FPGA双端口存储器映射优化算法

徐 宇[①②] 林 郁[③] 杨海钢*[①②]

[①](中国科学院电子学研究所 北京 100190)

[②](中国科学院大学 北京 100190)

[③](赛灵思电子科技北京分公司 北京 100101)

摘 要：FPGA存储器映射算法负责将用户的逻辑存储需求映射到芯片中的分布式存储资源上实现。前人对双端口存储器的映射算法研究相对较少，成熟的商业EDA工具的映射结果仍有不少改进空间。该文分别针对面积、延时、功耗这3个常用指标，提出一种双端口存储器映射的优化算法，并给出了具体配置方案。实验表明，在面向简单存储需求时，与商用工具Vivado的映射结果一致；在面向复杂存储需求时，面积优化和功耗优化的映射结果对比商用工具改善了至少50%。

关键词：FPGA；双端口存储器映射；延时优化；面积优化；功耗优化

中图分类号：TN43 **文献标识码**：A **文章编号**：1009-5896(2020)10-2549-08

DOI: 10.11999/JEIT190077

Optimization Algorithm of Dual-port Memory Mapping on FPGA

XU Yu[①②] LIN Yu[③] YANG Haigang[①②]

[①](*Institute of Electrics, Chinese Academy of Sciences, Beijing* 100190, *China*)

[②](*University of Chinese Academy of Sciences, Beijing* 100049, *China*)

[③](*Xilinx Incorporated, Beijing* 100101, *China*)

Abstract: FPGA memory mapping algorithm utilizes distributed storage resources on chip and cooperates with some auxiliary circuits to realize the different needs of users in designing logical storage functions. Previous studies on dual-port memory mapping algorithm are relatively few. There is still much space for improvement in the mapping results by mature commercial EDA tools. An optimization algorithm of dual-port memory mapping is proposed for area, delay and power consumption, and a specific configuration scheme is given. Experiments show that when facing simple storage requirements, the mapping results are consistent with those of commercial tools; when facing complex storage requirements, the mapping results of area optimization and power optimization are improved by at least 50% compared with commercial tools Vivado.

Key words: FPGA; Dual-port memory mapping; Delay optimization; Area optimization; Power optimization

1 引言

FPGA(Field Programmable Gate Array)的最大特点是电路可编程，它设计周期短，开发成本低，在灵活性、性能和功耗方面表现的较为均衡。丰富的异质IP(Intellectual Property)硬核(如乘/累加器、存储器等)的嵌入，极大拓宽了其应用场景。随着人工智能时代的到来，FPGA在并行计算、算法加速、异构计算、功能验证等方面扮演着重要角色。

为了满足用户设计对片上存储的需求，嵌入式存储器很早就作为一种专用IP被引入到FPGA芯片中[1,2]。固定大小的存储器块以阵列的形式排布在FPGA芯片中，可以通过编程对其工作模式进行配置。由于存储器块容量有限，当用户电路需要使用较大的片上逻辑存储空间时，单个存储器块将无法满足存储需求。此时，需要将多个物理存储器块组合在一起，并额外添加一些选通和译码电路，构成较大容量的逻辑存储器，才能够实现电路功能。在FPGA的EDA(Electronics Design Automation)开发流程中，这一过程称为存储器映射(memory mapping)。

早期，存储器只支持单端口工作模式。随着需求日益复杂，存储器逐渐开始支持双端口。双端口

收稿日期：2019-01-28；改回日期：2020-01-20；网络出版：2020-07-20
*通信作者： 杨海钢 yanghg@mail.ie.ac.cn
基金项目：国家自然科学基金(61474120, 61404140, 61704173)
Foundation Items: The National Natural Science Foundation of China (61474120, 61404140, 61704173)

存储器具有两组相互独立的读写控制线路，可以进行并行的独立读写操作。双端口存储器主要有两种工作模式：简单双端口模式和真双端口模式。存储器工作在简单双端口模式时，一个端口专门负责读取数据，另一个端口专门负责写入数据；在真双端口工作模式下，两个端口都可以支持读写操作。

存储器映射算法的研究已经取得了很多成果。文献[3]从减少物理存储单元使用个数的角度，文献[4,5]从降低功耗角度分别进行了单端口映射研究。文献[6]则研究了时序约束下，面积/功耗优化的单端口映射算法。文献[7,8]探索了将一些逻辑功能电路映射到双端口存储器实现的方案。近年来，基于FPGA的深度神经网络硬件加速器研究非常丰富，它既具有可重构性和灵活性，又实现了高性能和低功耗[9]，是一种很有前途的嵌入式解决方案[10]。针对不同结构的卷积神经网络(CNN)模型，文献[11–13]分别提出了端到端的RTL级FPGA电路编译优化框架。然而，片上存储资源有限，各种神经网络模型对其需求较大，合理利用这些存储资源变得非常重要。在双端口映射，特别是存储器在真双端口工作模式下的映射方面，研究相对较少。

Xilinx, Altera(Intel)等公司是商用FPGA芯片巨头，它们提供的EDA软件对存储器映射算法的优化目标主要有3个：面积、功耗和延时。存储器映射结果的优劣常会对电路的芯片资源使用情况、功耗和工作频率造成较大影响。我们发现，商用工具的映射往往只优化了存储器在某个工作状态下的指标，而忽视了存储器在其他工作状态下的指标优化，从全局来看，并未做到很好的平衡，本文对此进行了一些研究。

本文结构如下：第2节对FPGA存储器映射做了简要介绍，第3节提出了AlgoPower, AlgoDelay和AlgoArea面向3种不同优化目标的双端口存储器映射算法，第4节是实验设计，与商用工具的结果对比及分析。最后对全文进行了总结。

2 FPGA存储器映射

本文以Xilinx公司的Virtex-4系列FPGA芯片为例，介绍FPGA存储器映射[14]。Virtex-4芯片中的嵌入式存储器块容量大小为18 kbit，有两个端口A和B，它们分别有独立的读写数据端(写：dina, dinb；读：douta, doutb)。每个存储器块的地址深度和数据位宽可以独立进行配置，具体配置方式如表1所示[15]。

