1. Intro

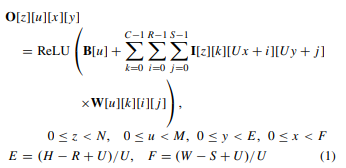
* 현대 CNNs의 당면 과제
* Energy efficiency
* Throughput
* Solution : 먼저, Data movement energy >>>>> computation
* Data reuse : locally reduce expansive computation (=Dram 액세스)
* Compression & data gathering
* CNN processing 으로 높은 처리량을 병렬화, 데이터 무브먼트 최적화, 겟 하이 에피션시

+ 더하여, 다양한 차원과 사이즈의 CNN을 최적화.

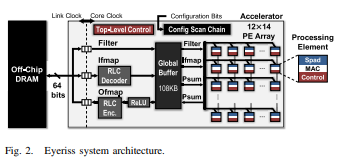
* Exploit
* Compute paradigm : Dataflow – 높은 병렬화, 온칩 오프칩에서 둘 다 최적화
* Data reuse : multilevel memory hierarchy (Caching 인듯)
* Hardware need : Reconfigure 할 수 있는 하드웨어 필요(궁극적 목표는 다양한 쉐잎이며 이 때 필요한게 FPGA가 아닐까)
* Data statistics : CNN은 ㅈㄴ 0이 많음. 따라서 압축과 적응형 프로세싱이 적용되고 이는 메모리 bandwidth와 processing power를 절약해줌
* Problems
* 현재 나와있는 solution들이 대부분 시뮬레이션이며, 제대로 tapeout된 칩이 없음.
* 심지어 FPGA역시 완벽한 결과를 반영하지 않음.
* 결국 fabricated chipe이 필요함 최신의 CNNs을 메저하기위해.
* 다양한 논문들이 진행됐었음. 그러나 이 논문들은 목적하는 compute schemes를 위한 DRAM 밴드위쓰를 확실히 체크하지 않았음. 파워를 체크하는 것만으로는 프로세서의 성능을 확실히 알아볼 수 없음. 왜냐하면, Dram access는 시스템의 에너지 효율을 dictating하는 가장 dominant factor이기 때문임.
* paper에서 할 것, CNN 칩을 tape out하고, 그 칩은 높은 처리량을 가진 CNN 추론을 지원하며 전체 시스템에서 전력 소모를 최대한 효율적으로 만들것임. (on chip off chip둘다)
* Eyeriss의 Feature
* 168개의 PE(processing elements)가 있고, 이는 4개의 memory hierarchy를 만들어냄. 이때, Data movement는 되도록 low-cost level에서 진행됨(PE scratch level, inter PE communication), 이 때 GLB ( large on chip global buffer)와 off chip DRAM이 포함됨
* CNN dataflow (RS)는 spatial architecture를 reconfigure함 (결국 주어진 shape에 맞춰 최적화 한다는 이야기임)
* NoC (Net on chip) 구조가 RS dataflow를 지원하기위해 동원됨. (multicast, single cucle 데이터)
* 아까 말했던 통계적 method, Run-length compression과 PE data gathering이 CNN 내부의 0데이터를 최대한 이용하게 하고, 에너지 효율성을 증가시켜줌
* Alexnet과 VGG-16net으로 평가할것임.

1. CNN basics

* CNN algorithm
* 현대의 CNN은 feature extraction & classification을 위해서 딥한 레이어를 씀.
* 해당 레이어는, input image data를 매우 추상적인 표현으로 변환함 (f맵)
* 또한 레이어는 필터를 ifmap에 적용한다. 그리고 ofmap을 만든다. (psum으로)
* Layer
* 각각의 필터와 르메은 4D : 각 필터와 feature map이 3D이고 2Dmulti-plane추가
* 또한 이 때에, 1D bias가 추가되는데, bias는 밝기관련 parameter임
* Tb 1의 shape parameter를 고려하면 아래와 같은 식이 완성된다.



1. System Architecture



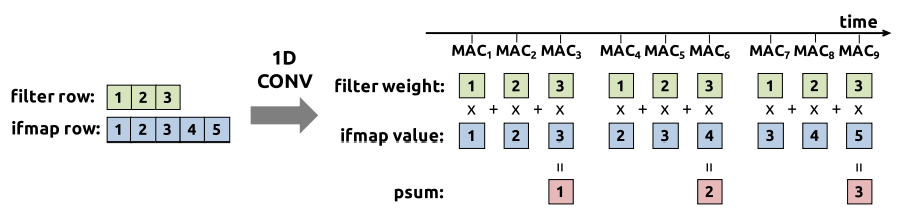
* 위는, 아이리스의 구성이다. 두개의 Clk domain로 구성되며, 각 processing clk, off-chip Dram communication clk으로 구성된다.
* 두개의 클락 도메인은, 각각 동기화되지 않은 FIFO를 사용한다.
* Data Transfer를 하기 위해, PE는 NoC를 통하여 주변 PE와 GLB에 커뮤니케이션을 할 수 있어야댐 혹은, spad라고 불리는 memory space가 필요함
* Overall, 4개의 계층이 존재하며, 이는 DRAM, GLB , inter-PE, spads임 (descending energy)

