Fang, Lu, and She approached the weaknesses of a multicore processor through a network-on-chip (NoC) system that works with the existing clock mechanism of the processor. A NoC is the central link for all cores in a processor and it’s power consumption is dominated by link and router power. They focused on mitigating power consumption and execution time by reducing the number of routers using a strategy dubbed PRCIES. The most common topologies for component connections are crossbar, mesh, folder, and torus, PRCIES is based on a mesh configuration. The main metrics used to evaluate their success were execution time, network latency and most significantly, power consumption. PRCIES consists of creating a mesh of routers, each of which connects four cores where inter-core interactions are managed by idle cores. It dynamically allocates virtual channels for the minimum unit of data(flit) and obtains the location of the next node by querying the NoC routing table. In the process of writing outputs and transferring data from a buffer to a switch, the system automatically searches for the first input to obtain the virtual network that contains the data awaiting transfer. In addition to the architectural modifications, PRCIES also deviates in terms of router policy. The policy modifications include, setting a counter for every input in the router to regularly monitor the amount of data sent to the router and prioritizing the input unit that carry’s the most data to be placed at the front of the query while simultaneously clearing and resetting the counter. If the required virtual channel is occupied, a conflict is produced and a search is conducted for the nearest idle virtual channel. In this policy, every input unit has the opportunity to acquire priority and obtain an output port[4].

Another group of researchers; Zhao,Cui,Xue and Feng tested a configuration that was composed of two different quad-core platforms. They demonstrated how two different cache-sharing policies affected performance interference with a synthesized STREAM kernel called “Interference”. Interference is designed to periodically generate private L2 cache miss and issue a shared L3 cache request in individual executions. When the processors are co-located and L3 is being contended for interference is expected[2]. They circumvent this issue in three principles as follows. 1.) the amount of memory required is greater than per-core private cache size but not shared cache size, ensuring that data missed in L2 can be found in L3. 2.)The memory that Interference requires is accessed to periodically generate L2 cache misses; at the same time random access invalidates the hardware pre-fetcher. 3.)The aforementioned access pattern is repeated enough times to ensure the kernel runs for a long enough time to exhibit stable behavior.

Ideally to improve the multicore processor, one must have a specific application in mind. In the absence of a predefined function there are still characteristics that would benefit a multicore processor on a general scale. Our team would propose that a multicore processor should utilize the composed processor of simple cores, with a mesh-router topology and implement an Interference type strategy for shared cache resources. This configuration would accelerate both serial and parallel threads in addition to a high level of reliability. It would also conserve power consumption as a result of the reduced router count and scheduling protocols.