디지털 시스템 설계 및 실험 프로젝트 보고서(4조)

2016-15051 차민철

1. 역할 및 contribution

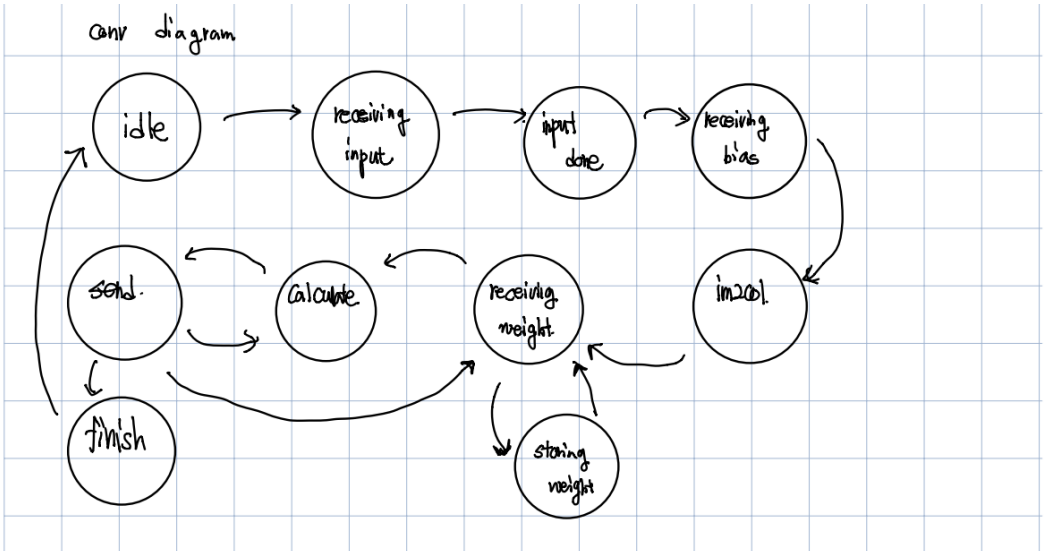
저는 PE를 만들었고 적절하게 받아지는 데이터에 맞추어 PE4를 작성하였습니다. PE4는 PE들을 4개 연결하여 1x4로 5개의 input을 한번에 계산 가능하도록 하였습니다.
이 PE4를 사용하여 conv와 fc에서 모든 calculate 과정을 제가 맡아서 진행하였습니다. 또한 계산 결과를 send하고 데이터를 몇 개를 더 받아와야 하는지 아니면 계산이 끝났는지를 control 하는 부분을 전부 맡아 진행하였습니다. 특히 conv layer에서 systolic array를 잘 적용하기 위한 im2col array 의 저장형식을 만들어 요청하였습니다.

