



(51) International Patent Classification:

G06F 9/48 (2006.01) G06F 1/20 (2006.01)
G06F 1/32 (2006.01)

(21) International Application Number:

PCT/SE2009/050594

(22) International Filing Date:

26 May 2009 (26.05.2009)

(25) Filing Language:

English

(26) Publication Language:

English

(71) Applicant (for all designated States except US): TELEFONAKTIEBOLAGET LM ERICSSON (publ)
[SE/SE]; SE-164 83 Stockholm (SE).

(72) Inventor; and

(75) Inventor/Applicant (for US only): VAJDA, Andras
[RO/FI]; Hulluksentie 3C12, FI-02430 Masala (FI).

(74) Agent: VALEA AB; Box 7086, SE-103 87 Stockholm (SE).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ,

EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))
- of inventorship (Rule 4.17(iv))

Published:

- with international search report (Art. 21(3))

(54) Title: METHOD AND SCHEDULER IN AN OPERATING SYSTEM

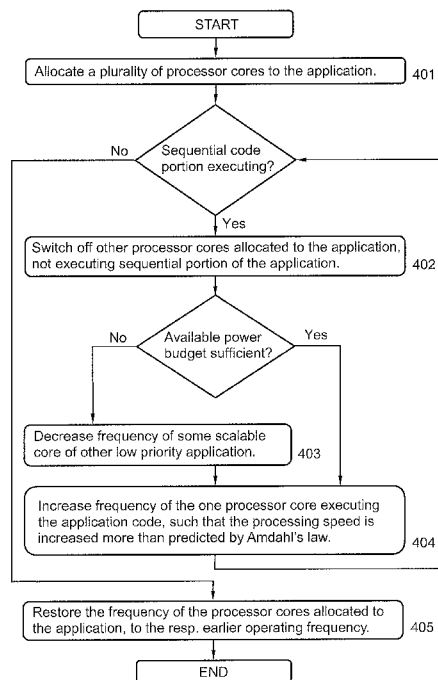


Fig. 4

(57) Abstract: Method and scheduler in an operating system, for scheduling processing resources on a multi-core chip. The multi-core chip comprises a plurality of processor cores. The operating system is configured to schedule processing resources to an application to be executed on the multi-core chip. The method comprises allocating a plurality of processor cores to the application. Also, the method comprises switching off another processor core allocated to the application, not executing the sequential portion of the application, when a sequential portion of the application is executing on only one processor core. In addition, the method comprises increasing the frequency of the one processor core executing the application to the second frequency, such that the processing speed is increased more than predicted by Amdahl's law.

METHOD AND SCHEDULER IN AN OPERATING SYSTEM

TECHNICAL FIELD

The present invention relates to a method and a scheduler in an operating system and,
5 more in particular, to a mechanism for scheduling processing resources on a multi-core chip.

BACKGROUND

The computing industry's development over the last decades has been largely defined by
10 a dramatic increase of the available computing power, driven primarily by the self-fulfilment of Moore's law. However, recently this development has been heavily impacted by the inability of scaling performance further through increased frequency and complexity of processor devices, due primarily to power and cooling issues; as an answer, the industry turned to Chip-MultiProcessors (CMP) or multi-core devices. It is predicted that
15 the advancement of manufacturing technology, in fact, the continued development according to Moore's law, will allow for chips comprising 100s or even 1000s of processor cores.

Currently operating system schedulers work on the principle of assigning tasks to
20 available hardware resources such as processor cores or hardware threads and sharing a core among multiple applications.

Current approaches have a number of limitations. They do not scale well to high number of cores within the same chip. Micro-managing each core does not pay off well and may
25 not be economical for simple cores. Further, current solutions do not consider dynamic variations in the processing requirements of applications. An application may be at different point in time executing in single threaded mode, while at other moments may require multiple threads of execution.

30 Chip multiprocessors pose however new requirements on operating systems and applications. It appears that current symmetric multi-processing, time-shared operating systems will not scale to several tens, less to hundreds or thousands of processor cores. On the application side, Amdahl's law, even it's version for chip multi-processors, will put a limit to the amount of performance increase that can be achieved even for highly
35 parallelized applications, assuming static symmetric or static asymmetric multi-core chips

as illustrated in **Figure 1**. For example, an application with only 1% sequential code can achieve, according to Amdahl's law, a theoretical speed-up factor of 7.4 on 8 cores, 14 on 16 cores, but only 72 on 256 cores and 91 on 1024 cores, compared with running on a single core.

5

Amdahl's law for CMP is expressed as:

$$Speedup_{dynamic}(f, n, r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{n}}$$

Where f is the fraction of the code that can execute in parallel, n is the number of processor cores, r is the number of processor cores that can be used jointly to execute sequential portions and $perf(r)$ is the resulting performance, assumed to be less than r .

Better scalability, approaching linear for highly parallelized applications, could theoretically be obtained with chips where computing resources could be utilized either as several cores executing in parallel, for the execution of the parallel sections of the applications; or as a single, fast, powerful core for running the single-threaded, sequential portions of the applications. However, so far there is no solution that would scale with the number of cores.

Thus there is a problem to increase the processing speed of application code in a computer.

SUMMARY

It is the object to obviate at least some of the above disadvantages and provide an improved performance of applications executing on a multi-core system.

25

According to a first aspect, the object is achieved by a method in an operating system for scheduling processing resources on a multi-core chip. The multi-core chip comprises a plurality of processor cores. The operating system is configured to schedule processing resources to an application to be executed on the multi-core chip. The method comprises allocating a plurality of processor cores to the application. Also, the method comprises switching off processor cores allocated to the application, not executing the sequential portion of the application, when a sequential portion of the application is executing on only one processor core. In addition, the method comprises increasing the frequency of the

one processor core executing the application, such that the processing speed is increased more than predicted by Amdahl's law.

According to a second aspect, the object is also achieved by a scheduler in an operating
5 system for scheduling processing resources on a multi-core chip. The processing resources are allocated to an application to be executed on the multi-core chip. The multi-core chip comprises a plurality of processor cores. The scheduler comprises an allocation unit. The allocation unit is adapted to allocate a plurality of processor cores to the application. Also, the scheduler comprises a switching unit. The switching unit is adapted
10 to switch off a processor core, allocated to the application, not executing a sequential portion of the application. Further, the scheduler comprises an increasing unit. The increasing unit is adapted to increase the frequency of the one processor core executing the application, such that the processing speed is increased more than predicted by Amdahl's law.

15

The present solution is based on a space shared approach for scheduling multiple processes on a many core chip. Also, the present solution rely on adapting performance of individual cores and the chip as a whole by controlled and scheduled variation of the frequency at which different cores execute, in contrast with current thread migration
20 strategies. Instead of trying to guess the best thread assignment policy in the operating system rely on application generated requests for computing resources, similarly to how memory is currently managed. Thereby is a near-linear scaling of speed with the number of processing cores achieved. Further, the present solution addresses the issue of dynamically adapting processing needs to the requirements of applications. In addition, as
25 the present solution is not relying on thread migration, hence improving cache usage characteristics. Thereby is an improved performance within a wireless communication system achieved.

Other objects, advantages and novel features of the invention will become apparent from the
30 following detailed description of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described more in detail in relation to the enclosed drawings, in which:

35

Figure 1 is a diagram illustrating Speed-up according to Amdahl's law.

Figure 2 is a schematic block diagram illustrating a system according to some embodiments.

5

Figure 3 is a schematic block diagram illustrating a Multi-core chip according to some embodiments.

Figure 4 is a flow chart illustrating embodiments of method steps in an operating system.