当存储器工作在真双端口模式时，数据位宽需要满足以下约束条件：

表 1 Virtex-4存储器块配置方式

i	索引						
	1	2	3	4	5	6	7
地址深度 $d[i]$	256	512	1k	2k	4k	8k	16k
地址位宽 $b[i]$	8	9	10	11	12	13	14
数据位宽 $w[i]$	72	36	18	9	4	2	1

(1) 端口A与B的位宽可不同，二者之间需要满足的比例关系为1, 2, 4, 8, 16或32。

(2) 对于每个端口，读写数据位宽可不同，二者之间需要满足的比例关系为1, 2, 4, 8, 16或32。

(3) 任意两个数据端(dina, douta, dinb, doutb)位宽之间的最大比例为32。

在设计映射算法时，需要兼顾4个视图下的优化情况，合理安排地址线、数据线的连接，考虑译码电路如何片选出需要的存储器块，选通电路的设计和控制信号的选择，保证功能正确可实现的前提下，完成目标优化。下面介绍存储器映射算法的3个优化目标。

(1) 面积(area)：FPGA片上存储器块数目有限，存储资源十分宝贵。面向面积的存储器映射优化，需要有效提高资源的利用率，让有限资源的芯片能够应用于更大规模的电路设计。

(2) 功耗(power)：低功耗在电路设计中越来越受到关注，很多实际应用场景都对低功耗设计提出了需求。存储器在读写操作时被触发，产生动态功耗。动态功耗与访存某个逻辑地址的数据时被触发的存储器块数目密切相关。如果减少每次操作触发的存储器块数目，可以很好地改善存储器的动态功耗。

(3) 延时(delay)：若用户设计使用了大容量的逻辑存储，综合后可能会在存储器数据输出端产生若干级选通电路，路径延时增加，存储器的访问路径甚至可能成为整个设计在芯片上最终实现的关键路径。减少数据读出需要经过选通电路的多路选择器(MUltipleXer, MUX)级数，可以改善相关路径的延时。

3 映射算法

由于简单双端口模式是真双端口模式的特殊情形，本文仅针对真双端口工作模式下的存储器映射方案进行讨论。我们用记号V(op,D,W)来表示存储器的某逻辑视图(op表示工作方式，分rd(读)和wr(写)两种，D表示地址深度，W表示数据位宽)。

3.1 功耗最低映射算法(AlgoPower)

3.1.1 基本思想

考虑视图V_1～V_4中数据位宽最小的$W_{\min}$和数据位宽最大的$W_{\max}$，其对应视图分别为$V_{\min}$和$V_{\max}$。

采用贪婪策略，根据存储器块支持的配置方式去配置$W_{\min}$，再确定$D_{\min}$的配置，从而完成视图$V_{\min}$下各存储器块的配置选择。根据$V_{\max}$，设计数据的写入方法，从而确定地址线、数据线的连接，最后确定译码电路(写数据视图)和选通电路(读数据视图)。根据位宽的倍比关系，确定其他视图的配置。图1是视图V_k下存储器映射方案图示。V_k被分为2个区域：Lf和Rt，区域是由配置方式为d[id](深度)×w[id](位宽)的存储单元以R(行)×C(列)方式排布形成的阵列。

3.1.2 算法步骤

步骤 1　确定$V_{\min}$, $V_{\max}$, $q_{\max} \leftarrow W_{\max}/W_{\min}$；

步骤 2　$C_{\min} \leftarrow \lfloor W_{\min}/w[\text{first}] \rfloor$, $R_{\min} \leftarrow \lceil D_{\min}/d[\text{first}] \rceil$, $r \leftarrow W_{\min} - C_{\min} \times w[\text{first}]$, $\text{id}_{\min} \leftarrow 0$, if$(r>0)$ $\{\text{id}_{\min} \leftarrow \underset{i}{\text{argmin}}(w[i])$ such that $w[i] > r\}$；

步骤 3　配置区域1部分，$\text{Lf}_{\min}(C_{\min} \times R_{\min})$的block配置为: $d[\text{first}] \times w[\text{first}]$;

步骤 4　$C_{\min\_r} \leftarrow 1, R_{\min\_r} \leftarrow \lceil D_{\min}/d[\text{id}_{\min}] \rceil$，配置区域2部分，$\text{Rt}_{\min}(R_{\min_r} \times C_{\min_r})$的block配置为：$d[\text{id}_{\min}] \times w[\text{id}_{\min}]$;

步骤 5　$s \leftarrow w[\text{first}]/w[\text{id}_{\min}]$，对于视图$V_k$，$q_k \leftarrow W_k/W_{\min}$，计算各区域的行列数及配置方式。$C_k \leftarrow C_{\min} \times q_k, R_k \leftarrow R_{\min}/q_k, C_{k\_r} \leftarrow C_{\min\_r}, R_{k\_r} \leftarrow R_{\min\_r}, \text{id}_k \leftarrow \text{id}_{\min}$, $q_k > s$时，$C_{k\_r} \leftarrow q_k/s \times C_{k\_r}$, $R_{k\_r} \leftarrow s/q_k \times R_{k\_r}, \text{id}_k \leftarrow \text{first}$，$q_k \le s$时，$\text{offset}_k \leftarrow \log_2 q_k, \text{id}_k \leftarrow \text{id}_k + \text{offset}_k$；

步骤 6　$\text{Lf}_k(C_k \times R_k)$的block配置：$d[\text{first}] \times w[\text{first}]$，$\text{Rt}_k(C_{k_r} \times R_{k_r})$的block配置：$d[\text{id}_k] \times w[\text{id}_k]$；

