FPGAs for Monte Carlo Transport

NERSC

Colin MacLean

About Me

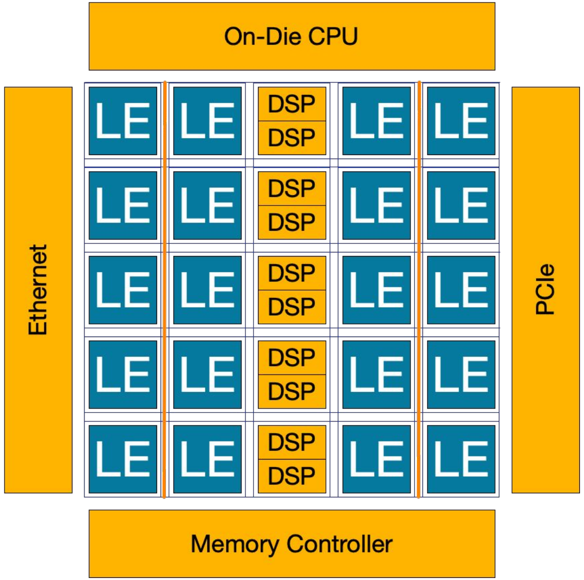
- I work in NERSC’s Advanced Technologies Group
  - ATG investigates upcoming computing technologies for NERSC’s HPC use cases
- Investigate hardware and software developments
- Currently studying FPGAs and other dataflow architectures

Meeting goals

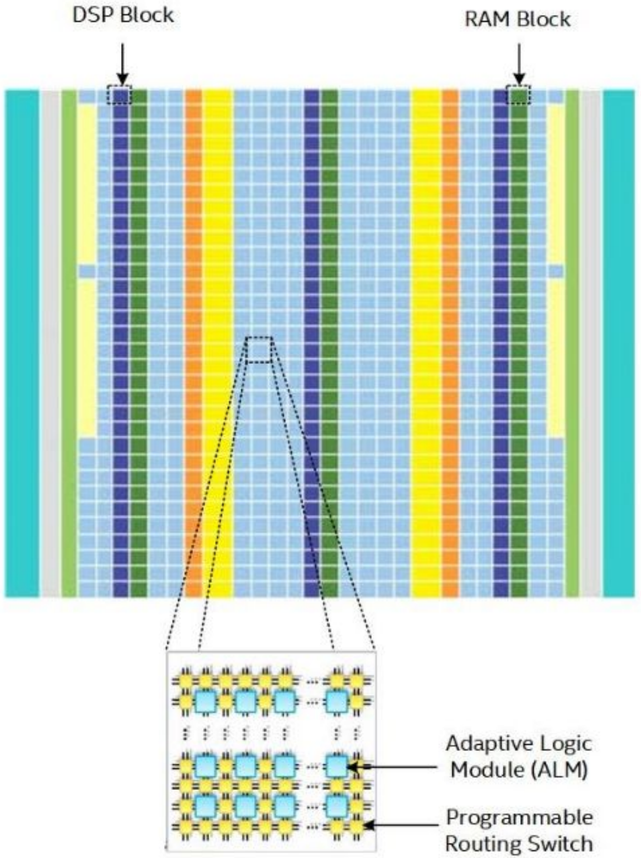
- Provide a brief, high-level description of FPGA accelerators and how they work
- Explain NERSC's Advanced Technologies Group interest and FPGA research
- Open discussion as to applicability of FPGAs for your application

What is an FPGA?

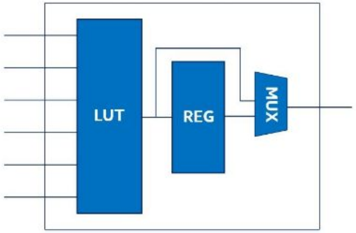
- Field Programmable Gate Arrays
- Integrated circuit designed to be configured by users
- Programs become FPGA circuits
- Lots of on-chip memory relative to CPU caches
- Ideal for pipeline parallelism



