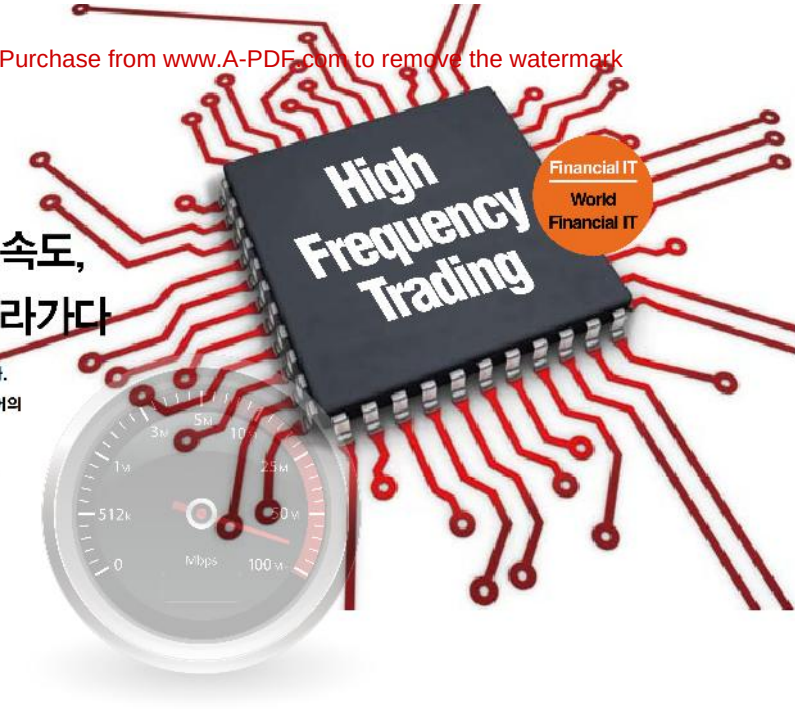

고빈도매매의 속도, FPGA로 더 올라가다

고빈도매매에서 중요한 것은 속도다. 속도를 더 높이기 위해서는 하드웨어의 조정이 필요하다. FPGA는 이를 뒷받침해줄 중요한 방법이다.

글 브라안 더우드, 닉 그랜니
사진 SHUTTERSTOCK

Wallstreet & technology 특약

마이크로세컨드 단위의 고빈도매매가 현실화 되었다. 열두 곳 가까운 제조업체들이 경쟁력이 높은 FPGA Field Programmable Gate Array기반 하드웨어를 내놓은 이후 지난 18개월동안 고빈도매매는 한 단계 도약했다. 이제는 저가의 2U에 2~4개의 양방향 10GBP/S 포트가 탑재된 5,000~15,000 달러의 PCIe 보드가 산업 표준이 된 듯하다.

크게 보면 고빈도매매는 '건슬링거'Gunslinger 비즈니스다. 총잡이 중에 가장 빠르게 총을 뽑는 자가 이기는 것처럼 가장 빠른 속도로 매매하는 자가 승리를 하고 승자가 거의 모든 것을 독식한다. 따라서 지속적으로 기술적인 경쟁력을 가져야 한다.

하드웨어가 동작하는 원리

밀리세컨드로 동작하는 기존의 자동화 매매 시스템은 데이터가 급증하는 상황에서는 데이터 처리가 늦어지기 마련이다. 이러한 지연 현상이 나타나는 이유는 시스템의 '비결정성' 또는 비선형 부하 처리로 인한 것이다. 즉 PC를 사용할 때 최대 부하가 걸리면 CPU가 급속히 느리게 동작하는 경험을 한다. 이와 같이 병목을 일으키는 부분을 CPU가 아닌 FPGA에서 처리하도록 하면 예측가능하고 지연이 적은 시스템을 만들 수 있다. 물론 FPGA는 프로그래밍이 가능한 '게이트의 묶음'에 불과하다. 또한 FPGA는 현재의 마이크로프로세서보다 낮은 클락 속도로 동작한다. 하지만 병렬로 연결하면 동시에 수백에서 수천가지를 처리할 수 있다. 따라서 뛰어난 기술적 능력과 결합하면 FPGA는 마이크로프로세서보다 몇 배 더 빠른 속도를 낼 수 있다. FPGA는 병렬처리를 통해 병목 로직을 없앨 수 있기 때문이다.

구조적 고려사항

많은 기술자들이 CPU를 기반으로 하는 하드웨어로 트레이딩 시스템을 개발하고 있다. 하지만 새롭게 FPGA를 도입하는 것은 '이종 프로세싱'Heterogeneous Processing이라고 불리는 분야와 접목해야 한다. 기본적으로 다른 하드웨어에 다른 방식으로 기능을 부여해야 하기 때문이다. 이를 몇 가지 개념으로 정리할 수 있다.

기본적으로 CPU에서 처리하는 모든 작업들은 동일한 순위를 가진다. CPU는 '교통 경찰'과 같이 작업를 처리하기 때문에 우선 순위를 정하여 처리한다. 반면 FPGA는 수백만 개의 게이트로 이루어졌지만 자원은 유한하고 이 또한 데이터 버스 및 메모리 컨트롤러와 같은 특정한 목적으로 사용된다.

게이트들만큼 자원들간의 연결(라우팅) 또한 제한적이다. 따라서 중요도가 낮은 로직은 CPU에 할당해 FPGA 자원을 보존하는 것이 유리하다.

