J Supercomput (2013) 66:305–319
DOI 10.1007/s11227-013-0908-9

Extending Amdahl's law and Gustafson's law by evaluating interconnections on multi-core processors

Tian Huang · Yongxin Zhu · Meikang Qiu · Xiaojing Yin · Xu Wang

Published online: 25 April 2013

Abstract Multicore chips are emerging as the mainstream solution for high performance computing. Generally, communication overheads cause large performance degradation in multi-core collaboration. Interconnects in large scale are needed to deal with these overheads. Amdahl's and Gustafson's law have been applied to multi-core chips but inter-core communication has not been taken into account. In this paper, we introduce interconnection into Amdahl's and Gustafson's law so that these laws work more precisely in the multi-core era. We further propose an area cost model and analyse our speedup models under area constraints. We find optimized parameters according to our speedup model. These parameters provide useful feedbacks to architects at an initial phase of their designs. We also present a case study to show the necessity of incorporating interconnection into Amdahl's and Gustafson's law.

Keywords Amdahl's law · Gustafson's law · Interconnection · Multi-core processor · Chip area · Model

1 Introduction

Amdahl's law [1] and Gustafson's law [2] are fundamentals rules in classic computer science theories. They guided the development of mainframe computers in the 1960s and multi-processor systems in the 1980s of the last century.

As we evolve into multi-core era, multi-core architecture designers integrate multiple processing units into one chip to work around the I/O wall and power wall of

T. Huang · Y. Zhu (✉) · X. Yin · X. Wang
School of Microelectronics, Shanghai Jiao Tong University, Shanghai, China
e-mail: zhuyongxin@sjtu.edu.cn

M. Qiu
Dept. of Electrical and Computer Engineering, University of Kentucky, Lexington, USA

single core architectures [3, 4]. Multi-core architectures provide a new dimension to scale up the number of processing elements (cores) and, therefore, the potential computing capacity. Now the technology is readily available and many-core processors with hundreds of cores go into mass production. The Tile Processor's first implementation, the TILE64 [5], contains 64 cores and can execute 192 billion 32-bit operations per second at 1 GHz. Tilera's Tile-Gx 100 [6] chip will reportedly scale to 100 cores without rewiring. More than one hundred cores are expected to be integrated into a single chip by 2015 [7]. Although Intel announced a road map to build a special-purpose 80-core processor [8] by 2011, their approach is conservative and slow-paced compared to the ambitious plans of smaller competitors. Leading manufacturers may share the same pessimistic view of multi-core scalability as that expressed theoretically by Hill and Marty [9].

In the multi-core era, some studies were made to extend Amdahl's law and Gustafson's law. Hill and Marty [9] proposed corollaries to Amdahl's law by modelling the performance caps of symmetric, asymmetric, and dynamic multi-core architectures. Woo and Lee [10] further incorporated energy models into Amdahl's law. Sun [11] reevaluated Amdahl's law and Gustafson's law in the multi-core era and pointed out that performance improvement will not be limited by the sequential processing term if users increase their computing demands when given more computing power.

When the laws of Amdahl and Gustafson were invented, all processors contained single core with no room for concern over on-chip communications between multiple cores. Amdahl and Gustafson naturally modeled an application as a pure computational workload without explicit considerations on inter-core communication. When we exploit more parallelism by mapping tasks onto more cores, inter-core communication overhead may rise rapidly, sometimes even exponentially [12]. Huge communication overhead and contention could no longer be ignored since it makes interconnection the bottleneck of system performance. Many algorithms of scheduling are developed to overcome this problem [13–16].

In this paper, we propose our new speedup models by respectively incorporating workload of data transmission into Amdahl's law and Gustafson's law. Since the silicon area of a chip is limited, architectural design decisions, such as number of cores, number of transistors allocated to one core, should be carefully made at the initial phase of design to coordinate with particular applications. Similar tradeoff about area also happens to cores and interconnects. We introduce interconnection into a silicon area cost model and incorporate the cost model into our speedup models. We analyse our speedup models under area constraints to evaluate the impact of communication on system performance. By analyzing our speedup models with respect to Amdahl's law, we work out the analytical expressions of the maximum performance that a given silicon area achieves for any tasks and corresponding optimized parameters of system-level architectural design with regard to both cores and interconnects.

Intuitively, it is impossible that the performance of a system can be infinitely improved by scaling up the number of cores and the problem size. As the problem size increases, the communication requirement usually grows rapidly so that the interconnects of original system can hardly handle it. In this paper, we try to find the most efficient architectural configuration that makes the best use of scalable computing model and prevents communication contention from excessively degrading the performance of a system.