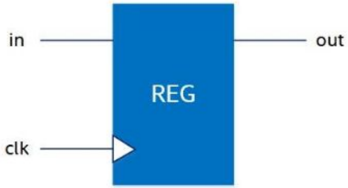
FPGA Architecture



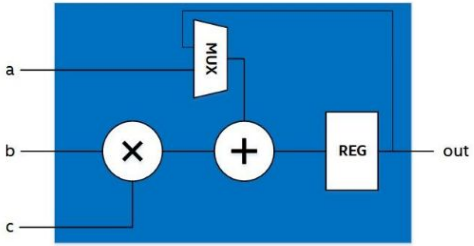
Adaptive Logic Module



Register

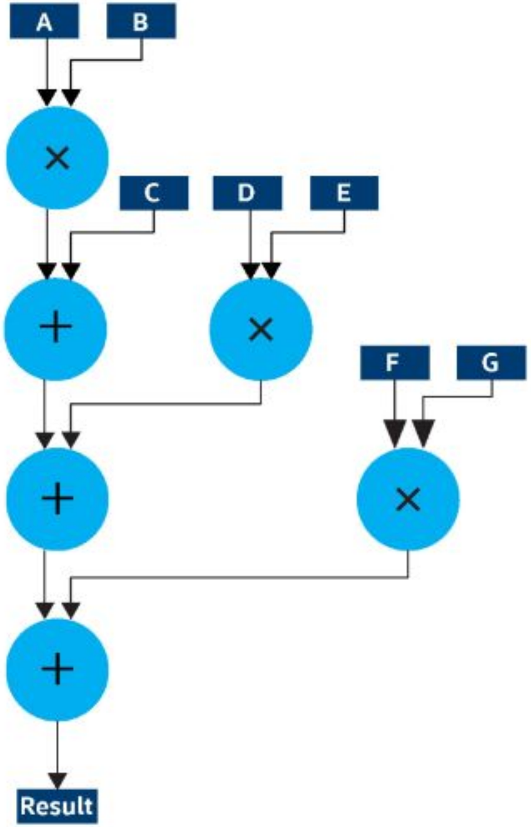


Digital Signal Processor



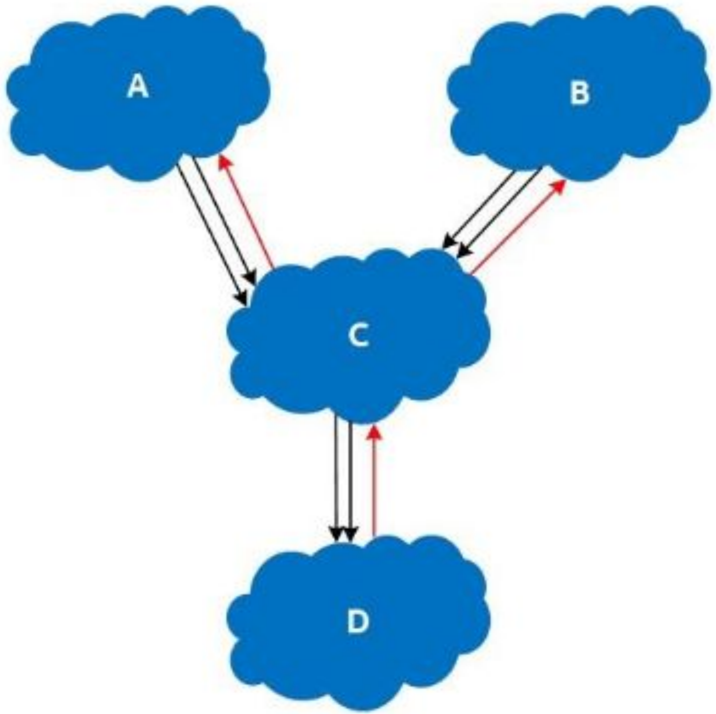
FPGA programs

- Think of FPGA programs as the compiler's instruction graph written into hardware
- Data flows through hardware, rather than instructions being executed on data
- Once a stage (row on figure to the right) is complete, the next data can start
- Ideally, initialization interval (II) is 1, meaning a new item is processed each clock cycle