systolic array를 사용하기 위해 PE를 만들어주었는데 기존의 systolic array와는 좀 다른 부분이 clk, rst_n in_a, in_b, out_a, out_b, out_c
외에 input으로 in_fin_a out_fin_a를 추가하여 계산하고자 하는 input이 끝났다는 신호를 추가적으로 넣어주었습니다. 이렇게 진행하면 마지막 PE block이 out_a 가 1이 된다면 계산이 완료되었다는 의미이고 그때 결과를 출력해주면 되기 때문에 이와 같이 진행했습니다.
이 PE4개를 이어 만든 PE4를 통해 계산을 하였습니다.
계산 accuracy가 줄어들 것을 염려해 signed로 out_c를 32비트로 지정해주었으며 PE4를 만들 때 bias를 넣는것도 추가하여 계산값 + bias를 quantize한 값을 temp가 항상 가지고 있다가 마지막 블록이 out_fin_a가 1이 되면 위에서 설명했듯이 계산이 끝난것이므로 그때 PE4가 속해있는 모듈에 계산이 완료되었음을 보내주도록 작성하였습니다.
한칸씩 PE를 옮겨가며 넣어주는 걸 구현하기 위해 PE_clk을 새로 만들어서 미리 숫자는 대기시켜놓은 상태로 PE_clk이 뛸때를 기준으로 한 숫자 열씩 업데이트되도록 구현을 하였습니다. 이와 같이 구현한 결과 2clk edge 만에 4개의 숫자열이 계산되는 것을 확인하였습니다.
conv layer를 작성하며 이 과정에서 im2col 된 결과를 저장하는 형식을 구상하였습니다. 그 형식을 통해 팀원이 만들어준 4줄의 im2col 된 feature를 제가 적절하게 끊어주어 팀원이 제공해준 bram에 저장한 weight parameter를 꺼내 계산을 한 후에 send에서 control을 알맞게 진행해주었습니다.
conv 부분에서 calculate의 control 부분은 9* in_ch만큼끊어서 32비트씩 즉 총 4개 output을 계산하였습니다. 그 과정에서 9* in_ch를 세는 카운터 한 개, 전체 지금까지 누적되는 카운터 한 개를 사용하여 첫 번째 카운터를 기준으로 in_fin_a를 넣어주었고 두 번째 카운터를 기준으로 언제 weight를 또 받아와야하는지, 어느 인덱스의 bias와 weight를 받아오는지 등의 컨트롤을 진행하였습니다. 이를 통해 데이터를 받아오는동시에 PE_clk 과 PE_set을 올려주면서 알맞게 시간 조절을 하여 지속적으로 PE4에 원하는만큼 들어가도록 진행하였습니다. im2col 부분도 계산하기에 적절하게 각 3x3 즉 9개의 im2col 마다 4개 output 씩 끊어서 4개의 bram에 순차적으로 저장하도록 하여 계산시 9*in_ch로 바로 인덱싱하여 계산 가능하도록 구상해서 요청하였습니다.

이렇게 저장된 숫자의 convolution, array multiplication 등의 전체 계산과정, quantization, relu 부분과 같은 숫자 처리 과정을 도맡아 진행하게 되었습니다.

마지막으로 저희가 프로젝트를 완성하고나서 세세한 버그들을 주도적으로 수정해나갔습니다. 주어주신 testbench를 바탕으로 control signal 관련된 작은 수정사항들을 전부 수정했습니다.

2. conv layer



state 설명

1) Idle:
시작 전 사용할 모든 값들을 초기화 command 신호가 1이 들어오기 전까지 계속 idle state를 유지하다가 command가 1이 되면 receiving input state로 이동한다.

2) Receiving_input:
VDMA를 통해서 feature data를 받는 state이다. Size가 8x8192 bram에 모든 feature data를 저장하면 input done state로 이동한다.

3) Input_done:
Command 신호가 2가 될 때까지 대기한다. command신호가 2가 되면 receiving bias state로 이동한다.

4) Receiving_bias:
VDMA를 통해서 bias data를 받는 state이다. Size가 32x64인 bram에 모든 bias data를 저장하면, im2col state로 이동한다.

