US 20050050310A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2005/0050310 A1**

Bailey et al. (43) **Pub. Date: Mar. 3, 2005**

(54) METHOD, SYSTEM, AND APPARATUS FOR IMPROVING MULTI-CORE PROCESSOR PERFORMANCE

(76) Inventors: **Daniel W. Bailey**, Austin, TX (US); **Todd Dutton**, Southborough, MA (US); **Tryggve Fossum**, Northborough, MA (US)

Correspondence Address:
BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030 (US)

(21) Appl. No.: **10/621,228**

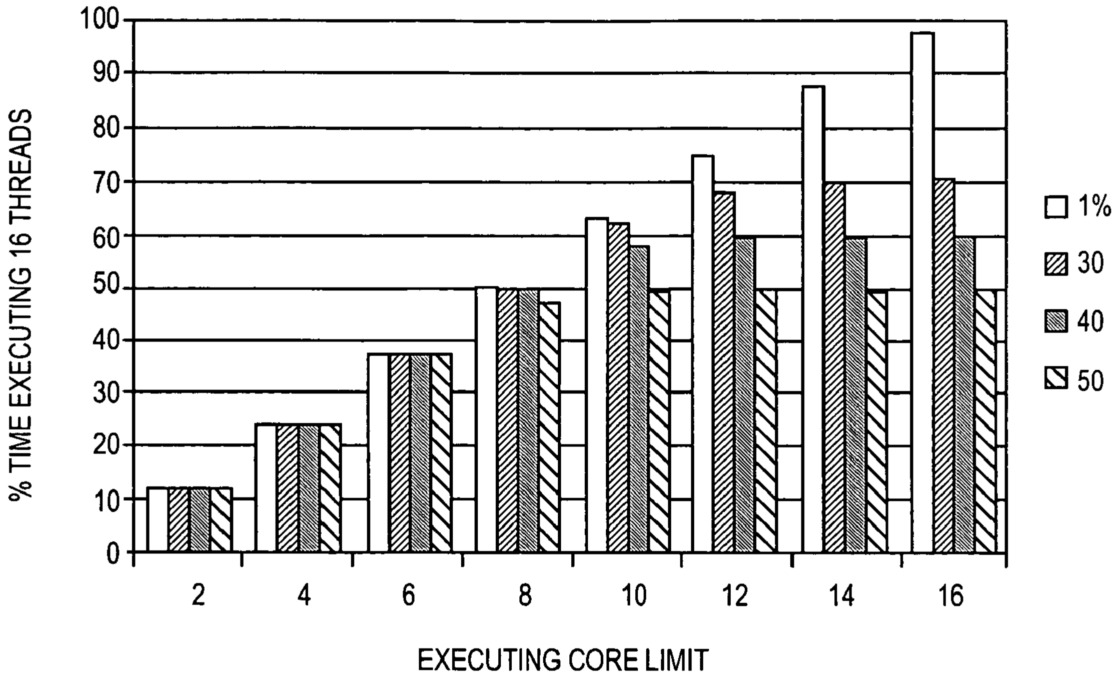
(22) Filed: **Jul. 15, 2003**

Publication Classification

(51) **Int. Cl.**^{7} **G06F 9/00**
(52) **U.S. Cl.** **713/1**

(57) **ABSTRACT**

A system, apparatus, and method for a core rationing logic to enable cores of a multi-core processor to adhere to various power and thermal constraints.