We also present a case study in which an image processing application, i.e. CT (Computer Tomography) [17], is applied to evaluate our new speedup models. We present a method to evaluate the parameters of parallelism of an application so that these parameters can be the input of the speedup models. This case study shows that interconnects impose huge impact on the performance of multi-core systems. Ignoring this impact will cause over-pessimistic view (Amdahl's law) or over-optimistic view (Gustafson's law) on speedup. It is necessary to carefully deal with interconnects at the initial phase of architectural designs. Modern hardware such as GPU, FPGA is more exquisite in interconnection design therefore becomes much more popular than a general-purpose multi-core processor in CT image reconstruction.

The rest of this paper is organized as follows. In Sect. 2, we incorporate interconnection into Amdahl's law and Gustafson's law. In Sect. 3, we incorporate interconnection into an area cost model and apply this cost model to the speedup models we proposed in Sect. 2. In Sect. 4, we analyse the speedup models under area constraints. A case study is presented in Sect. 5. Section 6 draws the conclusion.

2 Extending Amdahl's law and Gustafson's law

Amdahl's law and Gustafson's law are fundamentals rules in the theories of classic computer science. However, they are not suitable for modern multi-core architecture since inter-core communication contention has become a significant part of system overheads. In this section, we incorporate interconnection into these models so that they work more precisely in the multi-core era.

2.1 Extending Amdahl's law

Amdahl's law is made to analyse the effect of parallelism on system speedup given a fixed-size problem. It states that if a portion of a task, f, can be improved by a factor m, and the rest cannot be improved, then the portion that cannot be improved will quickly dominate the performance, and further improvement of the parallel workload will bring little impact. Speedup is defined as sequential execution time over parallel execution time in parallel processing. Let f be the portion of the workload that can be parallelized and m be the number of processors; then the parallel processing speedup implied by Amdahl's law is described in Eq. (1).

$$S_A(f,m) = \frac{1}{(1-f) + \frac{f}{m}} \tag{1}$$

$S_A(f,m)$ denotes Amdahl's speedup model. Amdahl treated task as a combination of sequential and parallel workload of pure computation. Since communication overhead has already become a significant part of workloads on multi-core architecture, we should take transmission workload into account.

The transmission overheads between local memories and caches are usually insignificant compared to those inter-core communications over Network on Chip

(NoC). As traffic on NoC grows exponentially with number of cores, we focus on the transmission over NoC in our categorization of an application task. As shown in Eq. (2), we redivide an application task into sequential and parallel workload of computation and transmission and analyse them, respectively.

$$1 = f_c^s + f_c^p + f_t^p + f_t^s \tag{2}$$

f_c^p and f_t^p, respectively, represent parallel workload of computation and transmission that can be speeded up by more cores and interconnects. f_c^s and f_t^s, respectively, represent the sequential part of computation and transmission that can not be improved by parallel execution. In the rest of this paper, we use $\hat{f}$ to represent the tuple of task parameters $(f_c^s, f_t^s, f_c^p, f_t^p)$.

We introduce a parameter i, the number of interconnects, to make practical sense with our model. Corresponding to our definition of transmission, i can be explained as the number of links of a single node or core in practical NoC on many core processors. Equation (3) shows the basic extension of Amdahl's law:

$$S_A(\hat{f}, m, i) = \frac{1}{f_t^s + f_c^s + \frac{f_c^p}{m} + \frac{f_t^p}{i}} \tag{3}$$

Equation (3) is similar to Eq. (1). It is a two-dimension extension of Eq. (1). We may easily draw a conclusion from Eq. (3) that as the execution time of parallel workload is significantly reduced by parallel computing and transmitting, the sequential workload of computation and transmission will dominate the total execution time and prevent tasks from achieving further speedup.

Intuitively, the more parallelism of computation exists, the more transmission overhead will be incurred. This issue is nevertheless ignored by Amdahl's law targeting a fixed-size speedup problem. We explicitly address this issue in our extension of Amdahl's law. We will consider additional transmission generated by more parallelism in our extension of Gustafson's law with the scaled-up size of problems.

2.2 Extending Gustafson's law

In 1988, Gustafson [2] addressed the issue of scalable computing coupled with a fixed-time speedup model. The fixed-time speedup model argues that powerful machines will be designed for large problems and problem size should scale up with the increase of computing capability. For many practical workloads, the scale of problem size is bounded by the execution time. In this paper, Gustafson's fixed-time speedup is denoted as S_G.

$$S_G = \frac{\text{Scaled workload within a fixed period of time}}{\text{Original workload within a fixed period of time}} \tag{4}$$

We use w and w' to respectively represent the original workload and the scaled-up workload. In the context of Gustafson's law, it is reasonable to suppose the parallel workloads of tasks are the only part that can be scaled, then we have $w' = (1 - f) \times$

$w + fmw$. Therefore,

$$\begin{aligned} S_G(f,m) &= \frac{w'}{w} \\ &= \frac{(1-f)\times w + fmw}{w} \\ &= (1-f) + fm \end{aligned} \tag{5}$$