10

Figure 5 is a block diagram illustrating embodiments of a scheduler in an operating system.

15 Figure 6 is a diagram illustrating Speed-up according to some embodiments of the present method.

DETAILED DESCRIPTION

The invention is defined as a method and a scheduler in an operating system, which may be put into practice in the embodiments described below. This invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art. It should be understood that there is no intent to limit the present method and/or scheduler to any of the particular forms disclosed, but on the contrary, the present methods and schedulers are to cover all modifications, equivalents, and alternatives falling within the scope of the invention as defined by the claims.

The present invention may, of course, be carried out in other ways than those specifically set forth herein without departing from essential characteristics of the invention. The present embodiments are to be considered in all respects as illustrative and not restrictive, and all changes coming within the meaning and equivalency range of the appended claims are intended to be embraced therein.

Figure 2 is a schematic illustration over a computer system 200. The computer system 200 may be situated within a computing device.

The computer system 200 comprises a user layer 215. An application software 210, within
5 the user layer 215, may run in the computer system 200.

The computer system 200 further comprises an operating system 225 and a hardware layer 235.

10 The operating system 225 comprises a scheduler 220. The scheduler 220 is configured to schedule processing resources.

The hardware layer 235 may comprise a multi-core chip 230. The multi-core chip 230, which may be referred to as a multi-core processor, comprises a plurality of processor
15 cores 231 and 232 although the cores 231, 232 in the multi-core chip 230 may vary in number in other embodiments. The processor cores 231 and 232 may further comprise further resources e.g. a cache memory, instruction processing engine, just to mention some arbitrary examples of core resources.

20 In the table below is summarized the expected speed-up as well as the suggested levels according to some embodiments. Even for 2048 core systems 200, five frequency levels (0, 1x, 2x, 4x, 8x) may suffice, with an acceptable impact on the theoretical performance. The performance differences are in the range of 1% - 25% for applications with degrees of parallelism between 0,5 and 0,999, higher for applications with less degree of parallelism.
25 However, all cores 231, 231, scalable or not, shall support at least levels 0 and 1x, i.e. it may be possible to switch off any core 231, 232.

Number of cores	Theoretical speed-up	Suggested factor
8	2	2
16	2,6	2
32	3,2	2
64	4	4
256	6,4	4
1024	10,1	8
2048	12,7	8

Embodiments of the present invention are thus based on the concept of a space-shared operating system 225, whereas the operating system 225 executes as service on some of
5 the cores 231, while other cores 232 execute just a microkernel without threading support. Cores 231, 232 may further be allocated to applications 210 upon request. The total power the many-core chip may use to execute applications 210 is managed by the operating system 225. This global resource is allocated to applications 210 through two resources; cores 231, 232 and frequencies at which these cores 231, 232 execute, i.e.
10 each application may be allocated a number of processor cores 231, 232, but the total computing power of these allocated number of processor cores 231, 232, i.e. the frequency at which these will execute, may be adjusted based on application needs and competing resource requests, within the limits set by the amount of scalable cores allocated to an application 210, the maximum frequency scaling factor that may be used
15 for those cores 231, 232 and the total power budget of the chip 230.

In order to allow the scheduler 220 to allocate computing resources, each application 210 may provide the operating system 225 with three parallelism parameters. The first parameter is degree of parallelism. The degree of parallelism may be expressed as a
20 fraction of the computation that may execute in parallel on multiple cores 231, 232. The second parameter is the minimum amount of cores 231, 232 that the application 210 needs in order to execute. The third parameter is the priority level of the application 210, which may appear quite similar to "process priority", as previously known.

25 Thus each application 210 may request two types of computing resources from the operating system 225. The first type of resource is processor core resource, which may be of type non-scalable, i.e. fixed frequency, or scalable i.e. scalable frequency. In the herein described non-limiting exemplary embodiments, the cores 231 are scalable and the cores 232 non- scalable. The allocation of this type of resource can occur at any time, e.g.
30 at application start-up based on the parallelism parameters and/ or during execution based on workload changes.

The second type of resource is execution speed. There are two scenarios for an application 210 to request allocation/de-allocation of such resources. Temporary

frequency boost/ restore to normal for the scalable cores 231 allocated to it, or shut off/ restore to normal for any core 231, 232 allocated to the application 210.

From application perspective, the present method may assume a behaviour similar to
5 handling memory today. Thus the application 210 may specify its parallelism parameters at start-up according to some embodiments. Also, the application 210 may request shutting off cores 231, 232 immediately when these are not needed, e.g. once a parallel computation has finished and a single-threaded sequence begins. Further, according to some embodiments, the application 210 may request frequency boosting as a resource
10 allocation request. The application 210 may according to some embodiments be responsible for requesting restoring to normal operating frequency of any core 231, 232 that was previously shut down. Further yet, the application 210 may be responsible for requesting allocation/ release of processor cores 231, 232 of various types i.e. scalable/ non-scalable.

15

According to some embodiments, the application 210 may be responsible to decide on which processor core 231, 232 a certain computation is to be executed, which requires a shift from today's approach where the application 210 just provides a set of threads that the operating system 225 will deploy on available resources

20

The present method is based on the following principal idea of having the processor cores 231, 232 on the chip 230 space-shared rather than time-shared. Thus each processor core 231, 232 may be allocated to one and only one application 210. Also, the allocation of processor cores 231, 232 may be geometry aware, in the sense that the operating
25 system 225 may group resources allocated to the same application 210 close to each other. Further, the operating system 225 may only shut off or free up allocated processor cores 231, 232 at the explicit request of the application 210 or when the application 210 exits. According to yet some embodiments, the operating system 225 may adjust the speed of individual scalable cores 231 if needed to accommodate new requests. All non-
30 allocated cores 232 may be kept in shut-off state. Further yet, the processing resource requests may have a wait time attached, such that a request with a zero wait time may return immediately, either with allocated resources or a negative response. A request with a positive wait-time may either return immediately with successfully allocated resources or be queued by the operating system 225 until it can be fulfilled, or a timer expires.

35

The present method according to some embodiments may begin with shutting off all cores 231, 232 that do not execute operating system code. When a new application 210 is started, it gets allocated, if possible, the amount of processor cores 231, 232 that it has specified in its parallelism parameters. Initially, all cores 231, 232 may be started at normal, 1x speed. Allocation of processor cores 231, 232 may be possible if the current power usage and the power required by the cores 231, 232 according to the application parallelism parameters is less than, or equal to, the total power budget, or the application 210 has a priority level that triggers re-scheduling of power budgets and the re-scheduling results in having enough power budget freed up.

10

When an application 210 requests the allocation of more processor cores 231, 232, the request may be fulfilled if the resulting power budget does not exceeds the total budget, or the application 210 has a priority level that triggers re-scheduling of power budgets and the re-scheduling results in having enough power budget freed up.

15

The operating system 225 actively re-calculates power budgets every time a application 210 requests shut off or restore of individual cores 231, 232 and may process pending requests if possible. Restoring may only succeed if the resulting power budget does not exceeds the total budget, or the application 210 has a priority level that triggers re-scheduling of power budgets and the re-scheduling results in having enough power budget freed up.

Restoring of operation of already allocated cores operating system 225 may have higher priority than allocation of new resources according to some embodiments.

25

When an application 210 requests frequency boosting or restoring for a core 231, 232, the operating system 225 may fulfil the request if and to the extent that the result does not mean exceeding the overall power budget of the processor chip. If the application 210 has a priority level that may trigger re-scheduling of power budgets, such a re-scheduling may be performed. If the request was to reduce frequency, the operating system 225 may process pending resource requests.