步骤 7　假设V_k地址线：$a_m a_{m-1} \cdots a_1 a_0$(位宽$m$+1)，数据线：$d_n d_{n-1} \cdots d_1 d_0$(位宽$n$+1)。

(1) 区域$\text{Lf}_k(i,j)(d[\text{first}] \times w[\text{first}])$：$\text{pLf}_k \leftarrow \log_2(q_{\max}/q_k), \text{stLf}_j \leftarrow w[\text{first}] \times (j-1)$。

地址：$a_{\text{pLf}_k + b[\text{first}] - 1} \cdots a_{\text{pLf}_k + 1} a_{\text{pLf}_k}$，数据：$d_{\text{stLf}_{j+1}-1} d_{\text{stLf}_{j+1}-2} \cdots d_{\text{stLf}_j+1} d_{\text{stLf}_j}$，

译码/选通：$a_m a_{m-1} \cdots a_{\text{pL}_k + b[\text{first}]} a_{\text{pLf}_k - 1} \cdots a_0 \leftrightarrow (i-1)$;

(2) 区域$\text{Rt}_k(i,j)(d[\text{id}_k] \times w[\text{id}_k])$：$\text{pRtFold}_k \leftarrow \text{id}_{\max} - \text{id}_k$，$\text{pRtBlk}_k \leftarrow \text{pLf}_k - \text{pRtFold}_k$，$\text{stRt}_j \leftarrow w[\text{id}_k] \times (j-1) + w[\text{first}] \times C_k$。

地址：$a_{b[\text{id}_k] + \text{pRtBlk}_k - 1} \cdots a_{\text{pLf}_k + 1} a_{\text{pLf}_k} a_{\text{pRtFold}_k - 1} \cdots a_1 a_0$，数据：$d_{\text{stRt}_{j+1}-1} d_{\text{stRt}_{j+1}-2} \cdots d_{\text{stRt}_j+1} d_{\text{stRt}_j}$，

译码/选通：$a_m a_{m-1} \cdots a_{\text{pRtBlk}_k + b[\text{id}_k]} a_{\text{pLf}_k - 1} \cdots a_{\text{pRtFold}_k + 1} a_{\text{pRtFold}_k} \leftrightarrow (i-1)$；

步骤 8　$\text{Lf}_k(i,j)$与$\text{Lf}_{\min}(i_{\min}, j_{\min})$是同一个RAM的不同视图，当且仅当：$((i_{\min}-1)/q_k + 1, j_{\min}) \leftrightarrow (i, (j-1)/q_k + 1)$; $\text{Rt}_k(i,j)$与$\text{Rt}_{\min}(i_{\min}, j_{\min})$是同一个RAM的不同视图，当且仅当(“≪”为向左移位操作符)：$\left(\dfrac{i_{\min}-1}{1 \ll (\text{pRtBlk}_{\min} - \text{pRtBlk}_k)} + 1, j_{\min}\right) \leftrightarrow \left(i, \dfrac{j-1}{1 \ll (\text{pRtBlk}_{\min} - \text{pRtBlk}_k)} + 1\right)$。

3.1.3 举例

4个视图分别为：V_1 (wr, 1k, 88), V_2 (rd, 2k, 44), V_3 (wr, 4k, 22), V_4 (rd, 4k, 22)。为方便说明，假设存储器块只支持表1中序号3～7的配置方式。$V_{\min}$为V_4, $W_{\min}$为22; $V_{\max}$为V_1, $W_{\max}$为88。采取贪婪策略，得到22 ($W_{\min}$) = 18×1 + 4。假设视图地址信号为a，数据信号为d，具体方案见表2。

3.2 延时最短映射算法(AlgoDelay)

3.2.1 基本思想

写视图不需要选通电路，故优化延时只需考虑读视图。假设视图V_1～V_4中数据位宽最大为$W_{\max}$；读视图中，数据位宽最大的W_{maxR}，对应视图为V_{maxR}。先确定V_{maxR}的配置方案。根据D_{maxR}和存储器块支持的最大深度配置，计算选通电路需要的最少级数L。先采用贪婪算法，覆盖D_{maxR}。为优化功耗，在不增加选通电路级数的情况下，从低地址开始，尽量依次将采用最大深度配置方式的存储器块调整为次大深度，并根据$W_{\max}$与W_{maxR}的比例关系选择合适的方式配置$\text{Down}_{\text{maxR}}$部分，从而完成$V_{\text{maxR}}$存储器块4个部分$\text{Up}_{\text{maxR}}$, Mid_{maxR}, $\text{Down}_{\text{maxR}}$和$\text{Rt}_{\text{maxR}}$的配置。再根据视图$V_k$相对$V_{\text{maxR}}$的位宽倍比关系，确定$V_k$的配置。然后，设计数据的写入方法，从而确定地址线、数据线的连接，最后确定译码电路(写视图)和选通电路(读视图)。图2是视图V_k下存储器映射方案图示。V_k被分为4个区域：Up, Mid, Down和Rt，区域是由配置方式为d[id](深度)×w[id](位宽)的存储单元以R(行)×C(列)方式排布形成的阵列。

3.2.2 算法步骤

步骤 1　确定V_{maxR}, $W_{\max}$；

步骤 2　对于读视图V_{maxR}，计算区域的行列数目。$N_0 \leftarrow \lceil D_{\text{maxR}}/d[\text{last}] \rceil, L \leftarrow \lceil \log_2 N_0 \rceil, N \leftarrow 2, C_{\text{maxR\_rt}} \leftarrow (W_{\text{maxR}} \& 1)$(“&”表示位与)，

区域1	区域2
$\text{Lf}_k(R_{k\_\text{lf}} \times C_{k\_\text{lf}})$: $d[\text{id}_{k\_\text{lf}}] \times w[\text{id}_{k\_\text{lf}}]$	$\text{Rt}_k(R_{k\_\text{rt}} \times C_{k\_\text{rt}})$: $d[\text{id}_{k\_\text{rt}}] \times w[\text{id}_{k\_\text{rt}}]$