Equation (5) means that a computer with m cores can deal with the parallel workloads m times heavier than a single-core computer does within the same time. The workload here only refers to the computational workload. Increasing the workload of parallel computation will also cause the growth of communication overhead. If the performance of interconnection does not grow with the number of cores, the total execution time will increase. Therefore, we introduce a factor Δ to extend Gustafson's law by revising S_G as

$$\begin{aligned} S_G(\hat{f},m,i) &= \frac{\text{new workload}}{\text{original workload}} \times \Delta \\ &= \frac{f_c^s + f_c^p m}{f_c^s + f_c^p} \times \frac{1}{f_c^s + f_c^p + f_t^s + \frac{K(m) f_t^p}{i}} \end{aligned} \tag{6}$$

Δ is a correction of execution time. The numerator of Δ equals to 1 and represents the execution time of original workload required by original processor and interconnection. The denominator represents the execution time of new workload with m processors and i interconnects. $K(m) f_t^p$ means the parallel transmission of new workload. $K(m)$ is a task specific function. It fundamentally grows with number of processors, when $m = 1$, $K(m) = 1$.

We apply a practical constraint $i < \frac{m\times(m-1)}{2}$ on Eq. (6) to prevent Δ from growing more than one. This restriction means the total number of interconnects i is no more than that of a crossbar network with m cores.

Δ is typically a value between zero and one. It usually decreases as the problem size increases. This factor prevents a task from being infinitely sped up by infinite number of cores.

3 Speedup model under area constraint

Hill and Marty used a simple cost model for the chip area of multi-core in [9]. That is

$$A(m,r) = m \times r \tag{7}$$

$A(m,r)$ represents the area of a chip, while m and r, respectively, stands for the number of cores and the number of transistors used to form a core. Hill and Marty considered the area costs of cores excluding those for interconnects. Recent multi-core chips are often implemented with large scale of network interconnects, which occupy almost half of on-chip area, such as TILE64 [5] and Intel Many Integrated Core Architecture [21]. In this section, we propose our new area cost model and incorporate interconnection into the area cost model and apply it to the speedup models we proposed in Sect. 2.

$$A(\psi) = m \times r + i \times \alpha \tag{8}$$

$A(\psi)$ represents the area of a chip. ψ is the tuple of architectural configuration. $\psi = (m, r, i, \alpha)$. i and α, respectively, denotes the number of interconnects and the number of transistors used to form an interconnect.

Before incorporating the cost model into our speedup models, we need to clarify conditions required to simplify the establishment and analysis of our speedup models. Since components other than interconnects such as the L2 cache do not grow exponentially with number of cores, we can assume that the resources spent on non-interconnect components are roughly constant in the multi-core variations. If each interconnect is linked to others via a full mesh, we do not need to take care of routing problems either. Each interconnect represents a clock domain connecting a certain number of cores and on-chip memories. We assume each clock domain has exactly the same number of cores and memories. Suppose memory access within a clock domain is fast enough to be ignored. So, the memory bound mentioned in Sun [11] can be presented as a kind of communication among clock domains. According to Moore's law, we also expect chip makers can take advantages of additional chip areas to create larger interconnects with better throughput and bigger core with better performance.

3.1 Amdahl's law under area constraint

The speedup of a symmetric multi-core chip depends not only on f_c^p and f_t^p, but also on allocation of silicon area to cores and interconnections. We do the following assumptions. On a multi-core chip, one core is assigned with tasks in sequential execution at performance $perf(r)$. m cores are used to execute in parallel at performance $perf(r) \times m$. One interconnect on the chip handles sequential transmission at performance $thrpt(\alpha)$, and i interconnects are used to transmit in parallel at performance $thrpt(\alpha) \times i$. Overall, we incorporate the assumptions above into Eq. (3) and get speedup as shown in Eq. (9).

$$S_A(\hat{f}, \psi) = \frac{1}{\frac{f_c^s}{perf(r)} + \frac{f_c^p}{perf(r)m} + \frac{f_t^s}{thrpt(\alpha)} + \frac{f_t^p}{thrpt(\alpha)i}} \tag{9}$$

3.2 Gustafson's law under area constraint

Similar to Sect. 3.1, we apply the same conditions and area constraints to Eq. (6). Then we drive S_G as a function of $(\hat{f}, \psi)$.

$$S_G(\hat{f}, \psi) = \frac{(f_c^s + f_c^p m)perf(r)}{(f_c^s + f_c^p)perf(r)}$$

$$\times \frac{\frac{f_c^s}{perf(r)} + \frac{mf_c^p}{perf(r)m} + \frac{f_t^s}{thrpt(\alpha)} + \frac{f_t^p}{thrpt(\alpha)i}}{\frac{f_c^s}{perf(r)} + \frac{mf_c^p}{perf(r)m} + \frac{f_t^s}{thrpt(\alpha)} + \frac{K(m)f_t^p}{thrpt(\alpha)i}}$$

$$= \frac{(f_c^s + f_c^p m)}{(f_c^s + f_c^p)} \times \frac{\frac{f_c^s + f_c^p}{perf(r)} + \frac{f_t^s}{thrpt(\alpha)} + \frac{f_t^p}{thrpt(\alpha)i}}{\frac{f_c^s + f_c^p}{perf(r)} + \frac{f_t^s}{thrpt(\alpha)} + \frac{K(m)f_t^p}{thrpt(\alpha)i}} \tag{10}$$