The operating system 225 may keep track of the period for which a scalable core 231 was running at boosted frequency; in order to avoid over-heating of the chip 230, the frequency may be scaled back after the pre-set maximum amount of time the core 231,

232 is allowed to run at such high frequencies. The scale-back may also trigger re-scheduling and servicing of outstanding requests. The scalable core 231 may be kept running at low frequency for a certain amount of time so as to allow a decrease in the temperature of the chip in order to avoid physical damage to the hardware, according to
5 some embodiments. Thus a timer 550 may be used in order to prevent the scalable core 231 from increasing the frequency before a certain predetermined time interval has passed. According to some embodiments the timer 550 may be activated at the moment when the scalable core 231 decreases the frequency from an increased frequency to a working frequency. A comparison between the measured timer value and a
10 predetermined time limit may then be performed, such that the scalable core 231 is prevented from increasing the frequency before a certain predetermined cooling period has passed, according to some embodiments.

Re-scheduling of resource allocations may be regarded as an advantageous component
15 of the present method. It may occur every time when a request cannot be fulfilled without over-stepping the power budget and the requesting application has a higher priority than at least some of the other executing applications 210.

During re-scheduling, the operating system 225 may basically modify the clocking of
20 frequency-boosted cores 231, 232. As stated before, an already allocated core 231, 232 may not be taken away forcefully by the operating system 225, neither may it be shut down. This may be a pre-requisite to allow applications 210 to perform correctly. However, modifying the frequency at which single-threaded portions of lower-prioritized applications execute is a sensible way of prioritizing applications, without starving any.
25 Hence, the operating system 225 may calculate how much the frequency scaling factor of some of the scalable cores executing lower-prioritized applications is to be reduced in order to fit in the power budget the requirements of higher prioritized applications. The applications 210 may be notified about the change and the application 210 with higher priority get its resources allocated.

30

There may be several different policies used by the scheduler 220 such as spreading the scale-back as evenly as possible to several applications 210, reducing the impact on each individual application 210, but impacting several applications 210. Also, the scheduler 220 may focus on just enough applications 210, the ones with the lowest priority, to free up
35 sufficient amount of power budget to fulfil the request.

Requests with non-zero wait time that cannot be fulfilled may be queued and may be serviced based on the priority of the application 210 making the request. If a request cannot be fulfilled before its timer expires, the request may be rejected and the application
5 210 notified.

In the space-shared operating system 225, the scheduling method outlined above may be implemented as two components, placed in the core-local micro-kernel and a central resource management service.

10

The core-local micro-kernel has the primary task of managing the hardware and interfacing the application 210 and the central resource management service. It provides e.g. the following services: Change the frequency of the core 231, 232, shut down the core 231, 232, receive and forward to the central resource management service requests

15 from the application 210.

In order to increase performance, replies from the central resource management service may be sent directly to the application 210, bypassing the local micro-kernel, according to some embodiments.

20

The central resource management service may be responsible for keeping track of all the processing resources i.e. scalable cores 231 and non-scalable cores 232 and the overall power budget. It has the function of allocating processing resources to newly started applications 210, re-calculating core frequencies and request adjustment from local micro-
25 kernels, wake-up / shut-down cores 231, 232 and managing and tracking pending resource requests.

The central resource management service itself may be implemented as an internally distributed service, i.e. executing on multiple cores 231, 232. "Central" in this context only
30 means that there may be one such service running in the system, visible to the distributed micro-kernels.

The present scheduling method thus may handle two types of resources: non-scalable cores 232 and scalable-cores 231. Both of these resources scale, quantity-wise, linearly
35 with the total number of cores 231, 232. In fact, the maximum number of scalable cores

231 equals $n/8$, while the maximum number of non-scalable cores 232 may be, obviously, less than n , where n is the total number of processor cores 231, 232 in the chip 230.

The present scheduling method may perform two tasks: keeping track of allocated cores 5 231, 232 i.e. which core 231, 232 to which application 210, and performing re-scheduling calculations.

The first task may complexity-wise be quite similar to memory management and there are algorithms with a constant complexity, not dependent upon the actual number of 10 resources or users, i.e. number of cores 231, 232 and amount of applications 210.

The task of re-scheduling may depend on the number of scalable cores 231, 232, limited to $n/8$ according to some embodiments. In fact, when a re-scheduling is to be performed, the scheduler may perform the following two steps:

15

Step 1

Identify the number of scalable cores 231 that are candidate for scaling back. These are cores 231 allocated to applications 210 with priority lower than the application 210 issuing the request that triggered the re-scheduling. As this information may be stored for each 20 active scalable core 231, which may be limited to max $n/8$, according to some embodiments, this information may be obtained by traversing a list with $n/8$ elements, yielding a complexity of $O(n/8)$.

Step 2

25 Out of the candidate scalable cores 231, the method may select which core 231 have their frequency reduced and by how much. The gain with each frequency-step can be pre-calculated, hence the total number of frequency changes may be calculated as a simple calculation, which may be independent of the total number of cores 231, 232, but dependent on the chosen policy; in any case it will be limited to the number of scalable 30 cores, hence $O(n/8)$.

In conclusion, the complexity of the re-scheduling algorithm may be $O(n/8)$ in best case, with a worst case complexity of $O(n/4)$. The key point is though that it scales linearly with the potential number of scalable cores and involves, even for a 1024 core chip, working 35 with tables of at most 128 elements.

Concerning memory, the scheduler may have to store only four pieces of information for each scalable core 231, to which application 210 it is allocated, current frequency scaling factor, dead-line until it can execute at this frequency and application priority, for
5 performance reasons, the amount of memory needed for even a 1024 core system may be just a few kilobytes. For each non-scalable cores information such as current state i.e. on/off and application information needs to be stored, adding a few extra kilobytes. Even on a 64 bit machine, the amount of memory required would be $n/8 * 3 * 8 + n * 8$, i.e. $11 * n$ bytes, about 11 kilobytes for a 1024 core chip.

10

Figure 3 illustrates a multi-core chip 230 according to some embodiments.

An aspect to be considered may be the geometry of the multi-core chip 230. This may be an advantage primarily in order to avoid hot spots and over-heating of the multi-core chip
15 230, thus the processor cores 231 that can execute at higher frequencies may be spread out as evenly as possible and surrounded by other processor cores 232, running at lower frequencies. According to some optional embodiments, a hierarchical chip interconnect may be built, by connecting the scalable cores 231 with a high-speed interconnect and then connect groups of other cores 232 to each scalable core 231. A possible solution,
20 out of a plurality of possible solutions for a 32 core chip according to some embodiments is shown in Figure 3. Each square represents a non scalable core 232, and/ or scalable cores 231, while the thick lines showing the high speed, top level interconnect.

Figure 4 is a flow chart illustrating embodiments of method steps 401-405 performed in
25 an operating system 225. The method aims at scheduling processing resources on a multi-core chip 230. The multi-core chip 230 comprises a plurality of processor cores 231, 232. The operating system 225 is configured to schedule processing resources to an application 210 to be executed on the multi-core chip 230.

30 The multi-core chip 230 may comprise one or several processor cores 231 of a first type which may be adapted to operate on either a first operating frequency or at least one second frequency. Also, the multi-core chip 230 may comprise one or several processor cores 232 of a second type which are adapted to operate on either the first operating frequency or to be switched off, according to some embodiments.

35

The processor cores 231 of the first type may be referred to as scalable cores and the processor cores 232 of the second type may be referred to as non-scalable cores. The first operating frequency of the processor cores 231 of the first type and the first operating frequency of the processor cores 232 of the second type may be different, according to
5 some embodiments. However, according to some embodiments, the first operating frequency may be the same for the processor cores 231 of the first type and the processor cores 232 of the second type.

