具有同步/异步混合状态存储器的低功耗有限状态机综合工具

C、 曹，M.O&apos;Nils和B.Oelmann

文摘：为了获得低功耗的有限状态机，一种有效的方法是将有限状态机分成两个子状态机或多个子状态机，然后采用动态功率管理，关闭所有未激活的子状态机，从而降低动态功耗。因此，FSM分区算法和寄存器传输级功率估计函数是本文的重点，因为这是设计低功耗分段FSM计算机辅助设计工具的关键问题。目标是一个实现体系结构，它基于同步和异步状态内存元素，与完全同步体系结构相比，这种结构能够实现更大的功耗降低。在面积增加18%的情况下，功率降低了77%。



1          介绍

体系结构级别的功耗优化通常涉及某种动态电源管理（DPM）方案，以降低动态功耗[1]。无论何时应用DPM，都必须将原始设计分成两个或多个单元，以便在空闲时动态关闭它们。一个自动优化过程将原始设计描述和主要输入信号的统计信息一起应用到一个具有成本函数的分区算法中，以寻求最佳划分。非琐碎问题的可能分区（候选）的数量太大，无法探究。因此，需要一种只选择最有希望的候选人的算法。在这些候选人中，有一个应该是最好的。在这方面，尽管缺乏关于最后执行情况的详细资料，但必须有准确的费用函数。

2          背景

对于分区有限状态机（FSM）设计的大多数方法，最初的设计描述是同步状态转移图（STG）。在STG上进行划分、成本估算和转换。第一步通常是识别具有高相互状态转移概率的状态簇。据说这些州是紧密相连的。目标是寻找具有强连接状态的小簇，因为它们将导致小的子fsm，这些子fsm大部分时间是活跃的，并且导致低平均功耗。每一个子FSM都需要一个空闲状态检测电路和一个关闭机制，两者都构成了一个功能开销。分裂者必须寻求最有利的

|  |
| --- |
| #工程技术学会2006年IEE在线论文集第20050048号doi:10.1049/ip-cdt：20050048号  该文件于2005年3月13日首次收到，修订格式为2005年9月15日  作者来自信息技术和媒体部，  瑞典中部大学，SE-85170 Sundswall，瑞典  电子邮件：曹操@密斯 |

空闲状态，考虑到这个开销。在Benini等人的早期工作中。[2] ，将具有高跃迁概率的所谓自循环作为独立的子fsm来实现。这项工作被概括为涉及许多状态的簇[3]。分区FSM中引入的主要功率开销来自这样一个事实，即在交叉转换（从驻留在另一个子FSM中的状态而不是目标状态的状态转换）的情况下，必须在该周期内对两个子FSM进行计时，以完成转换，这使得转换非常昂贵[4]。

Oelmann等人。[5] 已经证明，通过在每个sub-FSM中引入多个退出状态，sub-FSM不需要返回到单个的“reset state”[3]，从而可以消除双时钟要求。通过允许异步状态更改和具有多个退出状态，可以在同一个时钟周期内进行两个状态更改。使用异步控制的另一个优点是自由运行的全局时钟上的电容负载减少，随后可以降低功率。这种方法可以看作是在寄存器传输级（RT级）采用同步/异步混合方法进行FSM低功耗设计。与Chapiro[6]的工作不同，在这里全局异步局部同步系统的环境是异步的，混合模式电路是外部同步的，而内部是同步和异步方法的结合。

实现分区FSM的直接方法是为每个子FSM具有单独的状态存储器[3]（图1a）。或者，由于每次只有一个子fsm处于活动状态，所以状态存储器可以被所有子fsm共享[7]。这里的主要优点是减少了触发器的面积。然而，在这种情况下，需要全局状态来确定子fsm中的一个是活动的（图1b）。全局状态存储器（GSM）需要由全局时钟来计时，这大大增加了功耗。

与Oelmann等人相比，为了进一步探索异步电路的潜力。[5] 而Chapiro[6]在异步部分仅作为接口的情况下，本文讨论的CAD工具针对的是混合型

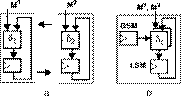


图1有限状态机a独立状态存储器b共享状态存储器的结构分解

同步/异步体系结构，由Cao和Oelmann[8]开发，具有共享的同步本地状态存储器（LSM）和全局异步状态存储器。其基本思想是在始终时钟部分（即LSM）中有一个同步存储器，以及用于GSM的异步存储器，后者更新的概率很低。这样，GSM只增加了非常低的功耗开销。所使用的关闭机制是输入门控以减少空闲组合逻辑中的功耗，门控时钟用于关闭LSM中暂时不需要的触发器。尽管引入了异步电路，但分区FSM的输入/输出行为仍然是一个周期地与原始的未分区同步机制相同。