图 1 AlgoPower映射算法图示(视图V_k)

表 2 AlgoPower映射策略方案

序号	视图	配置	地址端口连接	数据端口连接	译码/选通
#1	$V1$	1k×18	$a9\ a8\ \cdots\ a1\ a0$	$d33\ d32\ \cdots\ d17\ d16$	–
	$V2$	1k×18	$a10\ a9\ \cdots\ a2\ a1$	$d25\ d24\ \cdots\ d9\ d8$	$(a0)=0$
	$V3\ V4$	1k×18	$a11\ a10\ \cdots\ a3\ a2$	$d21\ d20\ \cdots\ d5\ d4$	$(a1\ a0)=00$
#2	$V1$	1k×18	$a9\ a8\ \cdots\ a1\ a0$	$d51\ d50\ \cdots\ d35\ d34$	–
	$V2$	1k×18	$a10\ a9\ \cdots\ a2\ a1$	$d43\ d42\ \cdots\ d27\ d26$	$(a0)=0$
	$V3\ V4$	1k×18	$a11\ a10\ \cdots\ a3\ a2$	$d21\ d20\ \cdots\ d5\ d4$	$(a1\ a0)=01$
#3	$V1$	1k×18	$a9\ a8\ \cdots\ a1\ a0$	$d69\ d68\ \cdots\ d53\ d52$	–
	$V2$	1k×18	$a10\ a9\ \cdots\ a2\ a1$	$d25\ d24\ \cdots\ d9\ d8$	$(a0)=1$
	$V3\ V4$	1k×18	$a11\ a10\ \cdots\ a3\ a2$	$d21\ d20\ \cdots\ d5\ d4$	$(a1\ a0)=10$
#4	$V1$	1k×18	$a9\ a8\ \cdots\ a1\ a0$	$d87\ d86\ \cdots\ d71\ d70$	–
	$V2$	1k×18	$a10\ a9\ \cdots\ a2\ a1$	$d43\ d42\ \cdots\ d27\ d26$	$(a0)=1$
	$V3\ V4$	1k×18	$a11\ a10\ \cdots\ a3\ a2$	$d21\ d20\ \cdots\ d5\ d4$	$(a1\ a0)=11$
#5	$V1$	1k×18	$a9\ a8\ \cdots\ a1\ a0$	$d15\ d14\ \cdots\ d1\ d0$	–
	$V2$	2k×9	$a10\ a9\ \cdots\ a1\ a0$	$d7\ d6\ \cdots\ d1\ d0$	–
	$V3\ V4$	4k×4	$a11\ a10\ \cdots\ a1\ a0$	$d3\ d2\ d1\ d0$	–

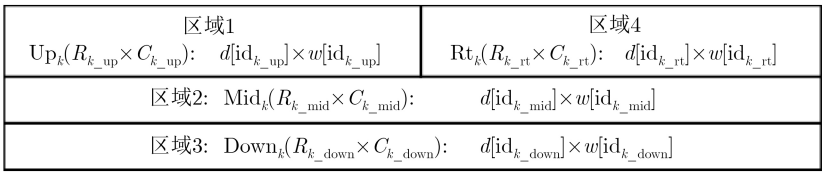


图 2 AlgoDelay映射算法图示(视图V_k)

$C_{\mathrm{maxR\_up}}\leftarrow dW_{\mathrm{maxR}}/2t, C_{\mathrm{maxR\_mid}}\leftarrow W_{\mathrm{maxR}}, C_{\mathrm{maxR\_down}}\leftarrow 0$, $R_{\mathrm{maxR\_mid}}\leftarrow\lfloor D_{\mathrm{maxR}}/d[\mathrm{last}]\rfloor$,

$R_{\mathrm{maxR\_down}}\leftarrow 0, R_{\mathrm{maxR\_up}}\leftarrow 0$, if $(W_{\mathrm{maxR}}>1)\{R_{\mathrm{maxR\_up}}\leftarrow N-R_{\mathrm{maxR\_mid}}\}$,

$D_{\mathrm{maxR\_r}}\leftarrow D_{\mathrm{maxR}}-d[\mathrm{last}]\times R_{\mathrm{maxR\_mid}}$, $\mathrm{id}_{\mathrm{maxR\_up}}\leftarrow \mathrm{last}-1$, $\mathrm{id}_{\mathrm{maxR\_mid}}\leftarrow \mathrm{last}$, $\mathrm{id}_{\mathrm{maxR\_rt}}\leftarrow \mathrm{last}$,

$D_{\mathrm{maxR_r}}>0$时：

$\mathrm{id}_{\mathrm{maxR\_down}}\leftarrow \underset{i}{\mathrm{argmin}}(d[i])$ such that $d[i]>D_{\mathrm{maxR\_r}}$,

$\mathrm{id}_{\mathrm{maxR\_down}}\leftarrow \max(\mathrm{id}_{\mathrm{maxR\_down}}, \mathrm{first}+\log_2(W_{\mathrm{max}}/W_{\mathrm{maxR}}))$,

if$(W_{\mathrm{maxR}}>1)\{R_{\mathrm{maxR_up}}\leftarrow R_{\mathrm{maxR_up}}-1\}R_{\mathrm{maxR_down}}\leftarrow 1, C_{\mathrm{maxR_down}}\leftarrow\lceil W_{\mathrm{maxR}}/w[\mathrm{id}_{\mathrm{maxR_down}}]\rceil$,

$R_{\mathrm{maxR\_rt}}\leftarrow R_{\mathrm{maxR\_up}}$, $R_{\mathrm{maxR\_mid}}\leftarrow R_{\mathrm{maxR\_mid}}-R_{\mathrm{maxR\_up}}$, $R_{\mathrm{maxR\_up}}\leftarrow R_{\mathrm{maxR\_up}}\times 2$;

步骤 3 假设V_k地址线：$a_m a_{m-1}\cdots a_1 a_0$(位宽$m$+1)，数据线：$d_n d_{n-1}\cdots d_1 d_0$(位宽$n$+1)。