The processor cores 231 the first type which are adapted to operate on either a first
10 operating frequency and at least one second frequency may be spread out over the multi-core chip 230 and surrounded by at least one processor core 232 of the second type which is adapted to operate on either the first operating frequency or to be switched off, such that hot spots on the multi-core chip 230 may be avoided.

15 All non-allocated processor cores 231, 232 comprised on the multi-core chip 230 may be kept in shut off state, according to some embodiments.

The operating system 225 may be configured to schedule processing resources to a plurality of applications 210 to be executed on the multi-core chip 230, according to some
20 embodiments.

To appropriately schedule processing resources on a multi-core chip 230, the method may comprise a number of method steps 401-405.

25 It is however to be noted that some of the described method steps 401-405 are optional and only comprised within some embodiments. Further, it is to be noted that the method steps 401-405 may be performed in any arbitrary chronological order and that some of them, e.g. step 401 and step 403, or even all steps may be performed simultaneously or in an altered, arbitrarily rearranged, decomposed or even completely reversed
30 chronological order. The method may comprise the following steps:

Step 401

A plurality of processor cores 231, 232 are allocated to the application 210.

The number of initial allocation of processor cores 231, 232 to the application 210 may be based on at least one application related parameter of: degree of parallelism, minimum amount of needed processor cores 231, 232 and/ or priority level, according to some embodiments.

5

Step 402

When a sequential portion of the application 210 is executing on only one processor core 231, another processor core 232, allocated to the application 210, not executing the sequential portion of the application 210 is switched off.

10

Step 403

The frequency of some of scalable cores allocated to other applications with a lower priority than the application 210 is decreased. Thereby the frequency increase of the core 231 executing the application 210 is possible to perform, within the overall power budget

15 of the system.

Step 404

The frequency of the one processor core 231 executing the application 210 is increased to a second frequency, such that the processing speed is increased more than predicted

20 by Amdahl's law.

The frequency of one or several processor cores 231 may be increased to the second frequency, for the same application 210, or for multiple applications according to some embodiments.

25

The second frequency may be set to any of: 2, 4 or 8 times the first operating frequency, based on the total number of processor cores 231, 232 on the multi-core chip 230.

According to some embodiments, a timer 550 may be set to a predetermined time value

30 and switching the frequency of the one processor core 231 back to the first operating frequency when the predetermined time has passed. Thus overheating and physical damage of the processor core 231 may be avoided.

The processor core 231 may according to some embodiments be prevented to re-

35 increase the frequency before a predetermined cooling period has expired. The cooling

period may be monitored e.g. by the timer 550. Thus the timer 550 may be set to a predetermined value at the moment of frequency decrease of the processor core 231 and a count down to zero may be initiated before the frequency of the processor core 231 is allowed to increase again, according to some embodiments. Alternatively, the timer 550
5 may be set to zero and start counting the time at the moment of frequency decrease of the processor core 231 and compared to a time threshold value. Thus the processor core 231 may be prevented to re-increase the frequency before the predetermined cooling period has passed. Thus overheating and physical damage of the processor core 231 may be avoided.

10

Further, the assumed total power consumption of the processor cores 231, 232 comprised on the multi-core chip 230 may be computed after the frequency increase and the frequency increase may be adjusted such that a predetermined total power consumption limit is not exceeded, according to some embodiments.

15

Step 405

This step is optional and may only be performed within some embodiments.

When the sequential portion of the application 210, which portion is executing on only one
20 processor core 231, is terminated, the frequency of the processor cores 231, 232 allocated to the application 210 may be restored to the respective first operating frequency, i.e. the frequency that was used before performing the step of increasing 404 the frequency.

25 However, according to some embodiments, the frequency may be restored to the first operating frequency when a predetermined time has expired, measured by the timer 550.

Figure 5 is a block diagram illustrating embodiments of a scheduler 220 situated in an operating system 225. The scheduler 220 is configured to perform the method steps 401-
30 405 for scheduling processing resources on a multi-core chip 230 to an application 210 to be executed on the multi-core chip 230. The multi-core chip 230 comprises a plurality of processor cores 231, 232.

For the sake of clarity, any internal electronics of the scheduler 220, not necessary for
35 understanding the present solution has been omitted from Figure 5.

The scheduler 220 comprises an allocation unit 510. The allocation unit 510 is adapted to allocate a plurality of processor cores 231, 232 to the application 210. Also, the scheduler 220 comprises a switching unit 520. The switching unit 520 is adapted to switch off a processor core 232 allocated to the application 210, not executing a sequential portion of the application 210. Further, the scheduler 220 comprises an increasing unit 530. The increasing unit 530 is adapted to increase the frequency of the one processor core 231 executing the application 210, such that the processing speed is increased more than predicted by Amdahl's law.

10

The scheduler 220 may further comprise a restoring unit 540. The optional restoring unit 540 is adapted to restore the frequency of the processor cores 231, 232 allocated to the application 210 when the sequential portion of the application 210, which portion is executing on only one processor core 231, is terminated.

15

According to some embodiments, the scheduler 220 may comprise a timer 550. The timer 550 may be set to a predetermined time value and switching the frequency of the one processor core 231 back to the first operating frequency when the predetermined time has passed.

20

The timer 550 may further be configured to monitor the cooling period of the processor core 231, such that the processor core 231 is prevented from re-increasing the frequency before a predetermined cooling period has expired.

It is to be noted that the described units 510-540 comprised within the scheduler 220 may be regarded as separate logical entities, but not with necessity as separate physical entities. Any, some or all of the units 510-540 may be comprised or co-arranged within the same physical unit. However, in order to facilitate the understanding of the functionality of the scheduler 220, the comprised units 510-540 are illustrated as separate units in Figure 5.

Figure 6 illustrates the results assuming a multi-core chip 230 comprising 256 processor cores 231, 232 where a specific application has 8, 16, 32 and 64 cores allocated, respectively. A near linear increase in performance may be observed for even less parallelized applications, such as e.g. the speedup for $f = 0,5$ is above 7 when using 8

processor cores 231, 232, but switching off all 256 processor cores 231, 232 when executing the sequential portion according to some embodiments of the present scheduling and resource management method.

5 **Computer program product in the operating system 225**

The method steps 401-405 in the operating system 225 may be implemented through one or more processor cores 231, 232 in the multi-core chip 230, together with computer program code for performing the functions of the present method steps 401-405. Thus a computer program product, comprising instructions for performing the method steps 401-10 405 in the operating system 225 may schedule processing resources on a multi-core chip 230.

The computer program product mentioned above may be provided for instance in the form of a data carrier carrying computer program code for performing the method steps 15 according to the present solution when being loaded into the processor core 231, 232. The data carrier may be e.g. a hard disk, a CD ROM disc, a memory stick, an optical storage device, a magnetic storage device or any other appropriate medium such as a disk or tape that can hold machine readable data. The computer program code may furthermore be provided as program code on a server and downloaded to the processor 20 core 231, 232 remotely, e.g. over an Internet or an intranet connection.

The present invention may be embodied as a method and a scheduler in an operating system 225, and/ or computer program products. Accordingly, the present invention may take the form of an entirely hardware embodiment, a software embodiment or an 25 embodiment combining software and hardware aspects all generally referred to herein as a "circuit". Furthermore, the present invention may take the form of a computer program product on a computer-usable storage medium having computer-usable program code embodied in the medium. Any suitable computer readable medium may be utilized including hard disks, CD-ROMs, optical storage devices, a transmission media such as 30 those supporting the Internet or an intranet, or magnetic storage devices etc.