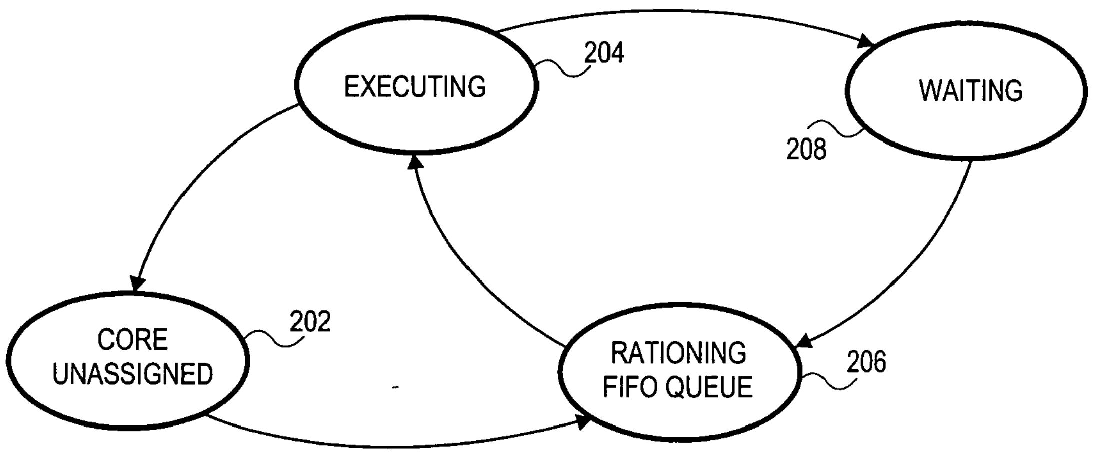

EXECUTING
204
WAITING
208
CORE UNASSIGNED
202
RATIONING FIFO QUEUE
206

FIG. 1

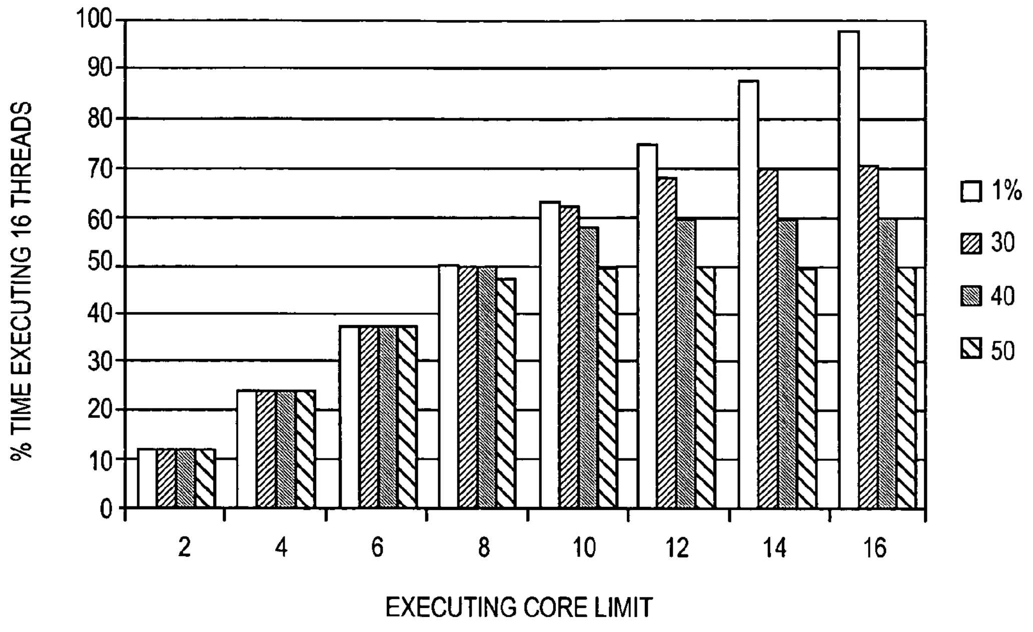

100
90
80
70
60
50
40
30
20
10
0
% TIME EXECUTING 16 THREADS
2
4
6
8
10
12
14
16
EXECUTING CORE LIMIT
1%
30
40
50

FIG. 2

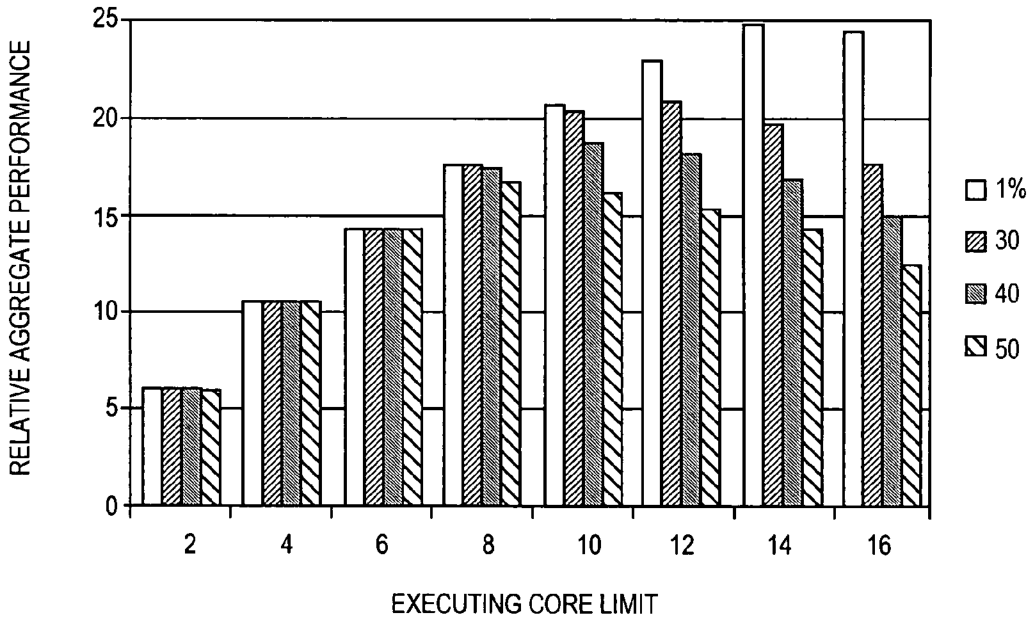

25
20
15
10
5
0
RELATIVE AGGREGATE PERFORMANCE
2
4
6
8
10
12
14
16
EXECUTING CORE LIMIT
1%
30
40
50

FIG. 3

CORE/RING STRUCTURE

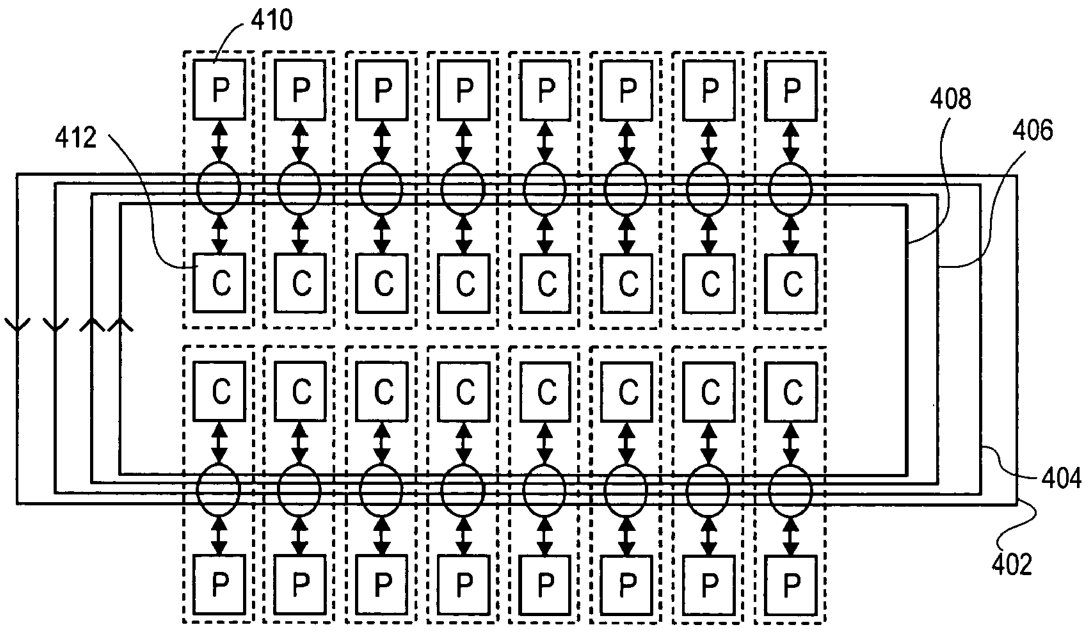


FIG. 4

METHOD, SYSTEM, AND APPARATUS FOR IMPROVING MULTI-CORE PROCESSOR PERFORMANCE

BACKGROUND

[0001] 1. Field

[0002] The present disclosure pertains to the field of power management. More particularly, the present disclosure pertains to a new method and apparatus for improving multi-core processor performance despite power constraints.

[0003] 2. Description of Related Art

[0004] Power management schemes allow for reducing power consumption to achieve low power applications for various types of and systems and integrated devices, such as, servers, laptops, processors and desktops. Typically, software methods are employed for systems and integrated devices to support multiple power states for optimizing performance based at least in part on the Central Processing Unit (CPU) activity.

