**Motivation**
Supporting precise exception for GPGPUs is very expensive.
- Too many in-flight instructions
- Problem if we use a ROB based structure
- Too many registers
- Problem if we use a check-point based mechanism

**Why Precise Exception?**
- Support debugging
  - Software support is limited
- Support context switch
- Support virtual memory
- Exception handlers
- Do we need to handle all cases?
  - Supporting virtual memory is critical

**Proposal: Guarded Execution**
Support only virtual memory using predicated execution
- Debugging support will be implemented using barrier feature
- Some of Interrupt/exceptions can be handled in a host processor
- Basic idea:
  - Marker:
    - Insert execution barrier to enforce that all the code
    - Execution barrier
  - The code after execution barrier are protected with predication (predicate bit is set by page fault check)
  - The penalty of execution barrier is not significant because the processor has multi-threading

**Basic Mechanisms**
ISA extensions
- Set start_marker
- `LD.pfchk, ST.pfchk`
  - Set pbit (page fault)
- `Sw.call.set: barrier + function calls`
  - Page miss sets predicate values
- All live-out instructions are guarded with predicate values
- Stop a `sw.call. Set`

**Re-execution Based Approach**
```c
set start_marker;
inst1
LD.pfchk.br mem[R1];
inst2;
(pbit)inst3;
(pbit)inst4;
inst5;
(pbit)(sw_call)); // call exception handler;
// re-start from the start_marker
//
// go back to the start_marker
```

**Large Arrays**
- `inst1`
  - `LD.pfchk.mem[R1];`
- `inst2;`
  - `(pbit)inst3;`
- `(pbit)inst4;`
- `inst5;`
  - `(pbit)(sw_call);` // call exception handler;

**Wait-Based Approach**
- `inst1`
  - `LD.pfchk.mem[R1];`
- `inst2;`
- `(pbit)sw_exp_call(); // call exception handler;
  - Compiler must know memory addresses at static time
  - Compiler inserts LD.pfchk speculatively to improve performance.
  - SW pipeling mechanism can also be used.

**Conclusion and Future Work**
Fine-grain context switch can be supported using predicated execution.
No hardware checkpoint mechanism is needed.
Future work: performance evaluations and detail mechanisms for global/stack operations