3设计模型介绍

在我们的设计模型中，分区子fsm共享同一个同步LSM，其中一个子fsm的控制由异步GSM控制。

# 3.1 STG改造

为了处理交叉转换，原始STG（代表单片FSM的行为）被转换为支持异步激活和停用子FSM的交互方案。图2中的示例用作STG变换的图示。

在FSM分割之后，图2中的STG被划分为两个子FSM（M1和M2）和状态子集

M1中的S1¼fs1、s4、s6g和M2中的S2¼fs2、s3、s5、s7g。M1和M2之间有三个交叉过渡。为了检测交叉跃迁，引入了一个额外的g态。g状态位于子有限状态机内部，该子有限状态机包含交叉转移的源状态，并且与目标状态具有相同的索引。STG变换后，在M1中形成了U1¼fs1、s4、s6、g2g和M2中的U2¼fs2、s3、s5、s7、g1、g6g两个新的状态子集。转换后的STG如图3所示，其中为清晰起见忽略输入/输出。

引入g态后，交叉跃迁的行为发生了变化。乘坐交叉路口转换s6！以s2为例（图3）。在M1中引入g2之后，第一个事件是转换s6！g2，M1内。然后，在第二个事件中，g2的检测使异步状态存储器将其状态从r1更新到r2（在边缘标记为r21，rþ2）。全局状态r1和r2分别表示激活的子fsm M1和M2。异步转换完成后，M1失活，M2激活。

整个交叉转换在一个时钟周期内完成。第一个事件是同步的，因为LSM在时钟信号的活动边缘被更新到g状态。应区分状态s6和g2

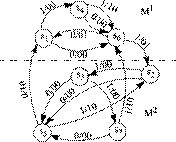


图2 FSM示例dk27

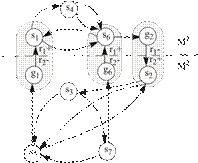


图3分解后的有限状态机中变换后的STG

当共享相同的全局状态r1时，从本地状态代码。第二个事件是异步的，因为GSM在检测到g状态转换后立即更新到r2。LSM仅由时钟信号触发，因此保持不变。在这个例子中，g2和s2应该共享相同的本地状态码，而它们的全局状态是不同的。

具有相同的局部状态码和不同全局状态的一组状态称为状态束。包含g-state的statebundle称为g-state-bundle。在图3中，有三个g状态束（g1，s1），（g2，s2）和（g6，s6），用灰色阴影区中的圆圈表示。

# 3.2状态绑定和编码

在3.1节中，我们通过一个例子提出了g状态束和状态束的概念。状态绑定的原因是：（1）它使状态能够共享相同的本地状态码；（2）它支持一种高效的异步切换机制；（3）g状态束实现了一种高效的时钟选通实现。

在STG转换之后，建立了一个状态束表。表的每一列表示一个状态束，包括一组具有相同本地状态码但具有不同全局状态码的状态集。表的每一行表示子FSM中具有相同全局状态码的状态。行数与子FSM的数量相同。由于g-state-bundle中的状态应该放在同一列中，所以我们从g-state-bundle开始构建表。

具体地说，让我们再次检查图3中的示例。它的状态束表由两行组成，分别表示M1和M2，max（jU1j，jU2j）¼6列，表示单个sub-FSM中更多的状态（也包括g状态）。首先，将三个g态束放入表单元（灰色阴影）。每个子FSM中的其他状态从最左边的空单元依次放入表中。造好桌子后，六

表1：状态捆绑表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| B | 电话号码 | 地下一层 | b2 | b3 | b4号 | b5型 |
| M1 | s1 | s6 | g2组 | s4 |  |  |
| 平方米 | g1号 | g6 | s2 | s3 | s5 | s7 |

得到了状态束，子有限状态机内部的状态转移可以看作是状态束的跃迁。

让我们来观察交叉过渡s6！再次是s2。从表1可以按以下顺序解释这种转换：（1）LSM内部从状态束b1到b2的水平状态转换，（2）GSM中从M1到M2的垂直状态转换。注意，状态包仍然是b2。