The terminology used in the detailed description of the particular exemplary embodiments illustrated in the accompanying drawings is not intended to be limiting of the invention.

As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless expressly stated otherwise. It will be further understood that the terms "includes," "comprises," "including" and/or "comprising," when used in this specification, specify the presence of stated features, integers, steps, operations, 5 elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof. It will be understood that when an element is referred to as being "connected" or "coupled" to another element, it can be directly connected or coupled to the other element or intervening elements may be present. Furthermore, "connected" or "coupled" as used 10 herein may include wirelessly connected or coupled. As used herein, the term "and/or" includes any and all combinations of one or more of the associated listed items.

CLAIMS

1. Method in an operating system (225) for scheduling processing resources on a multi-core chip (230), the multi-core chip (230) comprises a plurality of processor cores (231, 232), the operating system (225) is configured to schedule processing resources to an application (210) to be executed on the multi-core chip (230), the method comprises the steps of:
 - allocating* (401) a plurality of processor cores (231, 232) to the application (210),
 - when a sequential portion of the application (210) is executing on only one processor core (231), *switching off* (402) at least one processor core (232) allocated to the application (210), not executing the sequential portion of the application (210), and
 - increasing* (404) the frequency of the one processor core (231) executing the application (210), such that the processing speed is increased more than predicted by Amdahl's law.
2. Method according to claim 1, wherein the method comprises the further step of:
 - when the sequential portion of the application (210) executing on only one processor core (231) is terminated, *restoring* (405) the frequency of the processor cores (231, 232) allocated to the application (210), to the respective frequency performed before the step of increasing (404) the frequency.
3. Method according to any of the claims 1 or 2, wherein the multi-core chip (230) comprises one or several processor cores (231) of a first type, which are adapted to operate on either a first operating frequency or at least one second frequency, and one or several processor cores (232) of a second type, which are adapted to operate on either the first operating frequency or to be switched off.
4. Method according to any of the claims 1-3, wherein the frequency of one or several processor cores (231) are increased to the second frequency, for the same application (210), or for multiple applications.
5. Method according to any of the claims 3 or 4, wherein the processor cores (231) which are adapted to operate on either a first operating frequency and at least one second frequency are spread out over the multi-core chip (230) and surrounded by at least one

processor core (232) which is adapted to operate on either the first operating frequency or to be switched off, such that hot spots on the multi-core chip (230) are avoided.

6. Method according to any of the claims 1-5, wherein the second frequency is set to
5 any of: 2, 4 or 8 times the first operating frequency, based on the total number of processor cores (231, 232) on the multi-core chip (230).

7. Method according to any of the claims 1-6, wherein all non- allocated processor cores (231, 232) comprised on the multi-core chip (230) is kept in shut off state.

10

8. Method according to any of the claims 1-7, wherein the number of initial allocation of processor cores (231, 232) to the application (210) is based on at least one application related parameter of: degree of parallelism, minimum amount of needed processor cores (231, 232) and/or priority level.

15

9. Method according to any of the claims 1-8, wherein the step of increasing (404) the frequency of the one processor core (231) executing the application (210) comprises setting a timer (550) to a predetermined time value and switching the frequency of the one processor core (231) back to the first operating frequency when the predetermined time
20 has passed, such that overheating and physical damage of the processor core (231) is avoided.

10. Method according to any of the claims 1-9, wherein the step of increasing (404) the frequency of the one processor core (231) executing the application (210) comprises
25 computing the assumed total power consumption of the processor cores (231, 232) comprised on the multi-core chip (230), after the frequency increase and adjust the frequency increase such that a predetermined total power consumption limit is not exceeded.

30 11. Method according to any of the claims 1-10, wherein the operating system (225) is configured to schedule processing resources to a plurality of applications (210) to be executed on the multi-core chip (230).

12. Method according to any of the claims 1-11, wherein the method comprises the
35 further step of:

decreasing (403) the frequency of some of scalable cores allocated to other applications with a lower priority than the application (210), such that the frequency increase of the processor core (231) executing the application (210) is possible to perform, within the overall power budget of the system.

5

13. Method according to any of the claims 1-12, wherein the processor core (231) is prevented to re-increase the frequency before a predetermined cooling period has expired.

10 14. Scheduler (220) in an operating system (225) for scheduling processing resources on a multi-core chip (230) to an application (210) to be executed on the multi-core chip (230), the multi-core chip (230) comprises a plurality of processor cores (231, 232), the scheduler (220) comprises:

an allocation unit (510), adapted to allocate a plurality of processor cores (231,
15 232) to the application (210),

a switching unit (520), adapted to switch off a processor core (232) allocated to the application (210), not executing a sequential portion of the application (210), and

an increasing unit (530), adapted to increase the frequency of the one processor core (231) executing the application (210), such that the processing speed is increased
20 more than predicted by Amdahl's law.

