page1--各位老师同学大家上午好。我的毕业论文的题目是多核芯片的温敏低功耗调度研究。这篇文章也基本上是对我研究生期间的研究成果的一个总结

page2--我研究生期间主要对多核芯片温敏低功耗调度的两个问题进行了研究。(next)其一是芯片的动态功耗与温度管理，(next)其二是多核芯片的热分析方法

page3--首先我们进入第一个问题：对芯片的温敏低功耗调度

page4--从调度目的本身来讲，我们需要对调度目标有一些评测、估计的理论。(next)所以首先要介绍温度与功耗的分析模型。接着在已有的调度管理办法的基础上，本文提出了一种新颖的VP-TALk算法，进而又构建了一个DPTM原型系统，最后进行了充足的模拟实验来验证。

page5--首先是温度分析模型：这个领域的多数研究者都采用了电热等效和RC等效电路的方法描述芯片上，温度和功率的关系。(next)这个公式是一个典型的RC充放电电路。功率作为热流，类似于电路中的电流；温度则类似于电压。

page6--功耗计算模型：功耗可以分为两部分，(next)动态功耗和静态功耗。动态功耗是芯片工作所产生的功耗，静态功耗则是由于工艺提升后，漏电流所引起的那一部分功耗。根据前人的工作，已经有了这个依赖于芯片工艺参数的漏电流拟合公式(next)。在芯片的各个工作状态之间切换需要有一定的功率损耗。这个损耗值在依赖于切换工作状态的调度过程中是不可忽略的。所谓切换工作状态，可以理解为调节工作电压，所以切换损耗自然和切换前后的电压差成正相关关系

page7--(next)下面是在2010年到2012年之间发表出的一些调度管理方法。下面会简要的介绍一下各种已有算法。(next)而本文的工作就是在这些算法的基础上，(next)更好的利用DVS，来达到调度效果的提升

page8--一般芯片会分给某个任务一个合理的时间片窗口D，任务在最坏情况下的执行耗时为W。时间片一定是大于任务耗时的，所以有一种调度算法就把富裕的时间均摊在若干个小窗口中。这样做是为了让芯片在整个任务执行过程中能有间隔的适量的休息时间，从而拉低升高的温度。通过一定的数学推导可以去找到一个最优的小窗口个数

page9--在上面均摊想法的基础上，我们可以利用芯片的DVS功能，让芯片一直工作，但是工作在两个不同的强度下。高电压对应高强度，产生高功率，温度进而飙升；低电压工作状态下，功率被拉低，从而使得芯片有机会喘息，温度就会降低。如何选择两个合适的工作状态，如何确定应该分出多少个小时间窗口，都可以通过一定的数学推导得到

page10-最后这种方法更动态，或者说更注重在线调度。在每一个小时间片内，考虑两个指标：一个是剩余工作量与剩余大时间片的比例，另一个是芯片继续工作的情况下温度上升的速度与芯片马上停止工作的情况下温度下降的速度的比值(next)。这就是说在任务的实时性和芯片的温度之间做在线的比较，如果任务的实时性更紧迫，那么芯片继续工作；如果任务的实时性不那么紧迫，或者芯片温度蹿升的速度过快(next)，就要让芯片休息

page11-事实上TALk算法是最早提出的，M阶震荡反而是后提出的。本文提出了一种结合这两种算法的VP-TALk算法，其实就是说，在TALk的每个时间片内采用M阶震荡的两个工作状态，而整个大的时间片内，在线的使用TALk算法进行调度。(next)我们可以从图中看到本文算法的能耗调度改进效果，平均的能耗节省可以达到14%左右

page12-进而，我们把所有这些介绍的算法组合起来，构建了一套DPTM原型系统。我们先来看一看构建该系统的动机，然后在介绍系统中除了调度算法之外的其他组成部分

page13-构建原型系统的动机其实是基于负载率和调度算法的这样一种关系：我们看到负载率低于50%的时候最原始的固定周期的调度算法效果较好，而负载率超过50%后，TALk和M阶震荡分别成为了更好的调度算法。于是我们希望能有一个系统来根据负载率大小自动在线选择效果好的调度算法

page14-基于以上动机，我们的原型系统应该以"下一个时刻"的负载率为输入，然后动态的选择效果优良的调度算法。但是现实环境中我们往往不能已知下一个时刻的负载率，而只能根据历史负载状况预测、估计。这里本文采用我在本科毕设阶段所提出的一种组合成分分解预测模型：该模型假设任务量主要有三个独立的成分累加而成，(next)其一是趋势成分，(next)其二是周期成分，(next)最后是随机成分。这三种成分我们分别用最合适的数学模型建模。比如，(next)趋势成分用指数灰度模型拟合,(next)周期成分我们用傅里叶序列拟合，(next)随机成分用径向基函数神经网络拟合，通过这些模型获得各个成分的未来估计量，再累加到一起，得到整体任务的未来预测值

page15-有了负载率后，我们就需要原型系统能够在线评估各种调度算法，从而在线选择优良的调度策略。我们分配给每个调度策略一个得分，(next)在某一个时刻，这个得分刻画算法的历史的能耗情况，在一个峰值温度的约束下，(next)我们挑选一个历史最优的调度算法

page16-本文通过一组模拟实验测试了所提出的原型系统。在在线的情况下，所有调度算法会有一个平均的调度结果。而在离线情况下，每个时刻我们可以确定一个理想的调度算法。这两个对象就是本文提出原型系统的对比对象。本文中的第三章第三节有详细的对比，说明原型系统的调度效果优于源算法的平均调度效果。这里只给出与理想效果的对比结果。