It is also necessary to add area constraints to speedup model. With Eq. (9) and Eq. (10), we can apply quantitative analysis onto the speedup of an application over multi-core architecture such as Larrabee [18], Teraflops [19], SSCC [20] and Intel Many Integrated Core Architecture [21].

4 Analyzing original models and our extensions

In this section, we analyse our speedup models under area constraints to see the impact of interconnection on speedup models and to find optimized architectural configurations, which could be helpful to architects at the initial phase of their design.

4.1 Upper bound of Hill and Marty's speedup model

Hill and Marty came up with their simple cost model as shown in Eq. (7), and added it into their speedup model. Equation (11) is an equivalent form of their speedup model under area constraint:

$$S_A(f,m,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{m \times perf(r)}}. \tag{11}$$

f represents the parallel workload of a task. r and m, respectively, represent the complexity and number of cores. Hill and Marty suppose architects can expend area of r to create a core with performance $perf(r)$. $perf(r)$ could be an arbitrary function. Pollack's rule [22] states that performance increase is roughly proportional to square root of increase in complexity. Hill and Marty follow typical choices [9], and assume $perf(r) = \sqrt{r}$. We follow their choices in all analyses in the rest of this paper unless specified.

Suppose total silicon area is given, and $area = m \times r$. We find the optimal value of m shown in Eq. (12) by solving the zero point of first-order derivative of Eq. (11) with respect to m. We substitute m in Eq. (11) with Eq. (12) and get Eq. (13).

$$m = \frac{f}{1-f} \tag{12}$$

$$S_A(f,m,r)_{\max} = \frac{1}{2}\sqrt{\frac{area}{f(1-f)}} \tag{13}$$

We know from these results that when silicon area and processing technique is given, the speedup with respect to a certain task will never go beyond its corresponding upper bound shown in Eq. (13). The optimal number of cores with respect to overall speedup is closely related to the parallelism of the task.

4.2 Upper bound and optimized configuration of our extended Amdahl's law under area constraints

The analyses in Sect. 4.1 are not precise enough in multi-core era, especially when lots of communication contentions occur among cores. Next, we are going to analyse the upper bound of our speedup model under area constraint shown as Eq. (9).

Suppose architects can expand area of r to create a core with performance shown as following:

$$perf(r) = \sqrt{r} \tag{14}$$

We treat transmission as a special form of performance similar to computations in terms of tasks completed within a time unit, i.e. messages or packages delivered by interconnect resources. Therefore, the relation of performance of interconnection and area is similar to Eq. (14):

$$thrpt(\alpha) = \sqrt{\alpha} \tag{15}$$

Under these assumptions, we try to solve zero point of partial derivative of Eq. (9) with respect to m. We find

$$m = \frac{f_c^p}{f_c^s} \tag{16}$$

Equation (16) shows the optimal number of cores. It means that when the silicon area and $\hat{f}$ is given, the optimal value of m is only related to the parallelism of computation workload of the task. It is independent of the parameters r, i and α. This important feature is later used on solving upper bound of Eq. (9).

A similar method can be used to find the optimal number of interconnects:

$$i = \frac{f_t^p}{f_t^s} \tag{17}$$

From previous analyses, we know that Eq. (12), Eq. (16), and Eq. (17) are similar. Moreover, the optimized configuration of cores and interconnection are independent. Similar to analyses made to m in Eq. (16) and i in Eq. (17), we separately calculate the minimum execution time of cores and interconnection according to Eq. (9). Then we incorporate these expressions of minimum values to form the upper bound speedup of the entire system.

$$\begin{aligned} S_A(\hat{f}, \psi)\max &= \frac{1}{\min(\frac{f_c^s}{\sqrt{r}} + \frac{f_c^p}{m\sqrt{r}}) + \min(\frac{f_t^s}{\sqrt{\alpha}} + \frac{f_t^p}{i\sqrt{\alpha}})} \\ &= \frac{2}{\sqrt{\frac{f_c^s f_c^p}{area_c}} + \sqrt{\frac{f_t^s f_t^p}{area_t}}} \end{aligned} \tag{18}$$