[0005] Present power management schemes either decrease voltage or frequency or both for reducing power consumption. However, this results in decreased overall performance. Also, some methods incorporate analog designs that have various challenges relating to loop stability for transient workloads, calibration, and tuning.

[0006] With the introduction of processors with multiple cores, power management becomes a major concern because of the increase in cores operating at high frequencies and voltages and need to adhere to various power constraints, such as, thermal limits, maximum current, and Vcc range.

BRIEF DESCRIPTION OF THE FIGURES

[0007] The present invention is illustrated by way of example and not limitation in the Figures of the accompanying drawings.

[0008] **FIG. 1** illustrates a flowchart for a method utilized in accordance with an embodiment

[0009] **FIG. 2** illustrates a bar chart utilized in accordance with an embodiment.

[0010] **FIG. 3** illustrates a bar chart utilized in accordance with an embodiment.

[0011] **FIG. 4** illustrates an apparatus in accordance with one embodiment.

DETAILED DESCRIPTION

[0012] The following description provides method and apparatus for improved multi-core processor performance despite power constraints. In the following description, numerous specific details are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate logic circuits without undue experimentation.

[0013] As previously described, a problem exists for improving processor performance while adhering to power constraints. The present methods incorporate lowering the voltage or frequency at the expense of overall performance. In contrast, the claimed subject matter improves overall performance while adhering to power constraints. For example, a concept of "rationing the number of executing cores for a processor system" allows for increasing frequency as a result of disabling clocks to cores that are idle as they wait for a memory transaction to complete. For example, the claimed subject matter exploits the idle time period of processor cores by disabling the clocks to the core, that results in less power dissipation. Thus, a higher frequency can be utilized as a result of the decrease in power dissipation. In one embodiment, an appropriate executing core limit is calculated for the workload. Also, in the same embodiment, the number of executing cores are less than or equal to the number of available and ready threads. A thread is an independent set of instructions for a particular application.

[0014] In one embodiment, the claimed subject matter facilitates selecting a voltage/frequency operating point based on a prediction of the activity level of the threads running on all of the cores collectively. For example, TPC-C threads tend to be active 50-60% of the time, and spend 40-50% of their time idle, waiting for memory references to be completed. In such an environment, one would specify an executing core limit that would be, in one embodiment, 60% of the total number of cores on the die; if there were 8 cores, one would set the executing core limit to, in this case, five. One would then specify a voltage-frequency operating point that corresponds to having only five cores active and three cores inactive (low power state) at a time; this is a significantly higher operating frequency than one would specify if one was allowing all eight cores to be simultaneously active. The core rationing logic constrains the operations of the die, guaranteeing that no more than five cores (in this case) are active at any given moment. Statistics are gathered regarding the occupancy of the Waiting and Rationing queues (which will be discussed further in connection with **FIG. 1**); at intervals these statistics are analyzed to determine whether the operating point (executing core limit and its associated voltage/frequency pair) should be changed. If the Waiting queue tends to be empty and the Rationing queue tends to be full, that is an indication that cores are not making progress when they could be, and that to improve performance the executing core limit should be raised and the voltage/frequency reduced; conversely, if the Rationing queue tends to be empty, and the Waiting queue tends to be full, this may be an indication that one can increase performance by reducing the executing core limit and increasing the voltage/frequency point.

[0015] **FIG. 1** illustrates a flowchart for a method utilized in accordance with an embodiment. In one embodiment, the flowchart depicts a method for a state diagram.

[0016] In the same embodiment, the state diagram illustrates a predetermined state machine for a processor core in a system. In this same embodiment, the state machine facilitates the "rationing of the cores" to improve processor performance as a result of disabling clocks to cores that are waiting for a memory transaction to complete.

[0017] In one embodiment, the state diagram has four defined states, such as, a Core Unassigned state **202**, an Executing state **204**, a Rationing FIFO Queue state **206**, and a Waiting state **208**. Initially, the Core Unassigned state is defined as follows: each core does not have an assigned thread. Subsequently, in the event that a core has a thread assigned to it, the claimed subject matter transitions to the Rationing FIFO Queue state **206**. In one embodiment, FIFO is defined as a First In First Out.

[**0018**] Upon transitioning to the Rationing FIFO Queue state, a comparison between the number of executing cores and an executing core limit (ECL) is determined. In one embodiment, a processor or system specification determines the proper executing core limit in order to adhere to thermal power considerations. In one embodiment, the ECL is determined by a formula depicted later in the application. If the number of executing cores is less than ECL, the particular core transitions to the Executing state **204** if the core was the next one to be processed in the FIFO queue. Otherwise, the core remains in the Rationing FIFO queue **206**.