二进制代码按递增顺序从左到右分配给列。环境确保异步GSM以基本模式运行[9]。通过对全局状态使用指定的一个热编码，确保了一个无危害的实现。

# 3.3实施结构

假设单片机以I为输入，O为输出，分为子FSMs M1，M2。。。，明尼苏达州。原始状态子集S1，S2。。。介绍了新的状态组合U2-g状态。。。，对于M1、M2。。。，分别为Mn。所有子有限状态机共享相同的LSM，但有自己的组合逻辑。我们分解的FSM结构模型如图4所示。

g状态束检测逻辑（GDL）对LSM中的状态位进行解码。如果检测到g-state-bundle，则向GSM发送信号。

GSM决定当前激活的子FSM。它被实现为异步有限状态机。GSM中的状态转移仅在发生交叉转移时发生，即当检测到g状态时。在一个“分区良好”的FSM中，交叉转换的概率很低，GSM在大部分时间内都是空闲的，因此不会消耗动态功率。GSM中的状态信息直接用作LSM和子FSM（标记为M1，…）的组合部分（实现下一个状态功能）的控制信号。。。，图4中的Mn）。

对于主动子FSM-Mi，由于所需的本地状态位的数量可能小于本地状态位的总数，因此当前未使用的触发器通过时钟选通来禁用。全局状态控制时钟选通。

在任何给定时间，除了交叉跃迁事件外，只有一个子FSM处于活动状态。主动子有限状态机负责确定主输出和下一个局部状态。当不活动时，它的所有输入将被和门禁用，并且不会消耗任何动态功率。非活动副FSM的所有输出都设置为零。通过使用或门，从所有子fsm中采集相应的输出，可以得到正确的主输出和下一状态输出。

子有限状态机组合逻辑中的状态比特数对其实现规模和功耗有重要影响。FSM的这种分区导致子FSM的状态码和输入比特的数量减少。因此，可以实现面积和功率的降低。当发现一个小的子FSM大部分时间处于活动状态时，可以获得很大的功率降低。

关于基于同步/异步状态存储器的分区有限状态机实现的更多细节，我们让读者参考Cao和Oelmann的工作[8]。

4          设计流程和工具

如图5所示，该工具接受kiss2格式的标准基准[10]，其行为可以描述为未优化的同步stg。对于每个输入，给出了开关量和信号概率。假定了一个基于标准单元的设计流程，这意味着对于库没有超出通常提供的特殊要求。然而，该工具需要一些与单元库相关的信息，以便对异步元件进行准确的功率估计和门级综合。

|  |
| --- |
| 图4混合同步/异步FSM结构图5工具概述 |

为了实现功率估计，第一步是为有限状态机生成必要的统计信息。从行为有限状态机描述（STG）和初始输入概率出发，得到状态转移概率、状态概率和输出统计量、转移函数和输出函数。根据它们之间的相互状态转移概率，对状态进行聚类。然后使用一个算法来选择最有可能给出最佳划分的候选对象。在候选数目有限的情况下，可以进行更精确的RT级功率估计。每个候选项被合成为RT级描述，并估计功耗。从这些结果中，选择最佳候选，并在标准工具中生成RT级超高速集成电路硬件描述语言（VHDL）代码和用于逻辑综合的综合脚本。

5          分区FSMs的自动合成

# 5.1状态聚类

原始状态转移图可以看作是一个边加权无向图G（V，E）。二叉树是通过递归应用Kernighan–Lin双向划分建立的，其中状态根据其状态转移概率进行聚类，以最小化两个集合之间的交叉转移。在后期阶段被丢弃的冗余状态最初被引入V，形成V0，以确保jV0j（顶点数）是2的幂次方。假设jV0j¼2n，该算法的复杂度为O（n2 logn）。为了第二阶段的好处，树被建立在具有较高静态概率的左边的簇。每个级别上最左边的簇具有最高的静态概率。以基准dk27[10]为例，它有七个州。在引入一个冗余状态（8）之后，如图6所示构建完整的二叉树。

# 5.2候选生成

我们提出了一种有效的算法，在二叉树的每一级合并簇来生成分区候选。对于n个状态，该算法寻找从1路到n路划分的候选，其复杂度仅为O（nlog3n）。在有限的候选数目内，可以找到一个低功耗的良好分区。在图6所示的二叉树上应用图7中给出的算法，如图8所示生成候选。