It would be intuitive that the minimum execution time of cores and interconnection can be calculated for any task according to Eq. (9). In Eq. (18), silicon area usage is defined as following:

$$area = area_c + area_t = m \times r + i \times \alpha \tag{19}$$

area is the total silicon area of a chip. $area_c$ and $area_t$, respectively, mean the silicon area used to deal with computation and transmission. By solving the zero point of the first-order derivative of Eq. (18) with respect to $area_c$, we find the optimal configuration of $area_c$ and $area_t$ that makes Eq. (18) achieve its maximum value.

$$area_c = \frac{area \times (f_c^p f_c^s)^{\frac{2}{3}}}{(f_t^p f_t^s)^{\frac{2}{3}} + (f_c^p f_c^s)^{\frac{2}{3}}} \tag{20}$$

$$area_t = \frac{area \times (f_t^p f_t^s)^{\frac{2}{3}}}{(f_t^p f_t^s)^{\frac{2}{3}} + (f_c^p f_c^s)^{\frac{2}{3}}} \tag{21}$$

With these optimized configurations, we get the optimal value of r and α:

$$r = \frac{area \times (f_c^p f_c^s)^{\frac{2}{3}}}{(f_t^p f_t^s)^{\frac{2}{3}} + (f_c^p f_c^s)^{\frac{2}{3}}} \times \frac{f_c^s}{f_c^p} \tag{22}$$

$$\alpha = \frac{area \times (f_t^p f_t^s)^{\frac{2}{3}}}{(f_t^p f_t^s)^{\frac{2}{3}} + (f_c^p f_c^s)^{\frac{2}{3}}} \times \frac{f_t^s}{f_t^p} \tag{23}$$

It should be noted that the analytical expressions Eq. (16), Eq. (17), Eq. (20), Eq. (22), Eq. (23), and Eq. (18) are all based on the assumptions Eq. (14) and Eq. (15). We draw some conclusions on these analyses as guidelines for architectural design.

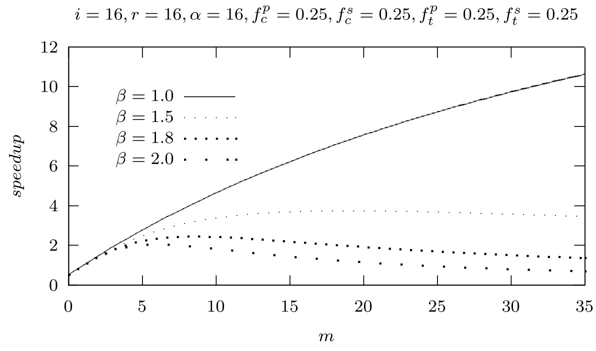
– For a fixed-size problem with given silicon area, the optimized number of cores and interconnects are respectively related to the parallelism of computation and transmission workload of a task.
– The upper bound speedup of architecture is closely related to the task running on it and the silicon area it costs.
– The area percentage spent on cores and interconnects is basically determined by the computation-to-transmission workload ratio of a task.

4.3 Optimized scaling of our extended of Gustafson's law under area constraints

Sun states in [11] that multi-core architectures are scalable under the scalable computing model. Sun considered that the fixed-time speedup of multi-core grows linearly with the scaling factor. In other words, an extra unit of area for cores provides a constant increment of speedup. Unlimited silicon area makes infinite increment of speedup possible. We are going to disprove it in this section and find the optimal value of scaling.

We follow the assumptions of *perf*(r) and *thrpt*(α) to analyse Eq. (10). $K(m)$ is a task specific function representing the growth rate of transmission workload generated by parallel computing. In many applications, algorithms have a polynomial complexity in terms of communication requirement. Similarly, Morad et al. [23] also

Fig. 1 Speedup vs. m



used polynomial function to model the time penalty of additional transmission. We can always take the highest degree term as β to represent the complexity of the algorithm. For any power function $K(m) = m^\beta$, we have

$$S_G(\hat{f}, \psi, \beta) = \frac{f_c^s + f_c^p m}{f_c^s + f_c^p} \times \frac{\frac{f_c^s + f_c^p}{\sqrt{r}} + \frac{f_t^s}{\sqrt{\alpha}} + \frac{f_t^p}{i\sqrt{\alpha}}}{\frac{f_c^s + f_c^p}{\sqrt{r}} + \frac{f_t^s}{\sqrt{\alpha}} + \frac{m^\beta f_t^p}{i\sqrt{\alpha}}} \tag{24}$$

In most cases, β is equivalent to or greater than 1. Multi-stage pipeline is a common example where β is close to 1. In worst cases, cores have to broadcast their computing result to almost all other cores, then β could be close to $m \times (m - 1)$. We choose β from 1 to 2 and plot a figure to see the impact of β on speedup.

Figure 1 is plotted following Eq. (24) under different case of β. When $\beta = 1$, the slope of the curve infinitely approaches $\frac{f_c^p}{f_c^s + f_c^p}$ as m increases. When $\beta > 1$, speedup raise to reach its peak value and turn to drop. Given larger values of β, we not only observe the degrading of peak value of speedup, but also the shifts of the peak value towards smaller m. In most cases of applications, β is greater than one. Obviously, communication contentions must be considered during scaling the number of cores of a system.