(1) 对于视图$V_k(W_k<W_{\mathrm{maxR}})$，$q_k\leftarrow W_{\mathrm{maxR}}/W_k$, $C_{k\_\mathrm{rt}}\leftarrow 0$, $R_{\mathrm{k\_rt}}\leftarrow 0$,对于视图$V_k$的某区域Rg第$i$行第$j$列的存储块$\mathrm{Rg}_k(i,j)$，配置$d[\mathrm{id}_{k_\mathrm{rg}}]\times w[\mathrm{id}_{k_\mathrm{rg}}]$：

$g\leftarrow\gcd(q_k, C_{\mathrm{maxR\_rg}})$, $p_1\leftarrow\log_2 g$, $p_2\leftarrow\log_2 q_k-p_1$,

$R_{k\_\mathrm{rg}}\leftarrow R_{\mathrm{maxR\_rg}}\times g$, $C_{\mathrm{k\_rg}}\leftarrow C_{\mathrm{maxR\_rg}}/g$, $\mathrm{id}_{k\_\mathrm{rg}}\leftarrow\mathrm{id}_{\mathrm{maxR\_rg}}+p_2$, $\mathrm{stRg}_j\leftarrow w[\mathrm{id}_{k\_\mathrm{rg}}]\times(j-1)$。

地址：$a_{b[\mathrm{id}_{\mathrm{k\_rg}}]+p_1-1}\cdots a_{p_1+p_2}a_{p_2-1}\cdots a_0$，数据：$d_{\mathrm{stRg}_j+w[\mathrm{id}_{\mathrm{k\_rg}}]-1}\cdots d_{\mathrm{stRg}_j}$。

译码/选通：idx1$\leftarrow 0$, idx2$\leftarrow 0$，对于V_k每一行：

如果row_id$\le R_{\mathrm{k_up}}$且$p_2==0$且$a_{b[\mathrm{id}_{\mathrm{k_mid}}]+p_1-1}$&1为1，则$\overline{a_m a_{m-1}\cdots a_{b[\mathrm{id}_{\mathrm{k_mid}}]+p_1}a_{p_1+p_2-1}\cdots a_{p_2}}\leftrightarrow$ idx1, idx1$\leftarrow$idx1$+1$ ；否则，$a_m a_{m-1}\cdots a_{b[\mathrm{id}_{\mathrm{k\_mid}}]+p_1}a_{p_1+p_2-1}\cdots a_{p_2}\leftrightarrow$ idx2 , idx2$\leftarrow$idx2$+1$ 。

(2) 对于视图$V_k(W_k\ge W_{\mathrm{maxR}})$, $q_k\leftarrow W_k/W_{\mathrm{maxR}}$, $\mathrm{offset}_k\leftarrow\log_2 q_k$，对于视图$V_k$的某区域Rg第$i$行第$j$列的存储块$\mathrm{Rg}_k(i,j)$，配置$d[\mathrm{id}_{k_\mathrm{rg}}]\times w[\mathrm{id}_{k_\mathrm{rg}}]$：

$R_{k\_\mathrm{rg}}\leftarrow R_{\mathrm{maxR\_rg}}$, $C_{k\_\mathrm{rg}}\leftarrow C_{\mathrm{maxR\_rg}}$, $\mathrm{id}_{k\_\mathrm{rg}}\leftarrow\mathrm{id}_{\mathrm{maxR\_rg}}-\mathrm{offset}_k$, $\mathrm{stRg}_{j,l}\leftarrow w[\mathrm{id}_{\mathrm{maxR\_rg}}]\times(l\times C_{\mathrm{maxR\_rg}}+j-1)(l=0,1,\cdots,q_k-1)$, $\mathrm{endRg}_{j,l}\leftarrow w[\mathrm{id}_{\mathrm{maxR\_rg}}]\times(l\times C_{\mathrm{maxR\_rg}}+j)-1(l=0,1,\cdots,q_k-1)$。

地址：$a_{b[\mathrm{id}_{k\_\mathrm{rg}}]-1}a_{b[\mathrm{id}_{k\_\mathrm{rg}}]-2}\cdots a_1 a_0$，数据：$d_{\mathrm{endRg}_{j,q_k-1}}\cdots d_{\mathrm{stRg}_{j,q_k-1}}\cdots d_{\mathrm{endRg}_{j,1}}\cdots d_{\mathrm{stRg}_{j,1}}d_{\mathrm{endRg}_{j,0}}\cdots d_{\mathrm{stRg}_{j,0}}$。

译码/选通：idx1$\leftarrow 0$, idx2$\leftarrow 0$，对于V_k每一行：

如果 $\text{row\_id} \le R_{k\_\text{up}}$ 且 $a_{b[\text{id}_{k\_\text{mid}}]-1}\&1$ 为 1，则 $\overline{a_m a_{m-1}\cdots a_{b[\text{id}_{k\_\text{mid}}]}} \leftrightarrow \text{idx1}$，$\text{idx1} \leftarrow \text{idx1}+1$；否则 $a_m a_{m-1}\cdots a_{b[\text{id}_{k\_\text{mid}}]} \leftrightarrow \text{idx2}$，$\text{idx2} \leftarrow \text{idx2}+1$。

步骤 4　(1)对于视图V_k的某区域Rg(Up, Mid或Down), $\text{Rg}_k(i_k, j_k)$与$\text{Rg}_{\text{maxR}}(i_{\text{maxR}}, j_{\text{maxR}})$是同一个RAM的不同视图，当且仅当：

$W_k \ge W_{\text{maxR}}$时，$(i_k, j_k) \leftrightarrow (i_{\text{maxR}}, j_{\text{maxR}})$；

$W_k < W_{\text{maxR}}$时，$(C_{k_\text{rg}} \times (i_k - 1) + j_k) \leftrightarrow (C_{\text{maxR_rg}} \times (i_{\text{maxR}} - 1) + j_{\text{maxR}})$。