图6 dk27全二叉树

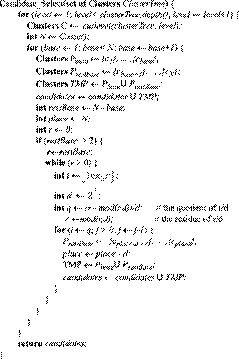


图7选择候选对象的算法

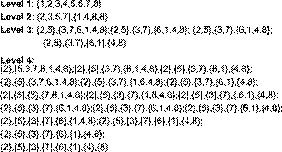


图8 dk27生成候选图

# 5.3功率估算

功率估计函数用于第5.2节中获得的分区候选，以找到具有最低功率的最佳分区。对于异步全局和同步本地状态存储器，门级实现是已知的。它与组合逻辑不同，因为它需要不同的功率估计技术。通过STG模拟器，得到输入和输出的统计信息，并用于功率估计。

5.3.1组合逻辑的功率估计：组合逻辑使用了Nemani和Najm[11]中提出的基于熵的功率估计方法。基于输入和输出的切换活动，使用了组合逻辑的转移表和熵

Xn Pcomb¼高rowi ktech Tcomi

i¼1

其中n是子FSM的数量，Hi是逻辑的熵，rowi是源于子FSM Mi的状态转换表中的行数，ktech是根据经验确定的常数，用于调整所使用的单元库，tcomt Mi激活的概率。我

5.3.2 GSM的功率：对于GSM，我们使用基于存储器结构的经验模型。虽然门级实现是已知的，但我们发现使用下面所示的宏模型更准确，它由两部分组成：括号中的表达式表示检测并启动从一个子FSM到另一个子FSM的转换的逻辑能力，以及表示异步状态记忆元件的求和项

PGSM¼ðkB pLSMBþkG pGþkG jgjÞ

XnþPC TCi

i¼1

括号内的表达式估计全局状态转移函数的幂，该函数是局部状态和全局状态的函数。第一项表示来自LSM的贡献，其中pLSM2B是本地状态位的切换概率。第二项表示来自全局存储器的贡献，其中pG是g状态的切换概率之和。g-state是启动全局状态转换的局部状态。第三项表示全局状态转移逻辑的复杂性，其中jgj是g状态的数目。总和项表示来自GSM设备的贡献，实现为Muller C元素，其中tc是我

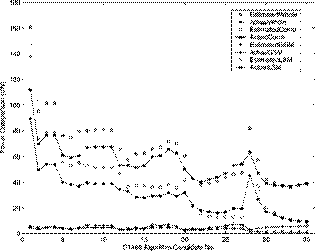


图9成本函数验证

全局状态转移的概率，即与次FSM Mi相关的交叉跃迁的概率。C元素的数量与子FSM的数量相同，表示为n。常数kB、kG和kG是根据经验确定的，并且基于单个FSM分区运行。

5.3.3 D触发器/触发器的功率估计：LSM由一组D触发器组成，通过

Xm PLSM¼PDFFi TDi

i¼1

其中tdi是触发器i激活的概率，m是LSM位的数目。我

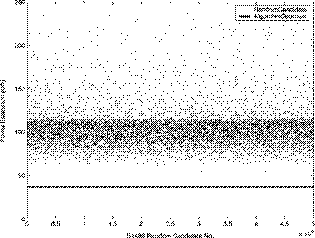


图10算法验证

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 有限状态机 | A、 O.盖茨 | P、 O.（兆瓦） | n | A、 D.闸门 | P、 D.（兆瓦） | A（%） | P（%） | cpu |
| 1488年 | 925 | 160 | 7 | 1090 | 37 | 18 | 77 | 2.7 |
| s820型 | 444 | 75 | 3 | 630 | 41 | 42 | 45 | 0.9 |
| 1494年 | 900 | 141 | 7 | 1092 | 38 | 21 | 73 | 3.3 |
| s832型 | 467 | 80 | 2 | 534 | 36 | 14 | 55 | 0.9 |
| 键盘 | 271 | 72 | 5 | 436 | 34 | 61 | 53 | 0.9 |
| scf公司 | 786 | 80 | 3 | 1067 | 54 | 36 | 33 | 12.7 |

                 表2：标准基准的结果[8]

5.3.4时钟网功率估算：时钟网的功耗按

Pclock¼jFFj Cclkin f VDD2 kbuffer kwire，其中jFFj是时钟触发器的平均数量，Cclkin是时钟输入的电容，VDD是电源电压，f是时钟频率，kbuffer是时钟缓冲器电容系数，kwire是线电容系数。

5.3.5分区架空电路的功率估算：

Poverhead¼PGatedComþPGatedFF，其中PGatedCom包括用于激活和停用组合逻辑的与门的功率，以及用于合并输出的或门；PGatedFF是用于激活和停用本地状态位的功率，基本上是与非门。