Scaling up the number of cores and the problem size is helpful to rise the speedup of computational workloads. However, the execution time required by new workload of transmission partially offset the speedup. To find the best scaling of problem and the number of cores, we are going to find optimal value of m. We define t_{saved} as the absolute value of execution time saved by scaling up the problem and the number of cores:

$$t_{\text{saved}} = \frac{f_c^p (m - 1)}{perf(r)} - \frac{(m^\beta - 1) f_t^p}{i \times thrpt(\alpha)} \tag{25}$$

In Eq. (25), the first term represents the execution time saved by parallel processing of computational workloads. The second term represents the execution time required by new workload of transmission. We consider the peak value of t_{saved} as the most efficient point that make best use of scalable computing model and prevent communication contention from excessively degrading the speedup of a system. Further increasing of m will cause the t_{saved} to turn gradually down to zero. We find the optimized m by solving the zero point of the first-order derivative of Eq. (25) with respect to m.

$$m = \sqrt[\beta-1]{\frac{f_c^p \times i \times thrpt(\alpha)}{f_t^p \times \beta \times perf(r)}} \tag{26}$$

Equation (26) together with Fig. 1 gives us some implications, which would contradict Sun's theory [11].

- An extra unit of area for cores does not provide a constant increment of speedup. An over-scaled number of cores could even degrading total speedup of architecture. The architect should carefully make decisions on proper m to maximize the speedup.
- Intercore communication modeled by β significantly affects the speedup of an architecture. Designer should put more effort on relieving the new transmission workload generated by scaling up task rather than blindly adding new cores to system.

5 A case study

In this section, we apply original speedup models and our extensions to a specific application, i.e. CT. The corresponding results will show that it is a must to explicitly consider inter-core communications in the speedup models of multi-core processors.

As a case study, CT is chosen to evaluate these models as it is a popular application that implies multi-level parallelism. It can also be easily scaled up to produce images of scalable resolutions. To apply speedup models onto the application, we need to extract task specific parameters f_c^s, f_t^s, f_c^p, and f_t^p from the application. As it is hard to derive these parameters directly from source code of the application, we introduce a task graph to evaluate the parallelism of the application.

5.1 Evaluating the parallelism of an application

A task graph is a directed acyclic graph (DAG) that consists of nodes and edges. It is a useful tool commonly used in the scheduling of multi-core or multi-processor systems [24–26]. The node and edge weights are usually obtained by estimation [27, 28] for which the parameters are determined using benchmark profiling techniques [29]. We use this task graph to quantitatively analyse the parallelism of task. A few methods [30, 31] have been proposed for evaluating parallelism of a task graph. These methods provide effective solutions to evaluate parallelism of specific applications so that speedup models can be put into practical use instead of theoretical analysis.

We do the calculation by following the above mentioned methods to derive the parameters of a typical CT calculation process as $f_c^s = 0.2$, $f_t^s = 0.1$, $f_c^p = 0.5$, and $f_t^p = 0.2$. It is noteworthy that though different parameters would be derived as there are different versions of source code and algorithms of CT with different resolution settings, variances in these parameters will not affect the correctness of our speedup models.

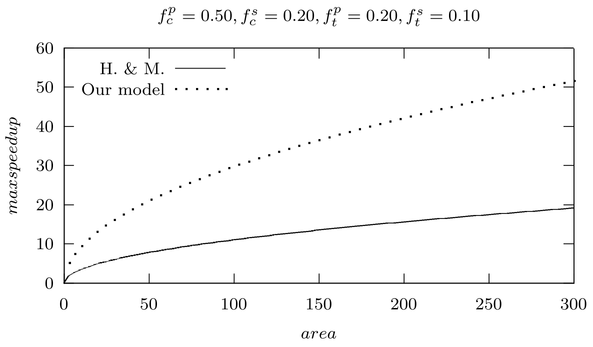


Fig. 2 Comparing upper bound of speedup with respect to Amdahl's law. The curves of *solid line* and *dashed line* are respectively calculated according to Eq. (13) and Eq. (18)

Table 1 Implementing CT algorithms on stand-alone FPGA based systems [17]

Hardware	Speedup	#transistors [32]	H. & M.	Ours
XC5VLX110T	100X	1100M	37X	103X
XC2VP30	50X	430M	23X	64X
Intel Pentium 4	1X	42M	7X	20X

5.2 Comparing speedup models with respect to Amdahl's law

We plot Fig. 2 according to Eq. (13) and Eq. (18) using the parameters calculated above. Thirty percent of the workloads of this application are transmissions. The solid curve blow represents Hill and Marty's old model while the dotted curve above represents our model. Intuitively, our model is more pessimistic. But this figure shows that our model promises larger speedup than Hill and Marty's model does. This is because Hill and Marty only exploit the parallelism of computation. In our model, more parallelism can be exploited from the workload of transmission.