[**0019**] Upon entering the Executing state, the core remains in this state unless an event occurs, such as, a memory reference and overheating event, and/or a fairness timeout. For example, a fairness timeout may be utilized to prevent a possible live lock state. In this context, a memory reference refers to a read or write operation to a particular memory address that does not reside in any cache memory coupled to the processor ("a miss in all levels of cache memory"). Therefore, an access to main memory is initiated.

[**0020**] If an event occurs as previously described, the core transitions to the Waiting state **208**. Upon completion of the event, the core transitions to the Rationing FIFO queue state. This sequence of cycling between states **204**, **206**, and **208** occurs until the particular thread is completed. Upon completion of the thread, the core transitions to the Core Unassigned State.

[**0021**] However, the claimed subject matter is not limited to the four defined states in the state diagram. The claimed subject matter supports different amounts of states. **FIG. 1** merely illustrates an example of limiting the number of executing cores to be less than the available number of threads. For example, one embodiment would allow for multiple waiting states. Alternatively, the waiting states could be replaced by another queue state. Also, other embodiments of state diagrams would allow multiple priority levels for cores, as well as having different waiting queues depending on the nature of the event that provoked exit from the executing state (memory wait, thermal wait, ACPI wait, etc).

[**0022**] Typically, a core executes a memory read or write operation and subsequently executes an operation that is dependent on that operation (for example, it makes use of the data returned by a memory read operation). Subsequently, the core "stalls" waiting for that memory operation to be completed. In such a case, it asserts a signal to the central core rationing logic indicating that it is stalled; this is the indication that it is eligible to be disabled by the core rationing logic. The core rationing logic responds to this signal by "napping" the core in question—it asserts a "nap" signal to the core, which causes the core to block instruction issue and then transition into a (cache-coherent) low power state. Furthermore, the core rationing logic puts an identifier for that core in the Waiting queue. When the memory operation completes, the core deasserts the "stall" signal; the core rationing logic responds to this by moving the identifier for that core from the Waiting queue to the Rationing queue. If the number of currently executing (not "napped") cores is less than or equal to the Executing Core Limit, the core rationing logic removes the oldest identifier from the Rationing queue, and deasserts the "nap" signal to that core.

[**0023**] **FIG. 2** illustrates a bar chart utilized in accordance with an embodiment. In one embodiment, the bar chart depicts a percentage time spent executing for a 16-core multiprocessor as calculated by a Monte Carlo simulation for a variety of workloads. The independent axis illustrates the ECL for 2, 4, 6, 8, 10, 12, 14, and 16. Also, there is a bar for each ECL at a different workload as simulated with a memory reference duty cycle (with respect to executing time) of 1%, 30%, 40%, and 50%.

[**0024**] Analyzing the 50% memory reference duty cycle highlights the fact that the percentage time executing saturates at 50%. Thus, processing the memory references consumes half of the executing time when the ECL is equal to the number of available threads.

[**0025**] **FIG. 3** illustrates a bar chart utilized in accordance with an embodiment. In addition to **FIG. 2**, **FIG. 3** illustrates the total performance as calculated by the product of the percentage time executing and the frequency. The total performance also incorporates the fact that frequency is inversely proportional to the ECL. As previously described, this relationship exists because as one reduces the number of executing cores, this results in reducing power dissipation. Therefore, the frequency can be increased to remain at the steady-state thermal limit.

[**0026**] Also, **FIG. 3** depicts the maximum percentage time executing is 70% for the 30% memory reference duty cycle. Also, the product of the saturation limit and the number of threads demarcates the onset of saturation. Of particular note is the onset of saturation because this may be the region for improved or optimum performance.