基本门（如D触发器和门）的功率来源于标准单元库。

整个分区有限状态机的功耗仅为上述各项的总和

P孔¼PcombþPGSMþPLSMþPclockþPoverhead

6结果

通过比较逻辑综合前后的估计功率，验证了功率估计函数的准确性。我们用psys（synoys）的功率估计作为参考。在图9中，对于基准s1488，可以比较来自估计函数Pwhole、PComb、PGSM和PLSM（标记为估计）的结果与来自power编译器（标记为Actual）的结果。采用0.18毫米技术，VDD为1.8伏，时钟频率为20兆赫。主输入概率和切换活动都设置为0.5。从三种不同的FSM基准（s820、keyb和s1488）的划分树的每一级中选择一系列候选对象进行验证。可以看出，估计函数与门级估计的结果吻合得很好。相关系数衡量两组数据之间的匹配程度，用于验证成本函数。使用相关系数的原因是我们希望找到一个实际功率最小的候选者（也是估计功率最小的候选者）。因此，这两个功率值的绝对差异并不重要。整个分区设计Pwhole的估计功率与实际功率之间的系数s820为0.77，s1488为0.98，键盘为0.88。

候选生成算法的关键是找到功耗最低的候选。为了验证这一点，我们随机生成了s1488的50000个分区，并将它们与工具选择的分区进行了比较。从图10可以看出，随机生成的分区中没有一个比由工具选择的分区更好。

为了说明我们工具的整体性能，表2显示了原始单片FSM和多路分区FSM的比较。标记为“A.O.”和“P.O.”的列分别表示原始单片FSM的面积和功率；标有“n”的列表示分区后的子FSM的数量；“A.D.”和“P.D.”分别表示分解后的FSM的面积和功率；以下两列表示面积增加的百分比和分别对分解后的fsm进行功率降低。

表2中的CPU时间用于在运行Windows2000的奔腾4，1.6GHz处理器上执行的状态聚类和候选生成算法。最大基准（scf121状态）的总时间为5min，这表明最耗时的部分是FSM合成到RT级和功率估计。这支持了我们的想法，即有一个候选人选择算法，以限制候选人的数量，尽早。

7          讨论和结论

本文提出了一种新的用于分段有限状态机综合的多向分割算法。尽管我们已经将其应用于混合的同步/异步架构，但它也可以用于完全同步的实现。我们还提出了RT级功率估计函数，这些函数对于选择功耗最低的候选对象具有足够的精度。所提出的算法具有较低的复杂度，这对于该工具的实际使用具有重要意义。该工具，如图5所示，已经完全用C语言实现。它符合标准的基于单元的设计流程，并与Synopsys工具集完全兼容。对于本文考虑的基准，我们的工具显著降低了功耗，平均为56%。

8          工具书类

1        Benini，L.和de Micheli，G.：“动态电源管理：设计技术和CAD工具”（Kluwer学术出版社，1998）

2        Benini，L.，Siegel，P.和de Micheli，G.：“通过为时序电路合成门控时钟来节省电力”，IEEE Des。《计算机测试》，1994年11月，第32-41页

3        Benini，L.，Vermeulen，F.和de Micheli，G.：“低功耗的有限状态机划分”。程序。IEEE国际研讨会。《电路与系统》，1998年，第2卷，第5-8页

4        Oelmann，B.和O&apos;Nils，M.：“低功耗门控时钟有限状态机的异步控制”。IEEE国际会议，电子，电路和系统，1999年，第915-918页

5        Oelmann，B.，Tammema–e，K.，Kruus，M.和O&apos;Nils，M.：“低功耗混合同步/异步实现的自动FSM合成”，VLSI设计杂志，（低功耗设计特刊），2001，12，（2），第167–186页

6        Chapiro，D.M.：“全球异步本地同步系统”。1984年，斯坦福大学博士论文

7        Tsui，C.Y.，Pedram，M.和Despain，A.M.：“面向两个和多级逻辑实现的低功耗状态分配”，IEEE Trans。计算机。辅助设计，1998，17，（12），第1281-1291页

8        Cao，C.和Oelmann，B.：“用于低功耗FSM设计的混合同步/异步状态存储器”。程序。欧洲微观研讨会。数字系统设计，2004年，第363-370页

9        Unger，S.H.：“异步顺序开关电路”

（威利国际科学出版社，1969）

10     Yang，S.：“优化基准的逻辑综合——用户指南，3.0版”。MCNC，技术报告，1991年

11     Nemani，M.和Najm，F.：“迈向高水平功率估计能力”，IEEE Trans。计算机。辅助设计。内部电路系统。，

1996年，15，（6），第588-598页