1 / 6

Speed-up according to Amdahl's law

Speedup

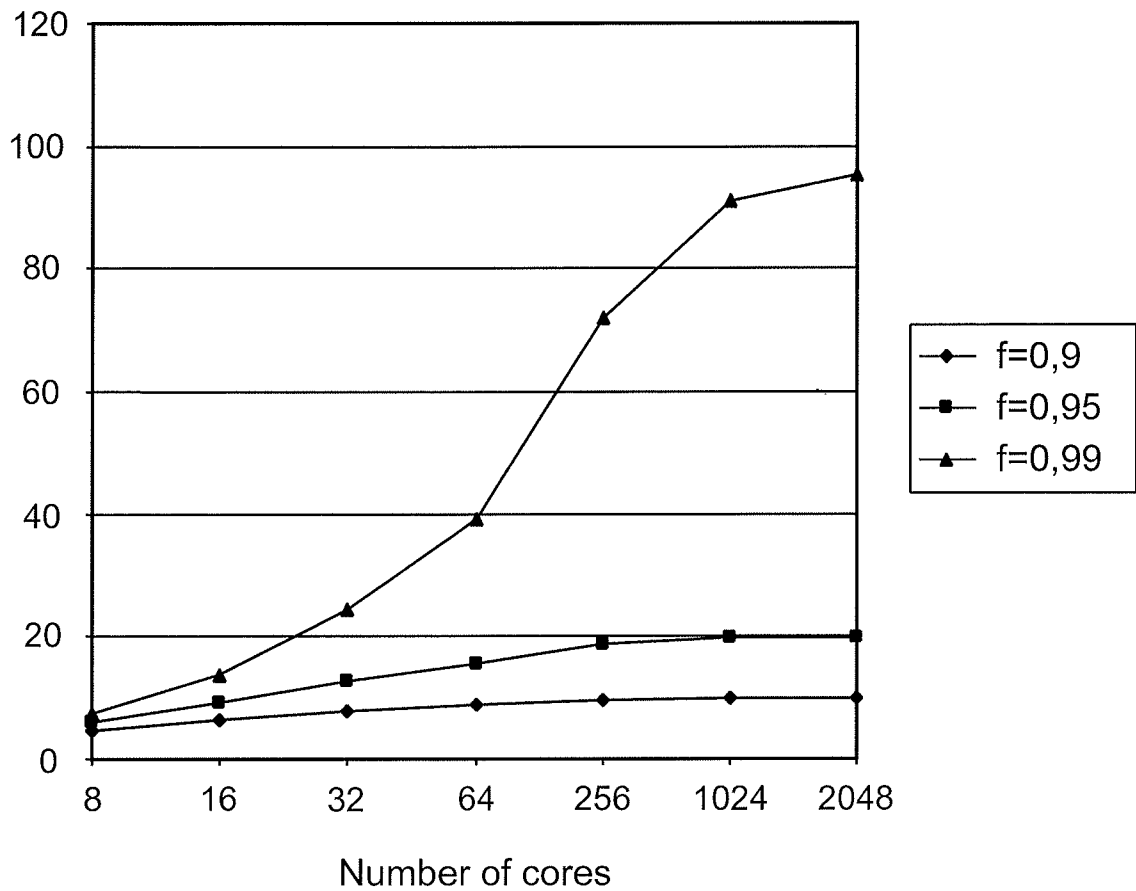


Fig. 1

2 / 6

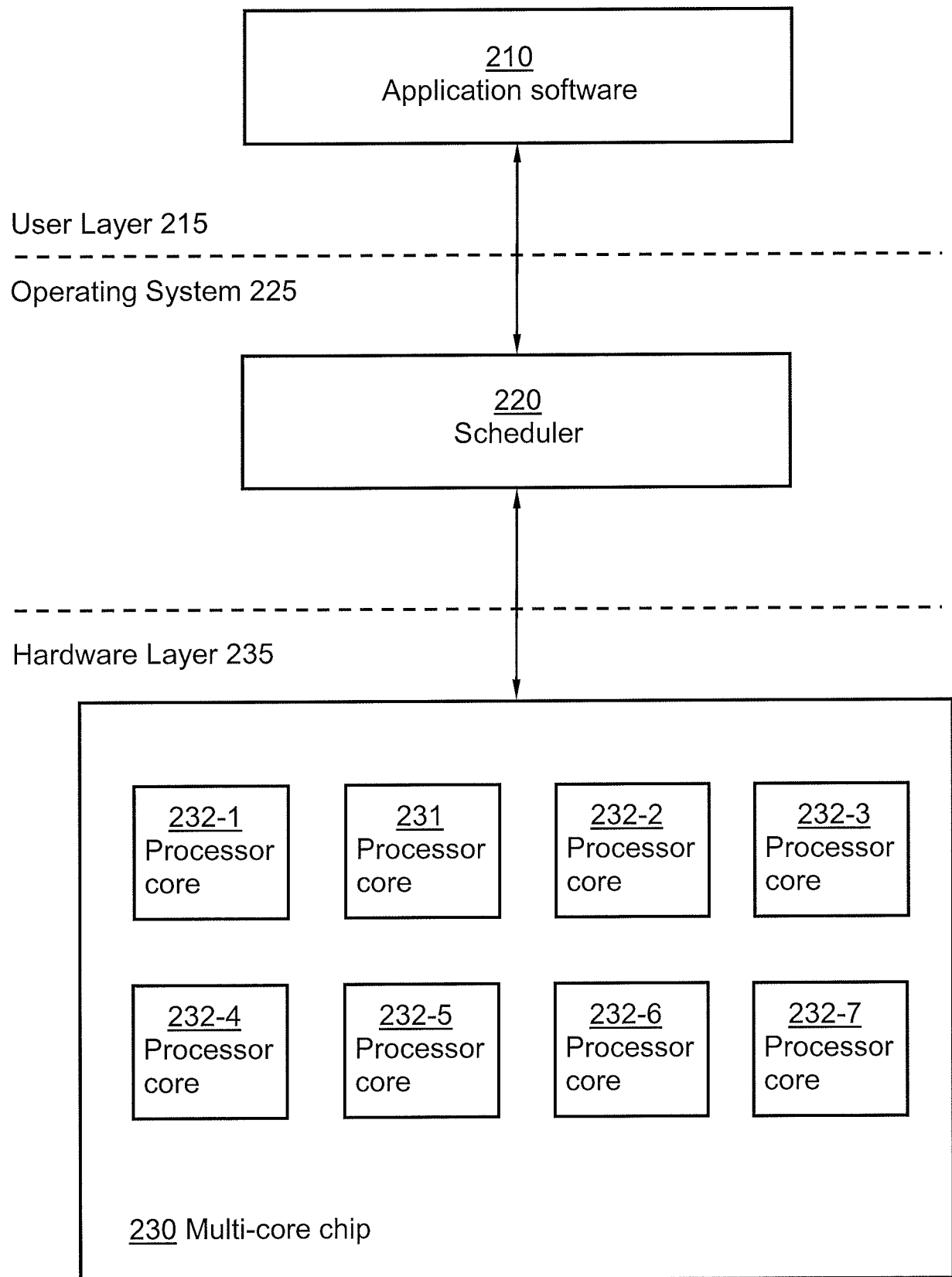


Fig. 2

3 / 6

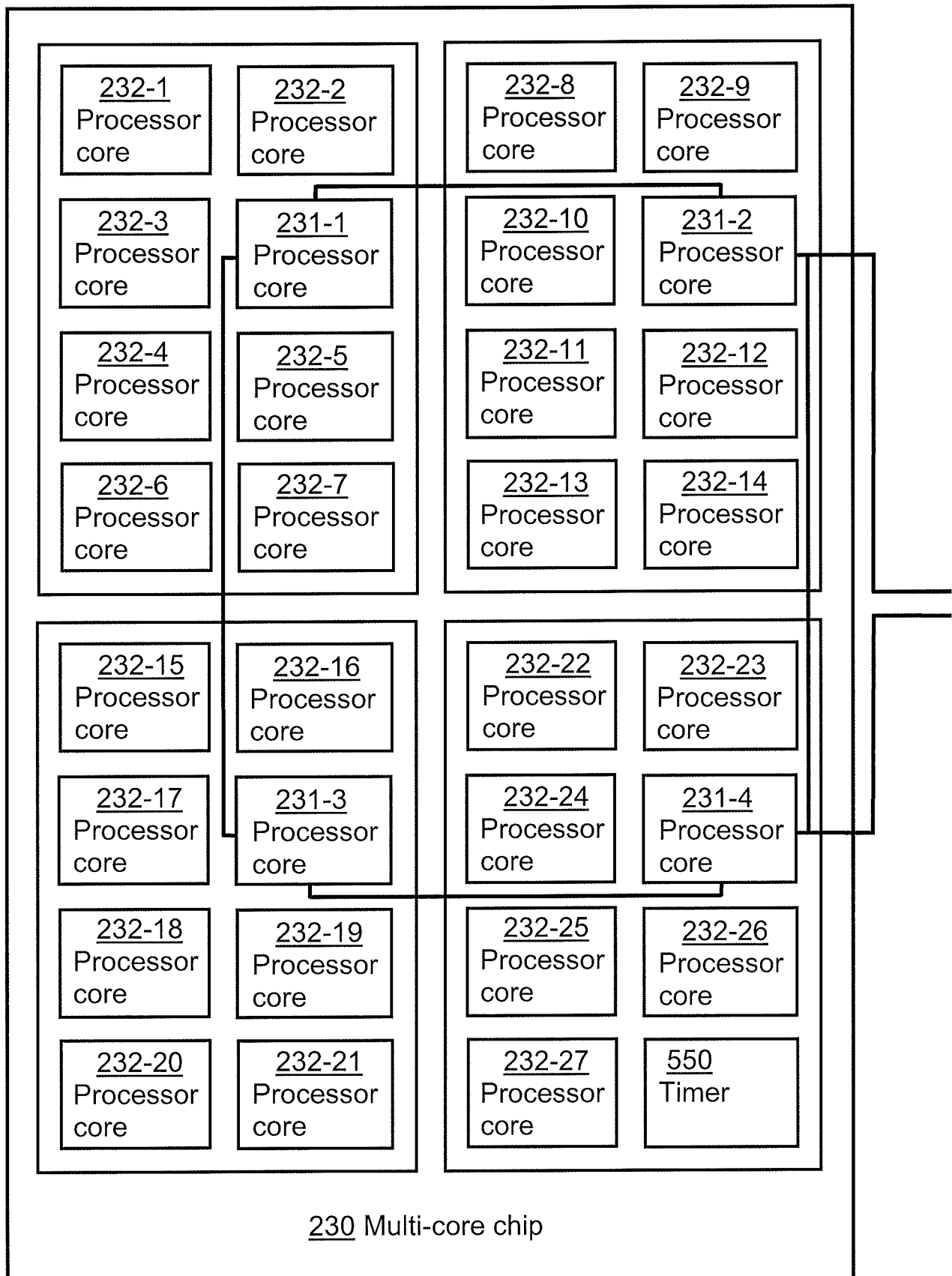


Fig. 3

4 / 6

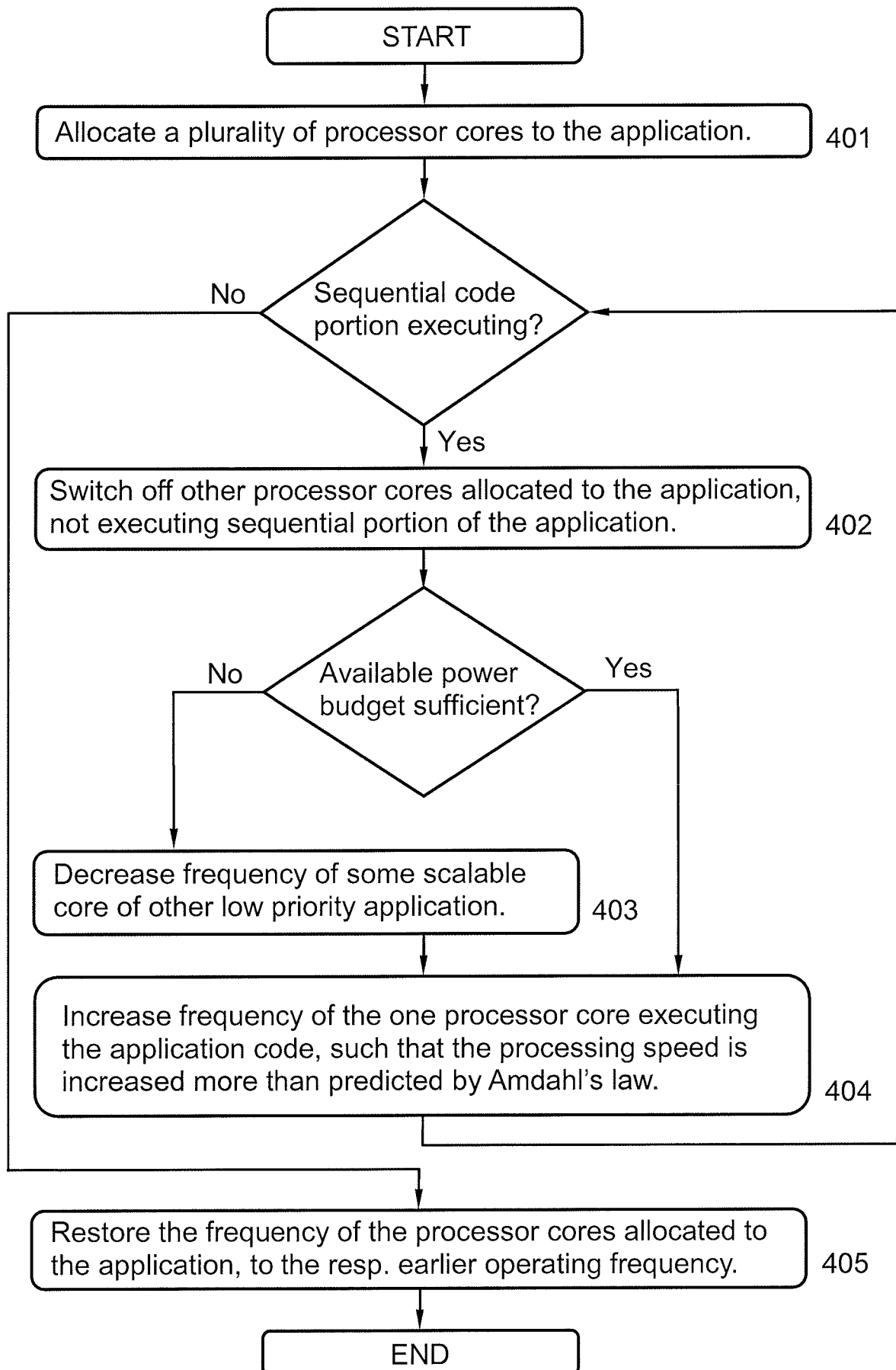


Fig. 4

5 / 6

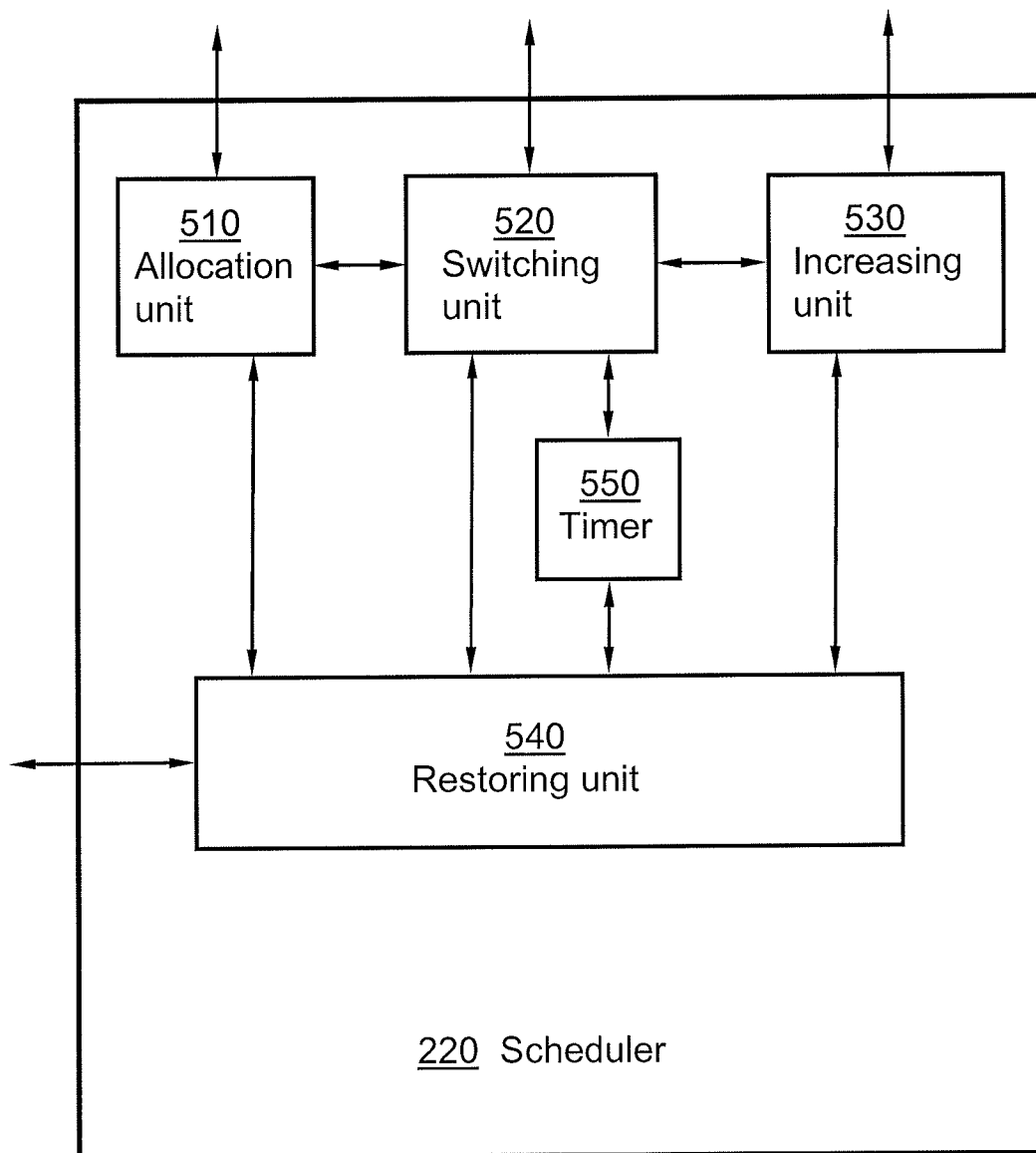


Fig. 5

6 / 6

Speedup

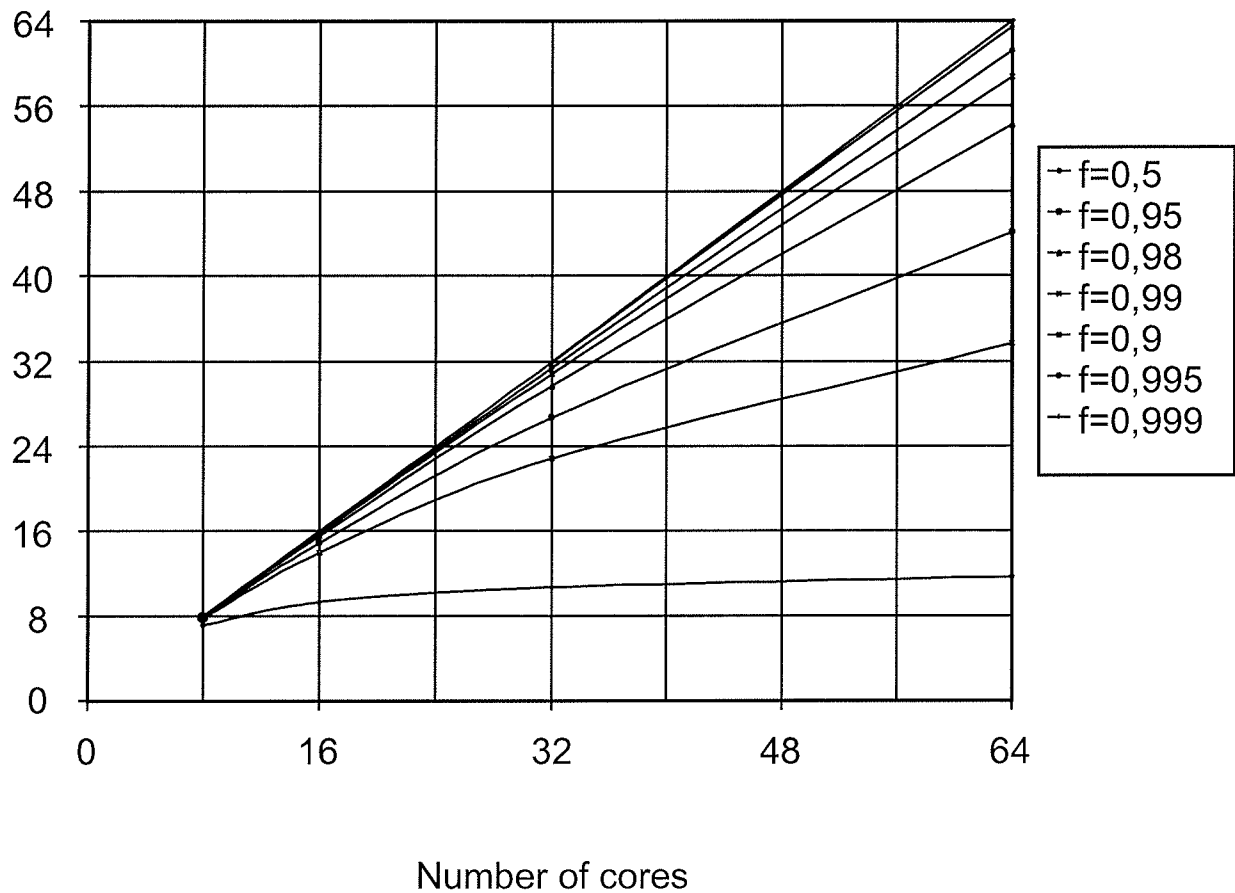


Fig. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/SE2009/050594

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F9/48 G06F1/32
ADD. G06F1/20

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EP0-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2007/019003 A (ADVANCED MICRO DEVICES INC [US]; CLARK MICHAEL T [US]) 15 February 2007 (2007-02-15) page 3, line 18 - line 30 page 5, line 36 - page 6, line 17 -----	1-14
X	US 2008/005592 A1 (ALLAREY JOSE [US] ET AL) 3 January 2008 (2008-01-03) paragraphs [0015] - [0017], [0020], [0021], [0049] - [0055] ----- -/--	1-14