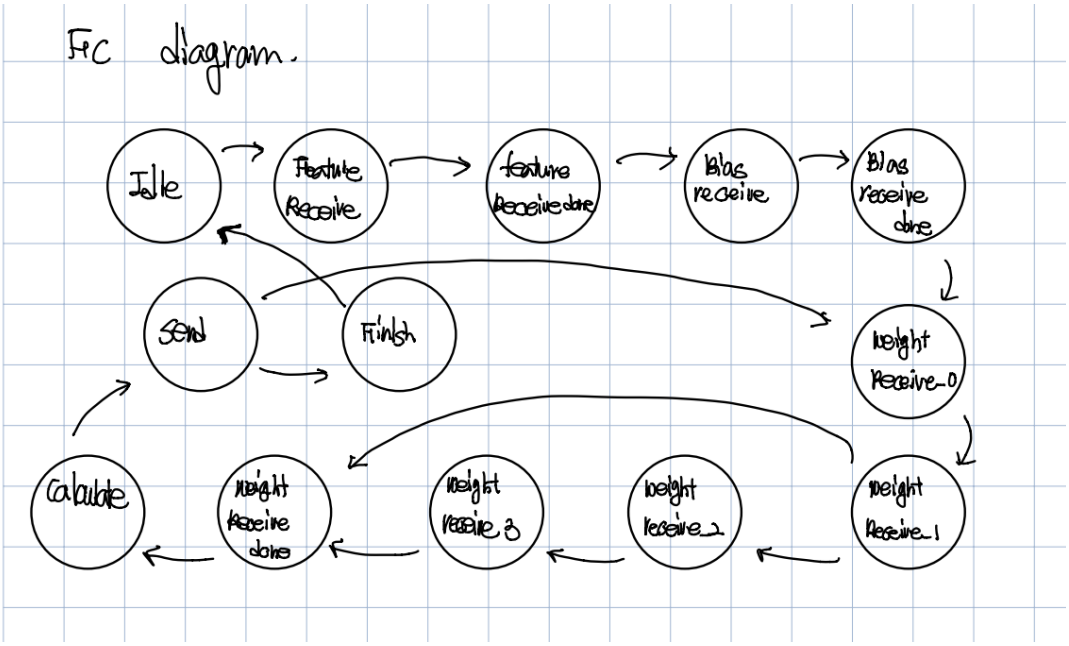
5) Im2col:
후에 나올 state인 calculate에 이용할 operands를 generate한다. Receiving_input state에서 8x8192 bram에 저장했던 정보들을 im2col 한 결과 값을 8x18432 bram 4개에 저장한다. 그 후 receiving weight state로 이동한다.

6) Receiving weight:
VDMA를 통해서 weight data를 받는 state이다. 한번에 모든 weight정보를 저장할 수 없으므로 288bits 크기인 reg가 다 차면 storing weight state로 이동한다. 만일 한 번 calculate를 할 만큼의 정보를 수신했다면, calculate stae로 이동한다.

7) Storing weight:
Receiving weight state에서 받은 288bits reg에 담긴 정보를 8x1024 bram 9개에 저장한다. 이를 마치면 다시 receiving weight state로 돌아간다.

8)calculate:
순차적으로 im2col 된 bram 과 알맞은 filter의 entry 를 받아와서
PE4로 숫자 4개 즉 32비트의 output을 계산해낸다.
계산 완료 후 sendstate로 이동
9) send
calculation 에서 받은 내용을 send 한 후에
1) 하나의 channel이 다 안끝난경우 -> calculate
2) 4개의 kernel 에 대한 계산 끝난경우 -> weight receive
3) output channel만큼의 전송이 다 끝난경우 -> finish

3. fc layer



state 설명

idle: start command가 들어오기 전까지 대기. command가 들어오면 featureReceive로 넘어감

featureReceive: input을 받는 state. 필요한 횟수만큼 데이터를 받으면서 bram에 저장. input을 모두 저장했다면 featureReceiveDone으로 넘어감

featureReceiveDone: module output으로 featureReceiveDone 신호를 출력 및 bias receive start command가 들어오기 전까지 대기. bias receive start command가 들어오면 biasReceive로 넘어감

biasReceive: bias를 받는 state. 동작 방식은 featureReceive와 동일. bias를 모두 저장했다면 biasReceiveDone으로 넘어감