또 가능한 많은 정보를 게이트에 저장할 필요가 있다. FPGA를 이용할 때 메모리 또는 CPU의 자원을 이용하면 시스템이 느려진다. 많은 개발자들은 메모리를 이용하는 것이 시스템을 효율적으로 구성하는데 중요하다는 것을 알고 있다. 그렇지만 FPGA의 세계는 다르다. 프로파일링(프로그램의 시간 복잡도 및 공간(메모리), 특정 명령어 이용, 함수 호출의 주기와 빈도 등을 측정하는 동적 프로그램 분석의 한 형태) 또한 제한적이다.

결국 FPGA 개발자는 컴퓨터나 메모리에 종속적인 프로그램을 FPGA로 옮기는 일을 해야 한다. 여기엔 행운도 왕도도 없다. 루프 풀기Loop Unrolling[1]를 사용하여 코드를 최적화하여야 하고 프로세스는 병렬화하여야 한다. 더구나 소스코드 중 어떤 부분을 FPGA로 옮길지를 판단하는 묘수도 없다.

Impulse C와 같은 개발도구는 시각화기능을 제공한다. 과부하가 걸린 버퍼를 확인할 수 있고 단계별 처리속도 분석으로 지연구간을 확인할 수 있다. 그렇지만 코드의 최적화는 무조건 반복적인 작업을 해야 한다. 병렬화를 효율적으로 할 수 있도록 어느 부분을 큰 단위coarse-grained의 코드로 할지, 코드를 디버깅하기 위하여 어디에서 브레이크포인트를 사용할지를 결정할 때 의사결정을 잘해야 한다.

다만 많은 FPGA 컴파일러가 확장성이 많은 예제를 제공하고있어 개발 경험이 없는 개발자라도 예제를 가지고 한 줄씩 코드를 추가하면서 배울 수 있도록 해준다. 매매시스템의 경우 대부분 턴키방식으로 시스템을 도입하고 독자적인 알고리즘만을 개발하여 추가하는 것과 비슷하다.

소프트웨어는 어떻게 동작하는가

C에서 VHDLVHSIC Hardware Description Language로, 이를 다시 Synthesis(VHDL과 같은 HDL 언어를 해석하여 동일한 하드웨어 토폴로지를 생성하여 내는 과정)한 이후 FPGA에 이식하는 과정은 4시간정도 걸린다. 하지만 소프트웨어 코드에 최적화하도록 FPGA를 배열하여야 하고 이를 위해 수백만 개의 게이트를 연결하는 일은 몇 시간이 걸린다. FPGA 하드웨어 컴파일을 위해 제공하는 비주얼 스튜디오 기반 C 인터페이스 역시 자동으로 단계별 지연 분석을 수행하며 PCIe 보드 상에서 FPGA를 컴파일 한다.

1980년대 HDLHardware Description Language이 품질 때문에 수작업으로 테이프 아웃(설계를 생산공정으로 넘기는 것)을 대체하지 못할 것이라고 한 사람들이 있었다. 그렇지만 HDL범주의 언어들이 발전하고 적기 생산이 가지는 이점이 중요해지면서 HDL이 수작업Hand Coding을 대체하였다. C, OpenCL 혹은 Vivado HLS와 같은 고급언어High-level programming language가 등장하면서 HDL도 같은 운명을 맞고 있다.

만약 하드웨어와 관련한 코딩을 해본 경험이 있다면 수작업 방식의 HDL이 자동화한 HLL보다 훨씬 뛰어나다는 것을 알 것이다. 만약 HLL로 새로운 프로그램을 개발하려면 HDL로 컴파일을 하고 최적화Hand Polishing를 위해 HDL프로세스를 재배열한 후 숙련된 기술자가 전 과정을 세심히 점검할 필요가 있다. 불행하게도 최적화하는 과정은 C언어로 개발된 소스와 무관하다. 이 때문에 어플리케이션을 유지보수하는 것이 더 힘들어지므로 가능하면 피하여야 한다.

간혹 FPGA에 컴파일한 후 리소스가 부족한 현상을 만날 수 있다. 때문에 FPGA에 컴파일할 때 주의하여야 하는 것은 FPGA 내에 특별 용도의 블록을 두는 것이다. 능숙한 FPGA 디자이너는 하드 DSP 블록 또는 하드 코어 또는 소프트 코어 프로세서를 사용하여 마법을 연출할 수 있다. 하지만 프로세스의 사용법을 모르고, 설계된 것보다 하나라도 더 많은 DSP 블록을 이용한 경우가 일어나면 리소스 부족을 발견할 때까지 성능은 급격히 떨어질 수 밖에 없다.

고빈도매매를 하고 있다면 치열한 경쟁 속에 있다. 특정한 FPGA 벤더의 임베디드 기술에 연관될수록, 더 빨리 (더 크고 더 빠른) 다른 플랫폼으로 옮겨간 경쟁자에게 뒤처질 가능성이 더 높아진다. 따라서 보편적인 방법을 가지고 더 큰 유연한 구조를 가져야 한다! 고급언어를 통한 설계는 이식성이 가장 높으며 새로운 하드웨어로 이동하는 시간을 극적으로 줄여준다.

개발을 위해서는 다음과 같이 권장하는 바이다. 가능한 고급언어HLL를 이용해야 한다. 고급언어를 더 많이 사용할 수록 C 프로그래머의 직무는 더 쉬어지고 새로운 하드웨어로 더 빨리 이동할 수 있기 때문이다. HDL 내의 속도를 위해서는 수동으로 처리할 수 있는 비트들에 대해 지속적으로 관심을 가져야 한다. 전반적으로 반복에 반복을 거듭한다. FPGA는 지속적 진화를 통하여 결과를 얻어가는 분야이기 때문이다.

1)최적화 기법 중에 하나로 루프의 횟수를 줄이는 방법이다.

*이 기사의 원문은 www.wallstreetandtech.com/trading-technology/a-guide-to-the-hft-arms-race/240165050에서 확인할 수 있습니다.