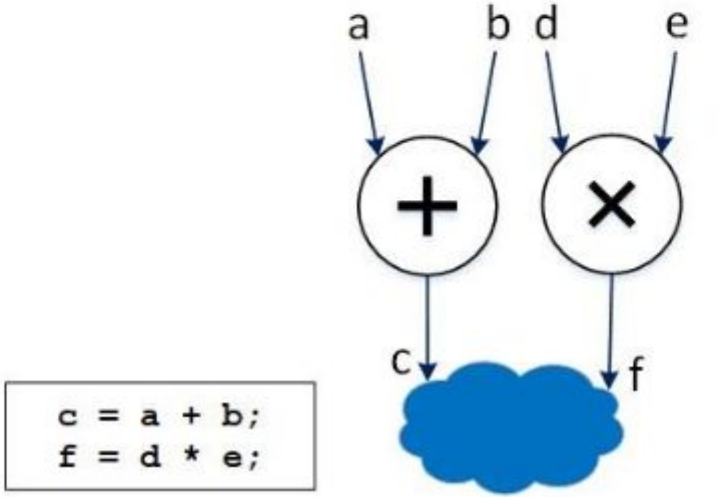
Dynamically Scheduled Logic

- Instructions connect to form kernels, which themselves can connect via pipes
- Some paths may take more clock cycles than others
- Where needed, shift registers can be used to delay output
- Pipes have a configurable depth to help reduce stalling
- If next data not available, sections can wait



FPGA “Vectorization”

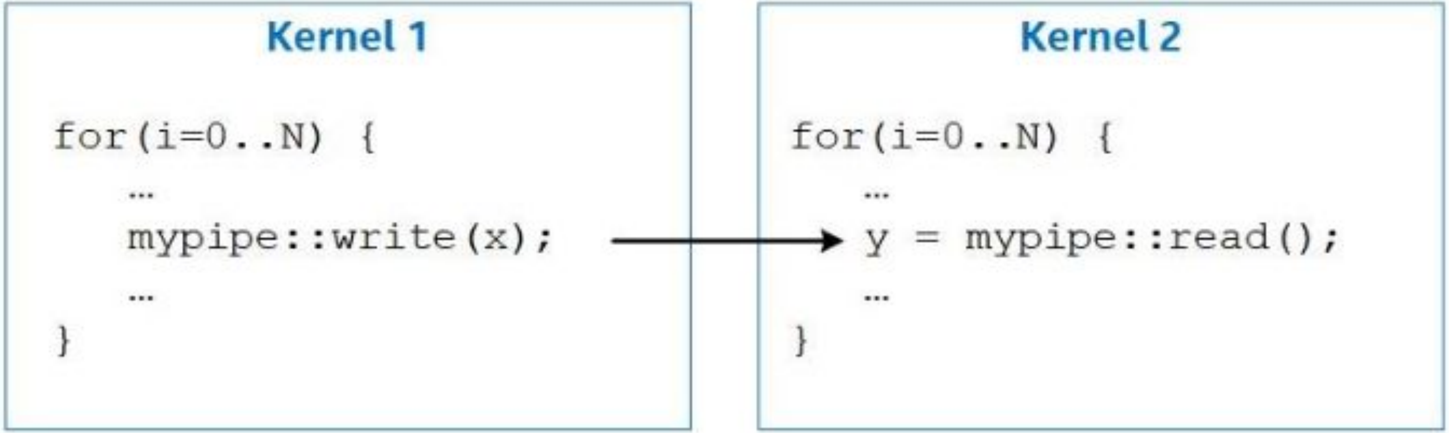
- When there is no data dependence, instructions can be mapped to hardware to occur simultaneously
- Like vectorization, but instructions executed are arbitrary



```
c = a + b;
f = d * e;
```

Task Parallelism

FPGAs are excellent for task parallelism, where multiple kernels running asynchronously can pass data back and forth via pipes and a producer/consumer model