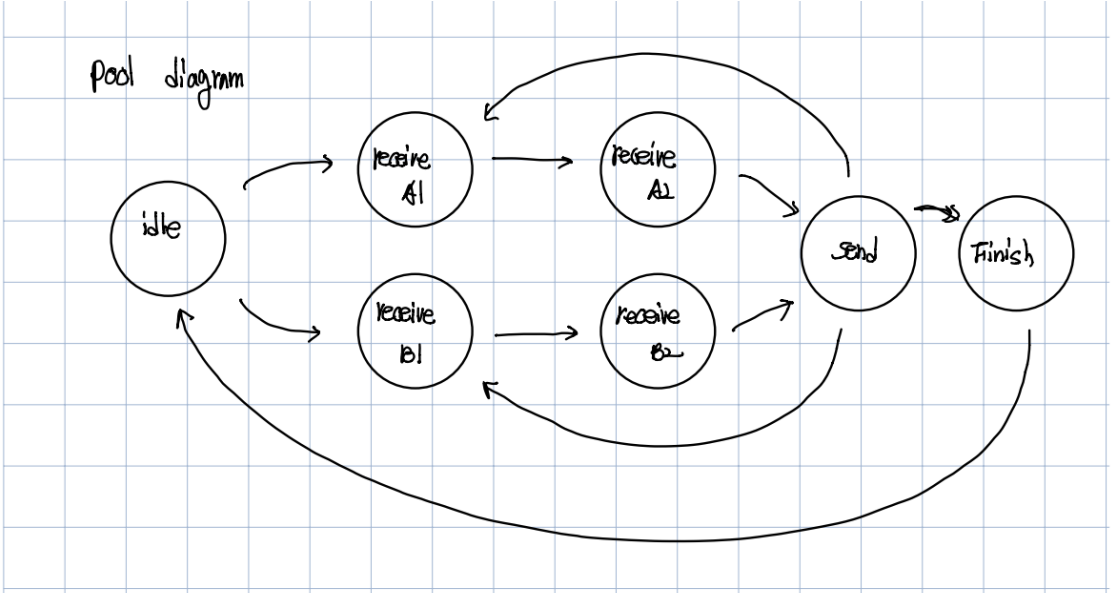
biasReceiveDone: module output으로 biasReceiveDone 신호를 출력 및 weight receive start command가 들어오기 전까지 대기. weight receive start command가 들어오면 weightReceive_0로 넘어감

weightReceive_0~3: weight 한줄을 받아 4개의 bram에 각각 저장하는 state. 동작 방식은 동일하고 weightReceive_3까지 끝나면 weightReceiveDone으로 넘어가고, 마지막 fc layer인 경우 weightReceive_1에서 바로 weightReceiveDone으로 넘어감

weightReceiveDone: module output으로 weightReceiveDone 신호를 출력, calculate 전 pe module 준비 및 fc start command가 들어오기 전까지 대기. 만약 이미 fc start command가 들어와 있는 상태라면 바로 calculate로 넘어감

calculate:

4. pool layer



state 설명

idle: start command가 들어오기 전까지 대기. command가 들어오면 featureReceive로 넘어감

featureReceive: input을 받는 state. 필요한 횟수만큼 데이터를 받으면서 bram에 저장. input을 모두 저장했다면 featureReceiveDone으로 넘어감

featureReceiveDone: module output으로 featureReceiveDone 신호를 출력 및 bias receive start command가 들어오기 전까지 대기. bias receive start command가 들어오면 biasReceive로 넘어감

biasReceive: bias를 받는 state. 동작 방식은 featureReceive와 동일. bias를 모두 저장했다면 biasReceiveDone으로 넘어감

biasReceiveDone: module output으로 biasReceiveDone 신호를 출력 및 weight receive start command가 들어오기 전까지 대기. weight receive start command가 들어오면 weightReceive_0로 넘어감

weightReceive_0~3: weight 한줄을 받아 4개의 bram에 각각 저장하는 state. 동작 방식은 동일하고 weightReceive_3까지 끝나면 weightReceiveDone으로 넘어가고, 마지막 fc layer인 경우 weightReceive_1에서 바로 weightReceiveDone으로 넘어감

weightReceiveDone: module output으로 weightReceiveDone 신호를 출력, calculate 전 pe module 준비 및 fc start command가 들어오기 전까지 대기. 만약 이미 fc start command가 들어와 있는 상태라면 바로 calculate로 넘어감

5. 추가 수정부분

scale_uart 수정: su_fc_control에서 weight recieve start 시 receive_size로 32가 아닌 받아야 하는 weight의 수 (W[“HSIZE”] * W[“VSIZE”]) 를 주도록 변경
ipynb 수정: 마지막 inference 결과 출력 시, label - 1이 아니라 label과 비교하도록 변경 (출력 결과인 max index가 1~10이 아니라 0~9로 출력되므로)