1. System control and Configuration

* 위에서 언급 했듯이, 두 개의 clk가 있으므로, two level의 control hierarchy가 존재함.
* Top : DRAM , GLB 관리 , GLB ,PE 관리, RLC Codec과 Rely관리
* Low : PE의 control logic 관리 (processing)
* 모든 PE가 identical하게 same clk에서 돌고 있어도, lock step이 필요치않음
* CNN layer의 작동
* 1794b의 scan chain layer에 순차적으로 로드하여 전체적 가속기를 재구성(100ms)
* 이 때, 비트에는 특정 형태의 필터 및 f맵처리를 위해 가속기 구성 PE어레이 매핑 및 NoC 데이터 전달의 패턴 설정 등이 포함댐 (offline에서 생성댐. Runtime에 정적)
* 이후, 가속기는 처리를 위해 DRAM에서 ifmap및 필터의 타일을 로드, 계산된 ofmap은 DRAM에 다시 기록댐

1. Processor 의 Efficiency factor

* 1. Reducing data movement
* 2. Exploiting data statistics
* Row stationary
* Any given CNN에 대해 pe array로 바꾸어줌. 그것은 reconfigurable하며 optimize함
* 이는 모든 data movement type( ifmap…등) 에 대해 최소화함
* Data reuse와 low cost movement를 통해 GLB와 DRAM의 access가 최소화됨
* Reuse 종류 : Cov reuse, filter reuse, Ifmap reuse를 함. (아래 그림 참조)



* Minimize Psum movement는 psum acc로 행해짐.
* 그러나, maximum reuse는 갑작스러운 reduction으로 행해지기 어렵고, 이유는 mac연산중이기 때문임. 이를 해결하기 위해서 systematic approach를 함.
* 1D convolutional Primitive in a PE
* 각각의 RS data가 parallel하게 들어옴. 이 때, primitive operator가 apply되고, 1개의 psum 만들어냄
* 다른 psum 들이 accumulate되고, 이에 따라 ofmap value를 만들어냄
* 처리를 위해, 각 primitive를 PE에 매핑하면, 1pair의 computation이 고정됨
* 위의 그림처럼 processing되는 동안 각 PE는 local spads를 쓸 수 있음
* Data의 window를 sliding하기에, spads는 filter row size에만 영향을 받음
* 일례로 alexnet은 11,5,3의 filter size이기에, min spads는filter, ifmap, psum 11, 11, 1과 같다. (psum은 당연히 1)
* NoC
* NoC는 GLB와 PE array뿐 아니라 inter PE communication역시 manage해줘야함
* RS data flow를 support하기위해, 3가지 시나리오가 존재함. 이는 각 CON1 , CON2 , CON4 CON5를 이야기하며, 이는 내용에 나와있음 (7page)
* NoC는 RS dataflow 에서 data reuse를 해야하며, 에너지 효율을 향상시킬 수 있어야댐
* PE 어레이에서, 높은 parallel processing을 위해, 충분한 bandwidth를 제공해줘야함(DATA)

1. Global Input Network (GIN)

* Single-cycle multicast를 GLB부터 같은 input이 apply되는 PE까지 apply해준다.
* ID tag로 arbitrary한 mapping을 cover해준다. (type, stride, mapping 등 다양한 변수)
* 이 때, 처음부터 all 분류는 costly하므로, hierarchy를 만들어 cover함 (X버스, Y버스)
* 모든 GIN은 three data types (filter, ifmap, and psum)을 갖고 있으며, 이 때, 12row를 위해 4b row ID, filter와 psum을 위해 4b col ID를 사용한다 (14cols)

1. Global output Network

* GON은 주로, processing pass를 거친 후 생성되는 psum을 GLB에 다시 올려주는 역할

1. Local Network

* 1쌍의 PE에 대해, 두개의 연속적인 col을 가진 row가 들어오면, 64b reg가 bottom에서 top으로 direct하게 pass시킨다. (바로 연산 할 수 있도록)
* 따라서 PE는 GIN과 LN에서 모두 input을 가져올 수 있다.
* 이러한 선택은 레이어에서 정적이며, CNN 모양의 Dataflow mapping에만 의존하는 scan chain configuration에 의해 제어된다.
* Processing element and Data Gating
* PE에는 FIFO가 존재하며, 이는 NoC와 computation의 work load균형을 맞춰줌
* PE가 한번 processing했던 p와q은 PE의 control에 따라 정적으로 구성됨
* Datapath는 3stage로 pipelined되어 있으며, 1개의 spad access(fetch)와 2개의 computation으로 구성되어 있음. 두개의 computation은 덧셈과 곱셈이며, 이 때, 곱셈의 결과는 32b에서 16b fixed point로 truncation됨. 16b의 선택은 순전히 offline experiment에 의해 조절할 수 있음.
* Data gathering logic은 ifmap의 0을 이용하며, 에너지를 절약함. Extra 12b이 zeros를 위해 buffer로 할당됨 당연히 이는 zero의 position를 기록
* 이 gathering logic은 zero buffer의 0이 ifmap에서 감지되면, 바로 spads read를 중단하고, MAC datapath로의 switching을 방지함