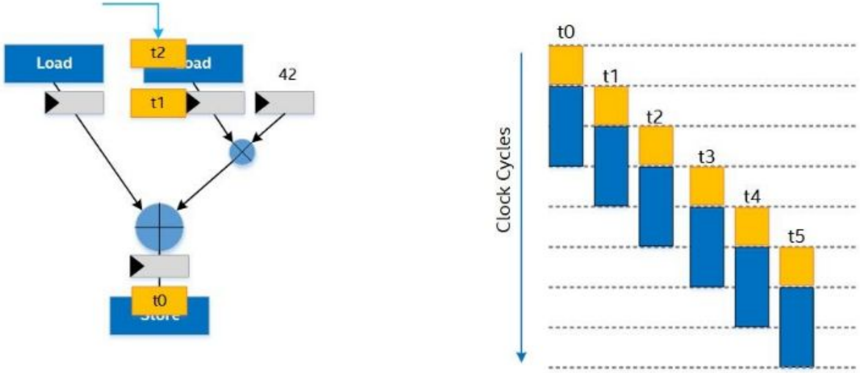
Kernel 1

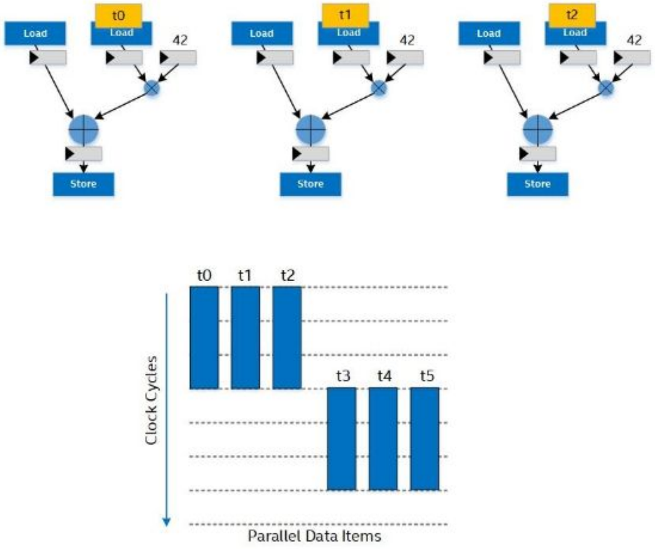
```
for(i=0..N) {
    …
    mypipe::write(x);
    …
}
```

Kernel 2

```
for(i=0..N) {
    …
    y = mypipe::read();
    …
}
```

Pipeline vs Data Parallelism



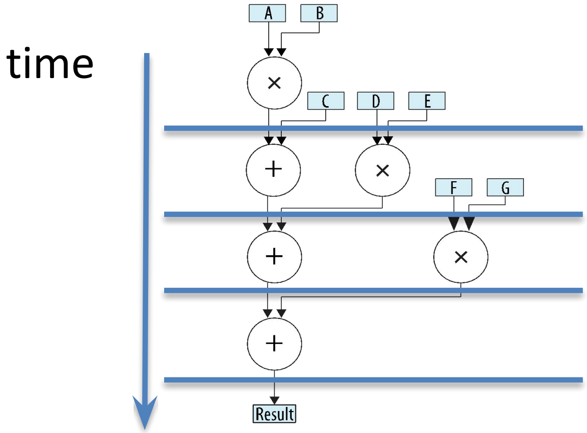


FPGAs can express both types of parallelism

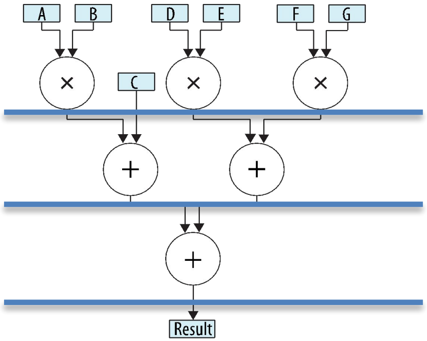
FPGA Optimization

- Single work-item kernel (circuit for a single thread)

result = (((A * B) + C) + (D * E)) + (F * G);