(2) $\text{Rt}_k(i_k, j_k)$与$\text{Rt}_{\text{maxR}}(i_{\text{maxR}}, j_{\text{maxR}})$是同一个RAM的不同视图，当且仅当：$(i_k, j_k) \leftrightarrow (i_{\text{maxR}}, j_{\text{maxR}})$。

3.2.3 举例

4个视图分别为V_1(rd, 36k, 2), V_2(wr, 18k, 4), V_3(wr, 9k, 8), V_4(rd, 9k, 8)。读视图V_{maxR}为：V_1 (rd, 36k, 4)。对采取贪婪策略，得到读视图最大深度36k (D_{maxR}) = 16k×2 + 4k。为满足地址需求，至少需要3个存储器块，故数据端最少需要2级选通。在满足该约束下，覆盖地址深度最多可以使用4个存储器块。因此，可以将1行配置为16k×1的存储器块优化为2行配置为8k×2的存储器块。

假设视图地址信号为a，数据信号为d，具体设计见表3。图3给出了V_{maxR}纵向各存储器块的起始地址和选通电路设计。

功耗优化前，V_{maxR}每次访存操作触发的存储器块加权平均数目为：$(16\text{k} \times 2 \times 2 + 4\text{k} \times 1 \times 1)/36\text{k} \approx 1.89$。

功耗优化后，在不影响延时的情况下，V_{maxR}每次访存操作触发的存储器块加权平均数目为：$(16\text{k} \times 2 \times 1 + 8\text{k} \times 1 \times 2 + 4\text{k} \times 1 \times 1)/36\text{k} \approx 1.44$　。

3.3 面积最小映射算法(AlgoArea)

3.3.1 基本思想

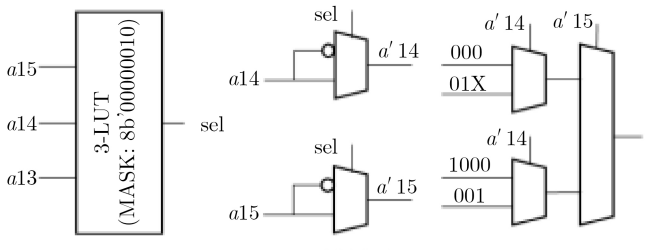
考虑视图$V_1 \sim V_4$中数据位宽最小的$W_{\min}$和数据位宽最大的$W_{\max}$，其对应视图分别为$V_{\min}$和$V_{\max}$。根据$W_{\min}$和$W_{\max}$的比例关系，适当缩小$V_{\min}$中存储器块的可配置位宽上限。采用保守策略，先得到一个对$V_{\min}$的初始化配置，不断迭代配置方案，直到面积无法进一步改善为止，从而完成视图$V_{\min}$下各

表 3　AlgoDelay映射策略方案

序号	视图	配置	地址端口连接	数据端口连接	译码/选通
#1	$V1$	8k×2	$a12\ a11 \cdots a1\ a0$	$d1\ d0$	$(a'\ 15\ a'\ 14) = 00$
	$V2$	4k×4	$a11\ a10 \cdots a1\ a0$	$d3\ d2\ d1\ d0$	$(a'\ 14\ a'\ 13) = 00$
	$V3\ V4$	2k×8	$a10\ a9 \cdots a1\ a0$	$d7\ d6\ d5\ d4\ d3\ d2\ d1\ d0$	$(a'\ 13\ a'\ 12) = 00$
#2	$V1$	8k×2	$a12\ a11 \cdots a1\ a0$	$d1\ d0$	$(a'\ 15\ a'\ 14) = 11$
	$V2$	4k×4	$a11\ a10 \cdots a1\ a0$	$d3\ d2\ d1\ d0$	$(a'\ 14\ a'\ 13) = 11$
	$V3\ V4$	2k×8	$a10\ a9 \cdots a1\ a0$	$d7\ d6\ d5\ d4\ d3\ d2\ d1\ d0$	$(a'\ 13\ a'\ 12) = 11$
#3	$V1$	16k×1	$a12\ a11 \cdots a1\ a0$	$d1$	$(a'\ 15\ a'\ 14) = 01$
	$V2$	8k×2	$a11\ a10 \cdots a1\ a0$	$d3\ d1$	$(a'\ 14\ a'\ 13) = 01$
	$V3\ V4$	4k×4	$a10\ a9 \cdots a1\ a0$	$d7\ d5\ d3\ d1$	$(a'\ 13\ a'\ 12) = 01$
#4	$V1$	16k×1	$a12\ a11 \cdots a1\ a0$	$d0$	$(a'\ 15\ a'\ 14) = 01$
	$V2$	8k×2	$a11\ a10 \cdots a1\ a0$	$d2\ d0$	$(a'\ 14\ a'\ 13) = 01$
	$V3\ V4$	4k×4	$a10\ a9 \cdots a1\ a0$	$d6\ d4\ d2\ d0$	$(a'\ 13\ a'\ 12) = 01$
#5	$V1$	4k×4	$a13\ a12 \cdots a1\ a0$	$d3\ d2\ d1\ d0$	$(a'\ 15\ a'\ 14) = 10$
	$V2$	2k×8	$a12\ a11 \cdots a1\ a0$	$d7\ d6\ d5\ d4\ d3\ d2\ d1\ d0$	$(a'\ 14\ a'\ 13) = 10$
	$V3\ V4$	1k×16	$a11\ a10 \cdots a1\ a0$	$d15\ d14 \cdots d1\ d0$	$(a'\ 13\ a'\ 12) = 10$

配置	$a15$	$a14$	$a13$	$a12$	$a11$-$a0$
8k×2	0	0	0	–	–
8k×2	0	0	1	–	–
16k×1	0	1	–	–	–
4k×4	1	0	0	0	–