[**0027**] In one embodiment, a self optimization formula is utilized to determine the appropriate ECL. In the formula, N depicts the number of threads that have context: % E depicts the percentage executing time; and %M depicts the percentage memory reference time. The formula is:

$$int\,(N\times(\%\,E/\,(\%\,E+\%M)))$$

[**0028**] **FIG. 4** depicts an apparatus in accordance with one embodiment. In one embodiment, the apparatus depicts a multi-core processor system with a plurality of processors **410** coupled individually to an independent bank of Level 3 (L3) Cache memory. In the same embodiment, a plurality of four busses form two counter rotating "rings"—a Request/Response (REQ0/RSP0) ring (**402** and **404**) in the clockwise direction, and a Request/Response ring (REQ1/RSP1) (**406** and **408**) in the counterclockwise direction. The circle in between the "P"s and the "C"s represents a pair of state devices for each ring. Thus, a set of circular pipelines are utilized for passing information from each processor core/cache bank to any other processor core/cache bank. The system interface logic contains the memory controllers for memory DIMMs, the router logic to handle the interconnection links to other processor dies and/or I/O subsystems, and assorted other system control logic (including the central core rationing controller).

[**0029**] While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure.

What is claimed is:

1. A method for disabling clocks to at least one processor core of a plurality of processor cores comprising:

calculating an executing core limit based at least in part on a workload;

executing an n number of available threads, wherein n is an integer,

enabling an m number of processor cores, wherein m is an integer and is less than or equal to n, the number of available threads.

2. The method of claim 1 wherein disabling clocks to at least one processor during an idle time period as the processor core waits for a memory operation.

3. The method of claim 1 wherein disabling clocks to at least one processor core results in decreased power consumption.

4. The method of claim 1 wherein disabling clocks to at least one processor core allows for increasing the operating frequency of that processor core.

5. A method for selecting a voltage and frequency operating point to at least one processor core of a plurality of processor cores comprising:

predicting an activity level of a plurality of threads running on all of the plurality of processor cores;

enabling a subset of the plurality of processor cores based at least in part on the activity level.

6. The method of claim 1 wherein the activity level is an executing core limit that is based at least in part on adhering to thermal power considerations.

7. The method of claim 6 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; % E depicts the percentage executing time; and % M depicts the percentage memory reference time. and the formula is:

$$int\ (N\times(\%\ E/\ (\%\ E+\%\ M)))$$

8. A state diagram for a plurality of multi-core processors comprising:

A first state for a core without an assigned thread;

A second state for a queue to store cores with an assigned thread;

A third state for enabling the core to run a thread; and

A fourth state to disable a core.

9. The state diagram of claim 8 wherein the queue is a first in first out (FIFO) queue.

10. The state diagram of claim 8 wherein the core transitions from a second state to the third state if the number of enabled cores is less than an executing core limit.

11. The state diagram of claim 10 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; % E depicts the percentage executing time; and % M depicts the percentage memory reference time. and the formula is:

$$int\ (N\times(\%\ E/\ (\%\ E+\%M)))$$

12. The state diagram of claim 8 wherein the core transitions from a third state to the fourth state if the core is idle as it waits for completion of a memory operation.

13. A method for a state diagram for a plurality of multi-core processors comprising:

assigning a first state to a core without an assigned thread;

assigning a second state for a queue to store cores with an assigned thread;

comparing the number of enabled cores to an executing core limit, assigning a third state for enabling the core to run a thread if the number of enabled cores is less than the executing core limit; and

assigning a fourth state to disable a core.

14. The method of claim 13 wherein the queue is a first in first out (FIFO) queue.

15. The method of claim 13 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; % E depicts the percentage executing time; and % M depicts the percentage memory reference time. and the formula is:

$$int\ (N\times(\%\ E/\ (\%\ E+\%\ M)))$$

16. The state diagram of claim 13 wherein the core transitions from a third state to the fourth state if the core is idle as it waits for completion of a memory operation.

17. A system of multi-core processors comprising:

at least one multi-core processor coupled to a cache memory, and coupled to at least two clockwise directional busses to receive requests and responses; and

a core rationing logic to manage the number of enabled cores to be less than or equal to an executing core limit.

18. The system of claim 17 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; % E depicts the percentage executing time; and % M depicts the percentage memory reference time. and the formula is:

$$int\ (N\times(\%\ E/\ (\%\ E+\%\ M))).$$

19. The system of claim 17 further comprises a system interface that contains:

a plurality of memory controllers for memory DIMMs;

a router logic to handle the interconnection links to other processor dies or I/O subsystems; and

the core rationing logic.

20. The system of claim 17 further comprises at least two counter- clockwise directional busses to receive requests and responses.

21. The system of claim 17 wherein the cache memory is a level three (L3) memory with a plurality of independent memory banks.

* * * * *