DFG generated and scheduled by Altera OpenCL offline compiler

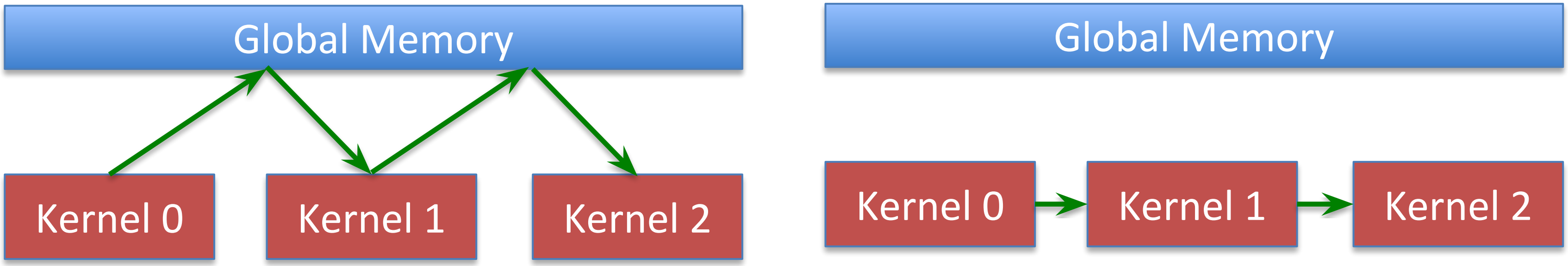


Balancing the graph with --fp-relaxed

Note: result may differ due to rounding errors

Pipelining between kernels

Kernels (circuits) can communicate via Global Mem (left) or directly via channels/pipes (right)



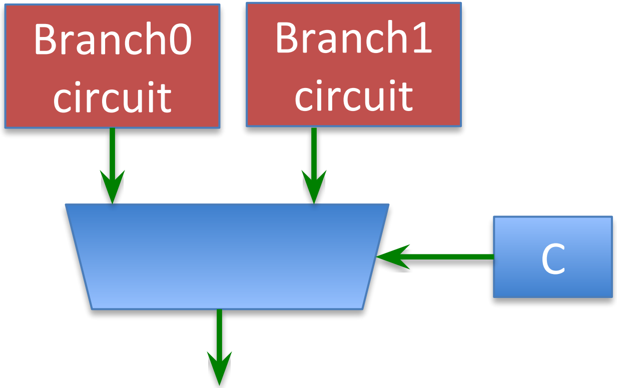
Register/BRAM FIFO

FPGA “pipes” avoid going back to global memory

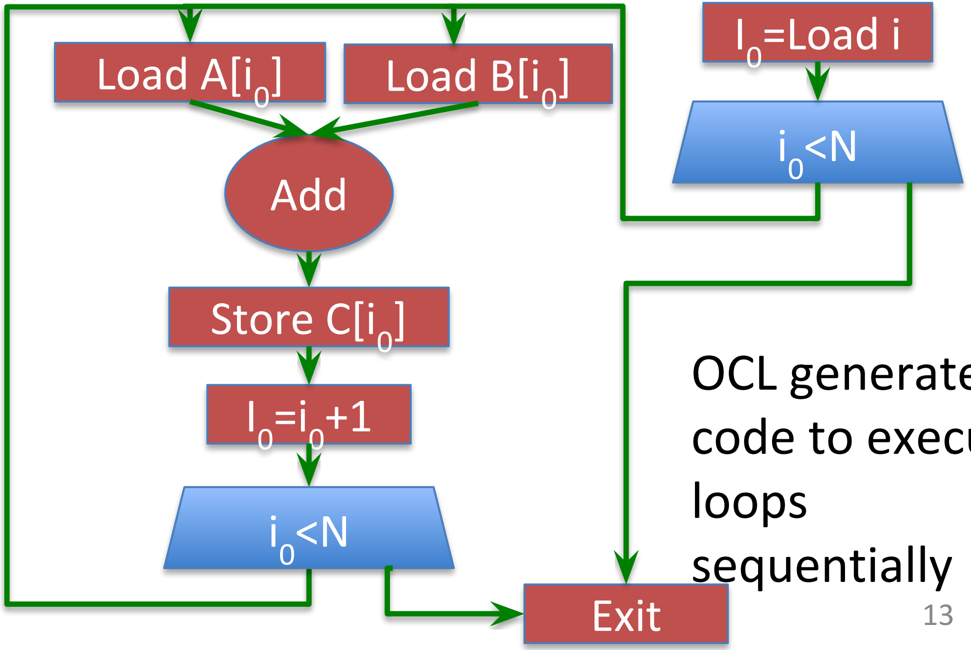
Conditional and Loop

```
if(C) {
      branch0();
}else {
      branch1();
}
```