(a) 起始地址



(b) 选通电路

图 3 V_{maxR}纵向存储器块设计

存储器块的配置选择。根据位宽的倍比关系，确定其他视图的配置，最后确定地址线、数据线、译码/选通电路。

3.3.2 算法步骤

步骤 1 确定$V_{\min}$和$V_{\max}$，$q_{\max} \leftarrow W_{\max}/W_{\min}$，$\text{offset}_{\max} \leftarrow \log_2 q_{\max}$；

步骤 2 endIdx $\leftarrow$ last，startIdx $\leftarrow$ first+$\text{offset}_{\max}$ $w, \leftarrow W_{\min}$ // 初始化映射配置方法，

for idx from startIdx to endIdx: $e_{\min}[\text{idx}] \leftarrow \lfloor w/w[\text{idx}] \rfloor$，$w \leftarrow w - e_{\min}[\text{idx}] \times w[\text{idx}]$，$\text{mem\_num} \leftarrow \sum_i (e_{\min}[i] \times \lceil D_{\min}/d[i] \rceil)$；

步骤 3 $e_{\text{tmp}} \leftarrow e_{\min}$,$\text{mem\_num}_{\text{tmp}} \leftarrow \text{mem\_num}$ //迭代优化面积，

while $(\text{mem\_num}_{\text{tmp}} \le \text{mem\_num})$:

$\text{mem\_num} \leftarrow \text{mem\_num}_{\text{tmp}}$，$e_{\min} \leftarrow e_{\text{tmp}}$，$\text{idx} \leftarrow \max(i)$ such that $e_{\text{tmp}}[i] \neq 0$，

$$e_{\text{tmp}}[\text{idx}-1] \leftarrow e_{\text{tmp}}[\text{idx}-1] + \left\lceil \frac{e_{\text{tmp}}[\text{idx}]}{2} \right\rceil,\ e_{\text{tmp}}[\text{idx}] \leftarrow 0,\ \text{mem\_num}_{\text{tmp}} \leftarrow \sum_i \left(e_{\text{tmp}}[i] \times \left\lceil \frac{D_{\min}}{d[i]} \right\rceil \right);$$

步骤 4 $q_k \leftarrow W_k/W_{\min}$，$\text{offset}_k \leftarrow \log_2 q_k$// 配置其他视图$V_k$，

for idx from startIdx to endIdx: $e_k[\text{idx}-\text{offset}_k] \leftarrow e_{\min}[\text{idx}]$；

步骤 5 类似AlgoDelay中的方法配置地址、数据、选通/译码电路，较简单，从略。

3.3.3 举例

4个视图分别为V_1(rd, 2k, 25)，V_2(wr, 512, 100)，V_3(rd, 1k, 50)，V_4(wr, 1k, 50)。$V_{\min}$为V_1，$W_{\min}$为25；$V_{\max}$为V_2，$W_{\max}$为100。计算可得endIdx为7，startIdx为3。对$W_{\min}$进行初始化配置，得到25 ($W_{\min}$) = 18×1+4×1+2×1+1×1。经过迭代，最终得到面积最小的$V_{\min}$配置方案：最少使用3个存储器块(2个1k×18, 1个2k×9)。根据其他各视图相对$V_{\min}$的倍比关系，易得如图4的视图V_1～V_4的配置方案。地址、数据、选通/译码电路较简单，从略。

4 实验

实验采用Xilinx公司的商用EDA工具Vivado 2018.2版[15]，选用的芯片为Virtex-4系列。

4.1 功耗优化

本文选择表4中的一系列逻辑存储配置，并列出了Vivado与AlgoPower面向功耗的映射算法结果。

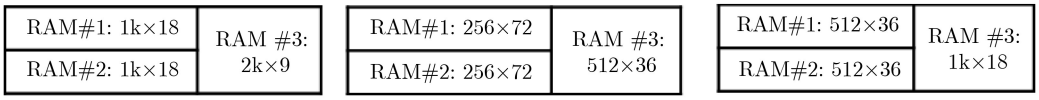
可以看出，各视图数据位宽一致的简单情形下，两种算法加权平均触发存储器块数目一致，当视图的数据位宽不一致时，AlgoPower平均触发的存储器数目明显少于Vivado的结果，优化比例超过至少50%，且最大与最小位宽之间的比例越大，优化效果越明显。本文算法大大改善了存储器平均动态功耗。

4.2 延时优化

Vivado没有提供面向延时最优的映射策略。本文提出的AlgoDelay算法在保证延时最优的同时，对功耗、面积也做了适当优化。本文选择表5中的一系列逻辑存储配置，并列出了AlgoDelay面向延时的映射算法结果。从表5中可以看出，各视图中的选通级数均达到了可能获得的最优结果。

4.3 面积优化

本文选择表6中的一系列逻辑存储配置，并列出了Vivado与AlgoArea面向面积的映射算法结果。由于综合考虑了所有视图，AlgoArea映射后的资源使用数目与Vivado一致或者优于其结果。简单配置情形，即读写位宽一致的情况下，两种算法使



(a) V_1存储器配置方案 (b) V_2存储器配置方案 (c) V_3, V_4存储器配置方案

图 4 AlgoArea映射策略配置方案

表 4 功耗优化实验结果

序号	地址1	读1	写1	地址2	读2	写2	Vivado平均触发数	AlgoPower平均触发数	优化比例(%)
1	10	32	32	10	32	32	1	1.00	0
2	10	32	64	10	32	256	8	1.14	85.7
3	11	16	32	10	32	256	8	1.09	86.4
4	11	16	32	10	32	128	4	1.06	73.5
5	11	16	256	10	32	128	8	1.33	83.4
6	11	32	32	10	64	64	2	1.00	50.0
7	11	32	64	10	64	128	4	1.11	72.2
8	11	16	32	11	16	128	4	1.05	73.8