☒ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search

15 March 2010

Date of mailing of the international search report

29/03/2010

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Michel, Thierry

INTERNATIONAL SEARCH REPORT

International application No
PCT/SE2009/050594

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	R. KUMAR ET AL: "A multi-core approach to addressing the energy-complexity problem in microprocessors" IN WORKSHOP ON COMPLEXITY-EFFECTIVE DESIGN, [Online] June 2003 (2003-06), pages 1-8, XP002573062 Retrieved from the Internet: URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.117.8180 [retrieved on 2010-10-08] page 1, right-hand column, line 26 - page 3, right-hand column, line 10 -----	1-14
A	WO 2005/010737 A (INTEL CORP [US]) 3 February 2005 (2005-02-03) paragraphs [0011], [0014], [0017], [0018] -----	1-14
A	US 2004/037346 A1 (RUSU STEFAN [US] ET AL) 26 February 2004 (2004-02-26) the whole document -----	1-14

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/SE2009/050594

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
WO 2007019003	A	15-02-2007	CN 101233475 A	30-07-2008
			DE 112006002056 T5	10-07-2008
			GB 2442919 A	16-04-2008
			JP 2009503728 T	29-01-2009
			KR 20080038389 A	06-05-2008
			US 2007033425 A1	08-02-2007
			US 2009187777 A1	23-07-2009
US 2008005592	A1	03-01-2008	NONE	
WO 2005010737	A	03-02-2005	CN 1577280 A	09-02-2005
			CN 101320289 A	10-12-2008
			GB 2420435 A	24-05-2006
			JP 4413924 B2	10-02-2010
			JP 2007535721 T	06-12-2007
			KR 20060031868 A	13-04-2006
			TW 280507 B	01-05-2007
			US 2006117199 A1	01-06-2006
			US 2006123263 A1	08-06-2006
			US 2006117200 A1	01-06-2006
			US 2006123264 A1	08-06-2006
			US 2005050310 A1	03-03-2005
			US 2007198872 A1	23-08-2007
US 2004037346	A1	26-02-2004	AU 2003265622 A1	11-03-2004
			CN 1692326 A	02-11-2005
			EP 1552367 A2	13-07-2005
			TW 226533 B	11-01-2005
			WO 2004019195 A2	04-03-2004
			US 2005180488 A1	18-08-2005