Load both branches because there is no notion of PC



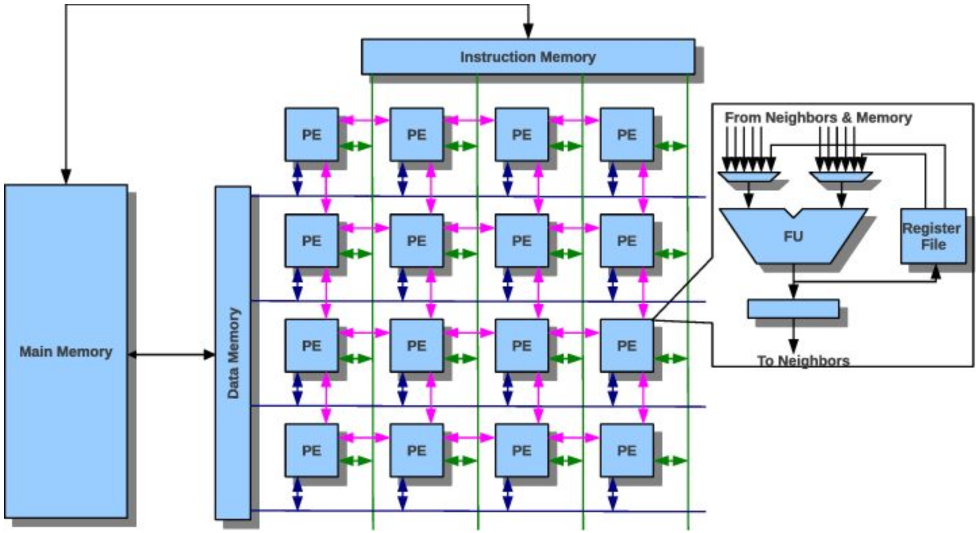
```
for(int i=0; i<N; i++){
      C[i] = A[i] + B[i]
}
```



OCL generates code to execute loops sequentially

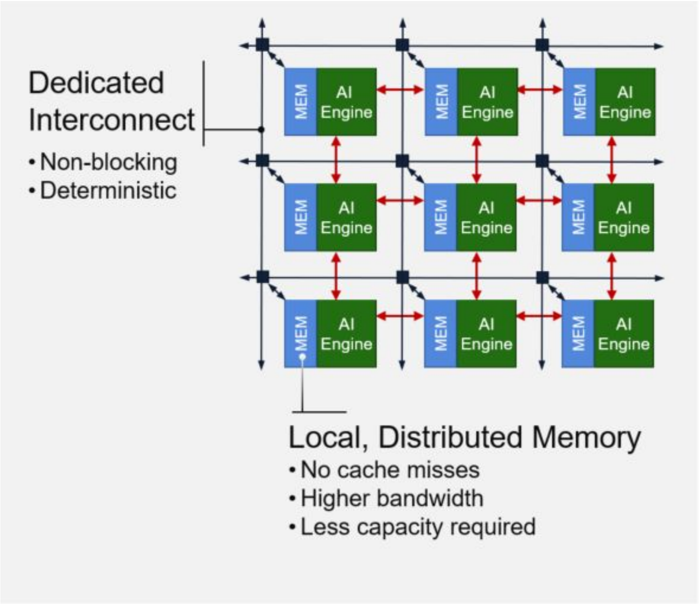
Similar Hardware

CGRAs (Coarse-grained reconfigurable architecture), like an FPGA but wide data paths and operations, not wires and gates



Systolic Arrays, such as Xilinx "AI Cores"

Elements of dataflow like simple CPU cores rather than logic gates



Dedicated Interconnect

- Non-blocking
- Deterministic

Local, Distributed Memory

- No cache misses
- Higher bandwidth
- Less capacity required

NERSC ATG Interest

- FPGAs have traditionally been programmed at a very low level by hardware engineers working with wires and gates. Higher level languages have been made available for FPGA programming recently.
- Traditionally used for low latency and long/narrow parallel workloads like signal processing, video encoding, etc.
- The NERSC Advanced Technologies Group is interested in studying the applicability of FPGAs for scientific use cases in light of recent programmability improvements.
- Moore’s Law is coming to an end
- FPGAs tend to be energy efficient, important as Dennard Scaling comes to an end
- We are looking for applications which work well on FPGAs

Monte Carlo Transport and FPGAs

- From talking with Jonathan, it sounds like FPGAs creating a FIFO queue of particles being pushed around a computational circle could work well.
- Code has a lot of branching, which has an area cost but not a time cost
- Can kernels for different types of particles, stages in the processing, etc. be extracted?