表 5　延时优化实验结果

序号	*V*1(rd)	*V*2(rd)	*V*3(wr)	*V*4(wr)	AlgoDelay选通级数
1	36k×4	72k×2	18k×8	9k×16	*V*1: 2 *V*2: 3
2	81k×16	324k×4	162k×8	162k×8	*V*1: 3 *V*2: 5
3	18k×4	9k×8	36k×2	18k×4	*V*1: 2 *V*2: 1
4	4k×25	2k×50	1k×100	4k×25	*V*1: 1 *V*2: 1
5	16k×4	32k×2	8k×8	4k×16	*V*1: 1 *V*2: 2

表 6　面积优化实验结果

序号	地址1	读1	写1	地址2	读2	写2	Vivado使用资源	AlgoArea使用资源	优化比例(%)
1	10	32	32	10	32	32	2	2	0
2	10	32	64	10	32	256	8	4	50
3	11	16	32	10	32	256	8	4	50
4	11	16	32	10	32	128	4	2	50
5	11	16	256	10	32	128	4	4	0
6	11	32	32	10	64	64	4	4	0
7	11	32	64	10	64	128	4	4	0
8	11	16	32	11	16	128	4	2	50

用的存储资源相同。复杂配置情形，即各数据端口位宽不一致的情况下，AlgoArea算法得到的存储器资源利用率相较于Vivado至少可以节省50%的存储资源。

5　结束语

本文提出了分别面向功耗、延时、面积3个主要优化目标的FPGA双端口存储器映射算法，并与Xilinx公司的商用EDA工具Vivado进行了实验比较，实验表明：(1)Vivado的双端口存储器映射策略，只在最大位宽的视图下功耗最优。面向功耗的AlgoPower映射算法，在各视图均实现了功耗最优。AlgoPower在复杂存储器配置情况下，加权功耗可以降低50%以上。(2)Vivado没有提供以延时为第一优化目标的双端口存储器映射策略。面向延时的AlgoDelay映射算法，在不浪费资源的情况下，最优化各视图中选通延时。(3)Vivado在简单存储器配置情形下，可以获得最优面积；复杂情形，面积较大。面向面积的AlgoArea映射算法，保证时序性能的前提下，实现了各位宽视图下面积最优化，并在面积优化过程中注意尽量优化功耗。复杂情形下，AlgoArea的映射结果至少可节省50%的存储资源。

FPGA被广泛应用于嵌入式系统神经网络硬件加速，本文对双端口存储器映射算法上的改进可以极大提高片上存储资源利用率，明显降低存储器动态功耗，有助于进一步挖掘FPGA芯片在人工智能领域的潜力。

参 考 文 献

[1] TRIMBERGER S M. Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology[J]. *Proceedings of the IEEE*, 2015, 103(3): 318–331. doi: 10.1109/JPROC.2015.2392104.

[2] KUON I and ROSE J. Measuring the gap between FPGAs and ASICs[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2007, 26(2): 203–215. doi: 10.1109/TCAD.2006.884574.

[3] WILTON S J E. Architectures and algorithms for Field-Programmable Gate Arrays with embedded memory[D]. [Ph. D. dissertation], University of Toronto, 1997.

[4] TESSIER R, BETZ V, NETO D, *et al*. Power-efficient RAM mapping algorithms for FPGA embedded memory blocks[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2007, 26(2): 278–290. doi: 10.1109/TCAD.2006.887924.

[5] HSU T Y and WANG Tingchi. A generalized network flow based algorithm for power-aware FPGA memory mapping[C]. The 45th ACM/IEEE Design Automation Conference, Anaheim, USA, 2008: 30–33.

[6] DU Fangqing, LIN C Y, CUI Xiuhai, *et al*. Timing-constrained minimum area/power FPGA memory mapping[C]. The 23rd International Conference on Field programmable Logic and Applications, Porto, Portugal, 2013: 1–4.

[7] HO W K C and WILTON S J E. Logical-to-physical memory mapping for FPGAs with dual-port embedded arrays[C]. The 9th International Workshop on Field

Programmable Logic and Applications, Glasgow, UK, 1999: 111–123.

[8] CONG J and YAN K. Synthesis for FPGAs with embedded memory blocks[C]. 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays, Monterey, USA, 2000: 75–82.

[9] MA Yufei, CAO Yu, VRUDHULA S, *et al*. An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks[C]. The 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, Belgium, 2017: 1–8.

[10] GUAN Yijin, LIANG Hao, XU Ningyi, *et al*. FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates[C]. The 25th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, USA, 2017: 152–159.

[11] LIANG Shuang, YIN Shouyi, LIU Leibo, *et al*. FP-BNN: Binarized neural network on FPGA[J]. *Neurocomputing*, 2018, 275: 1072–1086. doi: 10.1016/j.neucom.2017.09.046.

[12] GUO Kaiyuan, SUI Lingzhi, QIU Jiantao, *et al*. Angel-eye: A complete design flow for mapping CNN onto embedded FPGA[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018, 37(1): 35–47. doi: 10.1109/TCAD.2017.2705069.

[13] MA Yufei, SUDA N, CAO Yu, *et al*. ALAMO: FPGA acceleration of deep learning algorithms with a modularized RTL compiler[J]. *Integration*, 2018, 62: 14–23. doi: 10.1016/j.vlsi.2017.12.009.

[14] Xilinx. Virtex-4 FPGA user guide[EB/OL]. https://china.xilinx.com/support/documentation/user_guides/ug070.pdf, 2008.

[15] Xilinx. LogiCORE IP product guide block memory generator v8.4[EB/OL]. https://china.xilinx.com/support/documentation/ip_documentation/blk_mem_gen/v8_4/pg058-blk-mem-gen.pdf, 2019.

徐 宇：男，1990年生，博士生，研究方向为FPGA软件设计自动化.

林 郁：男，1982年生，高级工程师，研究方向为FPGA软件设计自动化.

杨海钢：男，1960年生，研究员/教授，博士生导师，研究方向微电子学与集成电路技术.

责任编辑：陈 倩