(a) Applications: What are the applications that motivate future systems? What advances in algorithms and programming systems will support these applications?

Single-threaded applications whose performance does not improve much even when they are run on the current multi-core architectures motivate the future systems. Programs must utilize multiple threads to exploit the parallel resources offered by modern microarchitectures to try and restore the historical trend of improvement in program performance.

As mentioned in our proposal, to support these applications, we will have to have a whole new system comprising of:

1. A New ISA Design: In order to embrace the respective strengths of the microarchitecture and the compiler, we need to redesign the interface between them, i.e. the ISA. This new ISA will allow the compiler to encode the facts it determines about code inside instructions. The microarchitecture can then use these facts to more efficiently perform parallel code transformations.

2. A Micro architectural Optimizer for Dynamic Parallelization: In concert with constructing an ISA that supports encoding of important, high level static information in instructions, we will have to design a new microarchitecture that can improve performance and adapt to user requirements by dynamic optimization of code.

3. Supporting Technology: Also, we will have to develop supporting technology such as hardware support for speculation and compiler tools to utilize the more expressive ISA.

(b) Systems: What hardware architectures and distributed systems will support future applications? What challenges do we face in design and management?

With the development of new system, future applications can be supported by all multi-core homogenous and heterogeneous architectures. The new ISA would allow for a rich, flexible representation of the program, instead of a rigid representation for a specific execution model and processor (multi-core, GPU, etc.), allowing the architecture itself to extract parallel performance. This allows binaries that are compiled in the near
term to still fully utilize future microarchitectures with newer capabilities. The new runtime optimization system would be responsible for parallelizing code for efficient execution on the underlying micro architectural configuration.

We will have to investigate several research questions pertaining to the design and management of the new system.
1. What is the best way to represent register and memory dependencies in the new ISA?
2. In addition to dependency information, what other information would be useful to the microarchitecture in extracting parallelism and performance from code?
3. Would it be beneficial for the compiler to encode hints about speculation within instructions (specifically hints about which dependencies to speculate)?
4. Would a new more restricted code layout help in facilitating more efficient runtime profiling, analysis, and optimization?

(c) Technologies: What emerging technologies will change fundamental assumptions in hardware and software? What constraints disappear? What challenges arise?

The assumption that single-threaded applications do not perform well on different multi-core architectures will be completely changed by the emerging technologies.

Constraints on programmers who are forced to write parallel programs to fully utilize multiple threads offered by the hardware will disappear. Also, constraints that different parallel versions of the programs need to exist if they have to run on different microarchitectures will disappear.

We will have a whole new challenge of designing the new ISA, new dynamic optimizer and the supporting technology.

(d) Methodologies: How should we perform interdisciplinary research that spans applications, systems, and technologies? How should abstraction layers evolve?
We need to focus on three different aspects: the full ISA specification, modification of the processor pipeline to support a dynamic optimizer, and a multi-threaded transactional memory system to allow our architecture to perform speculative code transformations. In the longer run, we need to extend our proposals to support speculative code transformations. The development of the Liberty Architecture consists of four milestones:

Instruction Set Architecture: Initial design exploration has begun on the ISA, focusing on the use of SSA form. We will investigate the design choices that best allow the representation of dependency information to the underlying architecture.

Non-Speculative Code Transformations: In addition to DOALL, DOACROSS, and DSWP transformations, we will also consider other transformations to increase single-threaded performance, such as traditional software pipelining. This milestone consists of a final selection of supported code transformations, and building them in our dynamic optimizer.

Speculation Support: ISA support and modifications to the memory subsystem are required to support a low-overhead multi-threaded transactional memory system and enable speculative DOALL, DOACROSS, and DSWP transformations. Initial designs have been proposed, but we need to implement and evaluate a transactional memory system, which will support all our target code transformations.

Speculative Code Transformations: With efficient hardware multi-threaded transactional memory support in place, we will extend our dynamic optimizer to efficiently gather and use dependence profiling information. Through a combination of these extensions, we will enable various speculative transformations to better utilize the underlying hardware configuration.

(e) Risks: What are the risks that threaten the success of XPS research directions? How do we guard and hedge against these threats?

Restricting research to particular architectures could be a possible risk. We need to guard against this by being flexible enough for future architectures. These future architectures could look wildly different than
what exists today, for example different compute elements with different capabilities. The simplest current example is GPU versus CPU core. But we expect future architectures to have very specialized elements such as programmable fabrics (FPGAs) etc. We need to make sure not to restrict current efforts to what exists today, which is one reason why we are representing programs in such a flexible ISA and allowing a dynamic optimizer for each individual architecture to optimize the code for its specific microarchitecture. We also need to focus on restoring the earlier abstraction between the microarchitecture and the user.