy-Area-Time)와 energy/(area×latency)(E/AT), 우리의 디자인이 각각 3-13배, 10-56%의 성능 향상을 보였다. E/AT 단위는 디자인의 평균 전력 밀도이다.

연산 블록을 적절하게 비활성화 시키고, 효율적인 메모리 binding을 고르는 파이프라이닝을 쓰면 에너지를 효율적으로 사용할 수 있다. 파이프라이닝은 throughput을 향상시키고, 효과 지연 시간을 감소시킨다. twiddle 연산 블럭을 비활성화 시키는 것은 에너지의 손실을 30%이상 감소시킬 수 있다. 또한 기본 빌딩 블럭으로 radix-4 알고리즘을 선택하면 디자인에서 복소수 곱셈의 숫자를 줄일 수 있다. N<64인 경우 SRAM, N>64인 경우 BRAM을 사용하는 데이터 버퍼와 사인/코사인 테이블을 선택하면 에너지 효율을 증가시킬 수 있다. 또한 병렬처리가 throughput을 향상시키고 마침내 에너지의 효율성을 향상시킴을 볼 수 있다. 한편, FPGA의 연결에 사용되는 에너지는 증가한다.

예를 들어, N=256이고, (V_p, H_p)=(4, 1)로 디자인하면 (V_p, H_p)=(1, 4)로 디자인 했을 경우보다 throughput은 두 배 더 높지만, 20%더 손실이 발생한다. 이것은 앞의 경우가 연결과 메모리 요소를 더 많이 사용하기 때문이다.

4. conclusion

FFT를 위한 에너지를 효율적으로 사용하는 디자인을 개발하기 위해서 우리는 에너지를 효율적으로 사용하는 기술과 성능 지표화된 FFT 구조에 대해 언급했다. 성능 예측과 디자인 고려사항들은 에너지를 효율적으로 사용하는 디자인을 위해 정의했다. 병렬 구조에서 많은 양의 에너지가 연결에서 손실됨을 관찰했다. 디자인 고려사항과 성능 예측을 모델링하는 작업을 진행 중에 있다.

참 고 문 헌

[1] S.Choi, J.-W.Jang, S.Mohanty, and V. K. Prasanna, "Domain-Specific Modelling for Rapid System-Wide Energy Estimation of Reconfigurable Architectures," Engineering of Reconfigurable System and Algorithms, 2002.

[2] S.Choi, Ronald Scorfano, V.K.Prasanna, and J-w.Jang, "Energy Efficient Signal processing using FPGAs," to appear in ACM Field Programmable Gate Array, 2003.

[3] S.Choi, "Domain Specific Modelling and Energy Efficient Designs for Signal Processing Kernels using FPGAs," Doctoral Dissertation, in preparation, 2003.

[4] C.Dick, "The Platform FPGA: Enabling the Software Radio," Software Defined Radio Technical Conference and Product Exposition, November, 2002.

[5] P.Master and P.M.Athanas, "Reconfigurable Computing

Offers Options For 3G," Wireless Systems Design, pp.20-23, 1999.
[6] A.V.Oppenheim and R.W.schafer, Discrete-Time Signal Processing, Prentice Hall, 1989.
[7] L.Shang, A.Kaviani, and K.Bathala, "Dynamic Power Consumption in Virtex-2 FPGA Family," International Symposium on Field Programmable Gate Arrays, 2002.
[8] Xilinx Application Note, Virtex-2 Series and Xilinx ISE 4.1i Design Environment, http://www.xilinx.com,2001.
[9] G.Yeap, Pratical Low Power Digital VLSI Design, Kluwer Academic Publishers, 1998.

장 주 욱

e-mail : jjang@sogang.ac.kr
1983년 서울대학교 전자공학(학사)
1985년 한국 과학기술원(석사)
1993년 미국 University of Southern California 컴퓨터 공학과(박사)
1985~1994년 삼성전자 컴퓨터 개발실
1995~현재 서강대학교 전자공학과 교수
관심분야 : 병렬처리, IT SoC, 인터넷 프로토콜

한 우 진

e-mail : birth0531@eeca1.sogang.ac.kr
2005년 서강대학교 전자공학(학사)
2005~현재 서강대학교 전자공학과 석사과정
관심분야 : IT SoC, 인터넷 프로토콜, ad-hoc

최 선 일

e-mail : seonil@usc.edu
현 재 미국 University of Southern California 전자공학 박사과정
관심분야 : 슈퍼컴퓨팅, 병렬 분산 시스템, 네트워크 컴퓨팅

Gokul Govindu

e-mail : govindu@usc.edu
현 재 미국 University of Southern California 전자공학 박사과정
관심분야 : 슈퍼컴퓨팅, 병렬 분산 시스템, 네트워크 컴퓨팅

Viktor K. Prasanna

e-mail : prasanna@usc.edu
인도 Bangalore University 전자공학(학사)
인도 Indian Institute of Science 전자공학(석사)
미국 Pennsylvania State University 컴퓨터학(박사)
현 재 USC(Univ. of Southern California) 컴퓨터학과 교수
관심분야 : 연산처리, 병렬 분산 시스템, 네트워크 컴퓨팅, 임베디드 시스템