page17-能量消耗方面，该系统和理想调度效果几乎很接近，有很大规模的重合

page18-而峰值温度，在60摄氏度以下很大的差别，但是幸好这个温度范围对于芯片并不是很紧要。在高于70摄氏度的情况下，本文的原型系统与理想效果的差别就不那么明显了

page19-OK，这就是本文的第一部分。

page20-下面我们进入本文的第二部分，针对多核芯片的热分析方法。

page21-首先需要简单介绍一下对芯片散热系统的建模。(next)然后介绍目前在科研领域很流行的开源热分析软件HotSpot。接着，这里还需要强调一下漏电流对芯片温度的影响。最后详细介绍本文所提出的热分析方法，以及通过实验验证的精度和速度效果

page22-首先让我们看看芯片的散热系统。(next)热量由内核Die产生，这些热量有两个流通通道，也就是散热通道，我们分别叫做主散热通道和辅散热通道。在主散热通道中，热量通过TIM介质，导热层直到散热片，完成流通到芯片外。而通过衬底、印刷电路板等介质的这条辅散热通道，它的散热能力和主通道会差一两个数量级。所以我们在建模中以主通道为主，忽略了辅通道。

page23-学术界较为公认的HotSpot采用的仍然是电热等效原理，通过构建等效热容和等效热阻组成的等效电路(next)，然后采用分析电路的手段，比如基尔霍夫定律，来得到热分析结果。我们可以看看它是如何对主散热通道建模的。内核层的每一个模块对应到TIM介质层有一个热阻，然后这些热流全部汇集到扩热层的中心模块上。从扩热层到散热片仍然是类似的从上向下构建等效热阻，最后外界环境设定为恒定的45摄氏度，类似于电路中的接地。整个电路采用基尔霍夫定律，(next)通过求解矩阵方程获得温度。之前的介绍中我们已经看到，漏电流功耗和温度具有着一个拟合关系。这就导致矩阵方程的左边的未知量T与后边的已知量P是会相互影响的。

page24-一种很直观的解决办法就是迭代地计算刚才的矩阵方程，直到温度矩阵和功耗矩阵都达到稳定。通过实际的实验操作我们看到，大概迭代6-7次温度和功耗就才会收敛，而且温度逐渐上升的程度并非可以忽略的，也就是说不考虑电热耦合效应，会得到过于乐观的温度分析结果；而另一方面，如果考虑漏电流对温度的影响，就会是正常分析时间的6-7倍

page25-这就迫使我们提出更快速而且准确的热分析方法。本文对芯片从建模上的简化出发，提出了这三种结构级热分析模型。(next)下面就一一介绍这三种方法

page26-首先是模块级的稳态热分析方法。我们抛开扩热层、散热片这些我们不关心的对象，直接对内核按照逻辑功能进行模块划分(next)。每个功能模块之上都会被施加一个功耗作为热流。单独考虑一个功能模块，我们认为它的温度之所以上升，一方面是施加在其上的热流所导致，另一方面是因为所有其他模块对它的影响所导致；更进一步地，我们假设，模块自身的热流使它的温度增长是线性的，而且其他模块的热流也使该模块有线性增长。这两个线性增长趋势分别被称为该模块的的自热阻，和模块间的互热阻。我们通过HotSpot经过若干次的单位功率采样，很容易就可以得到自热阻与互热阻组成的热阻矩阵。有了热阻矩阵之后，任何的功率输入都只需要线性叠加操作，就可以得到温度分布。

page27-接着，我们在进一步简化核内的功能模块。一般热分析的目标就是处理器核心的温度，因为往往核心的温度是最高的。我们采用上面模块级同样的方法，但是用核心代替整个处理器。但是这样的简化有些过头，后面的试验也证明这会导致过大的分析误差。

page28-第三种方法就是特别针对这样的情况而提出。对于某一个功能模块，一方面我们考虑该模块所在处理器内部的所有其他模块对它的影响；另一方面，外部模块则统一为一个处理器对该模块的影响。在具体计算上，其实就是一组互热阻的加权平均。这样，考虑的温度上升因素，也就是计算量，是介于考虑所有模块间的影响和所有只考虑处理器之间的影响之间的。

page29-最后我们来看一看(next)这三种方法的计算精度和分析效率。

page30-首先是多个计算精度实验中的一组结果数据。这是一个4处理器的floorplan，100组功率输入分析误差的平均值。我们抛开峰值温度不占主导的cache模块，仅仅关注核心模块的分析误差。简化程度最高的方法CoreTAM误差最大，简化程度最低的BlockTAM误差最小，三种方法误差都低于8%，当然比较优良的BlockTAM对所有模块都能达到仅3%左右的误差。此外，我们看到，BlockInsideCoreTAM的核心热分析精度和Cache热分析精度有着迥异的差异

page31-光有理想的分析精度还不够，我们做了如此多的模型简化过程，很大的动机是为了加速热分析过程。这组实验的实验对象是一个16处理器的floorplan，分析1000次统计出来的时间数据。由于本文的方法都需要利用HotSpot提取热阻系数矩阵，我们把总耗时分为分析和提取两部分，这样能够突出分析加速比。分析加速比很理想，能达到50-60倍的加速，但是把提取时间均摊进来，加速就降低到了13-14倍。

page31-总而言之，这篇文章做了如下三项工作：提出一种新颖的调度算法，构建了一套原型系统，探索了结构级热分析方法。

page32-最后，就像我在致谢中写的那样，要感谢所有支持帮助我的人，也谢谢答辩委员会的各位老师。