Here is an example of FPGA implementation for CT [17]. Zhiqiang et al. implement CT algorithms on PC and two FPGA based systems, respectively. Since FPGA implementation are strong application-specific, we can approximately consider that their performances are maximized. Table 1 shows the details. The column of "speedup" shows the actual speedup measured by the authors. We take the number of transistors as silicon area and use old models and our extension to estimate the speedup. As shown in the last two columns, the speedup estimated by our model preserves ratio between different hardware compares to that of Hill and Marty's. Our estimation is more close to the actual speedup especially when silicon area is large. In this case, Hill and Marty's model under-estimate the speedup because of ignoring the impact of interconnections.

5.3 Comparing speedup models with respect to Gustafson's law

Figure 3 is plotted according to Eq. (5) and Eq. (24). We suppose that $r = 4$, $i = 4$, $\alpha = 4$, $\beta = 1.33$. Our model also works well on other values, but this set of values clearly show the properties of our model. The curve of the dashed line, which represents the speedup model of our extensions, quickly reaches its upper bound and

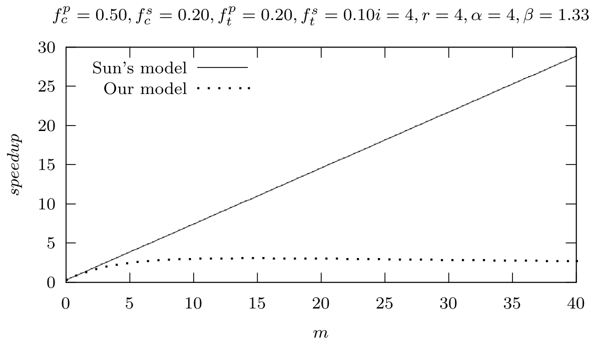


Fig. 3 Comparing speedup models with respect to Gustafson's law. $r = 4$, $i = 4$, $\alpha = 4$, $\beta = 1.33$

Table 2 Keck et al. [33] implement SART on both QuadroFX 5600 and Tesla C1060

Hardware	#CUDA cores	Time	Speedup	Sun's	Ours
Quadro FX5600	128	1448 s	1.00X	1.00X	1.00X
Tesla C1060	240	955 s	1.516X	1.63X	1.51X

decreases slowly. This tendency is much more pessimistic than Sun's theory reflected as the solid line in Fig. 3. According to Eq. (26), $m = 451.8508$ achieves the maximum value of execution time. Sun's model shows an over-optimistic view on the speedup. According to our model, we do enumeration and find that when $m \approx 17$ the maximum *speedup* ≈ 3.5 is achieved. There is a large gap between these two points of interests. Computer architects are suggested to explore all the values in this gap as reasonable candidates to scale up the number of cores in their design.

We present another example of the CT algorithm based on GPU. As shown in Table 2, Keck et al. [33] implemented the Simultaneous Algebraic Reconstruction Technique (SART) on both QuadroFX 5600 and Tesla C1060 using CUDA 2.2. The column of "#CUDA cores" shows the number of CUDA cores (similar to CPU core) in each hardware. The column of "Time" and "Speedup", respectively, represents the time consumption and actual speedup measured by the authors. We take the number of cores and use the old model and our extension to estimate the speedup. From the last line of the table we know that the speedup estimated by our model is more close to the actual speedup compares to that of Sun's model. It seems that Sun's model is not that over-optimistic in this case. This is because Tesla C1060 is a great video card dedicated for super-computing. Its internal interconnections are elaborately designed and enhanced to match the super computing power of 240 cores. If its interconnection cannot afford the workload of transmission, Sun's model will certainly over-estimate the speedup and fail.

6 Conclusion

To address the new practical issues of communications in multi-core architecture design, we reexamined and extended Amdahl's law and Gustafson's law. We explicitly modeled communication time to divide task execution into four parts: the parallel and sequential workload of computation and transmission. We incorporate the interconnection into Amdahl's law and Gustafson's law to evaluate how the interconnects affect the overall speedup of a system. Then we propose a hardware cost model in which the area of cores and interconnects are both taken into consideration. We apply our hardware cost model to our speedup models and demonstrate the analyses of our speedup models under area constraints. According to these analyses, we derive several analytical expressions of optimal configuration parameters with respect to architectural design. Finally, we demonstrate a case study to compare the speedup models of our extensions and their original counterpart. This case proves that inter-core communications must be explicitly considered in the speedup models of multi-core processors.

Acknowledgement This paper is partially sponsored by the National High-Technology Research and Development Program of China (863 Program) (No.2009AA012201).

References

1. Amdahl GM (1967) Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of the April 18–20, 1967, spring joint computer conference. ACM, New York, pp 483–485
2. Gustafson JL (1988) Reevaluating Amdahl's law. Commun ACM 31(5):532–533
3. Borkar S (2007) Thousand core chips—a technology perspective. San Diego, CA
4. Furber S (2008) The future of computer technology and its implications for the computer industry. Comput J 51(6):735–740
5. Wentzlaff D, Griffin P, Hoffmann H, Bao L, Edwards B, Ramey C, Mattina M, Miao CC, Brown Iii JF, Agarwal A (2007) On-chip interconnection architecture of the tile processor. IEEE MICRO 27(5):15–31
6. George L (2009) More Cores Keep Power Down. "Computing now". from. http://www.computer.org/portal/web/computingnow/archive/news041
7. Semiconductor Industry Association (2007). International Technology Roadmap for Semiconductors. From http://www.itrs.net/Links/2007ITRS/Home2007.htm
8. R CJ (2007) Intel's teraflops chip uses mesh architecture to emulate mainframe. EETimes Product Brief
9. Hill MD, Marty MR (2008) Amdahl's law in the multicore era. Computer 41(7):33–38
10. Woo DH, Lee HHS (2008) Extending Amdahl's law for energy-efficient computing in the multi-core era. Computer 41(12):24–31
11. Sun X-H, Chen Y (2010) Reevaluating Amdahl's law in the multicore era. J Parallel Distrib Comput 70(2):183–188
12. Sinnen O, Sousa LA (2005) Communication contention in task scheduling. IEEE Trans Parallel Distrib Syst 16(6):503–515
13. Benoit A, Hakem M, Robert Y (2009) Contention awareness and fault-tolerant scheduling for precedence constrained tasks in heterogeneous systems. Parallel Comput 35(2):83–108
14. Sinnen O, To A, Kaur M (2011) Contention-aware scheduling with task duplication. J Parallel Distrib Comput 71(1):77–86
15. Guo M, Nakata I, Yamashita Y (2000) Contention-free communication scheduling for array redistribution. Parallel Comput 26(10):1325–1343
16. Xiaoyong T, Kenli L, Degui X, Jing Y, Min L, Yunchuan Q (2006) A dynamic communication contention awareness list scheduling algorithm for arbitrary heterogeneous system. Montpellier
17. Zhiqiang Q (2010) Implementing medical CT algorithms on stand-alone FPGA based systems using an efficient workflow with SysGen and simulink. In: Yongxin Z, Xuan W, Jibo Y, Tian H, Zhe Z, Li Y, Feng Z, Yuzhuo F (eds) Computer and information technology (CIT), 2010 IEEE 10th international conference, pp 2391–2396

18. Larrabee, http://en.wikipedia.org/wiki/Larrabee_(microarchitecture)
19. Teraflops Research Chip, http://en.wikipedia.org/wiki/Teraflops_Research_Chip
20. Single-chip Cloud Computer, http://en.wikipedia.org/wiki/Single-chip_Cloud_Computer
21. Intel Many Integrated Core Architecture. http://en.wikipedia.org/wiki/Intel_MIC
22. Pollack's Rule, http://en.wikipedia.org/wiki/Pollack's_Rule
23. Morad TY, Weiser UC, Kolodny A, Valero M, Ayguad E (2006) Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors. IEEE Comput Archit Lett 5(1):14–17
24. Vee V-Y, Hsu W-J (1999) Applying cilk in provably efficient task scheduling. Comput J 42(8):699–712
25. Kwok Y-K, Ahmad I (1999) Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput Surv 31(4):406–471
26. Kwok Y-K, Ahmad I (1996) Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors. IEEE Trans Parallel Distrib Syst 7(5):506–521
27. Cosnard M, Loi M (1995) Automatic task graph generation techniques. In: Proceedings of the twenty-eighth Hawaii international conference on system sciences, vol. II
28. Wu MY, Gajski DD (1990) Hypertool: a programming aid for message-passing systems. IEEE Trans Parallel Distrib Syst 1(3):330–343
29. Hwang K, Xu Z, Arakawa M (1996) Benchmark evaluation of the IBM SP2 for parallel signal processing. IEEE Trans Parallel Distrib Syst 7(5):522–536
30. Jereb B, Pipan L (1992) Measuring parallelism in algorithms. Microprocess Microprogram 34(1–5):49–52
31. Jain KK, Rajaraman V (1994) Parallelism measures of task graphs for multiprocessors. Microprocess Microprogram 40(4):249–259
32. Transistor count, From Wikipedia http://en.wikipedia.org/wiki/Transistor_count
33. Keck B, Hofmann HG, Scherl H, Kowarschik M, Hornegger J (2009) High resolution iterative CT reconstruction using graphics hardware. In: Nuclear science symposium conference record (NSS/MIC). IEEE, New York, pp 4035–4040