

AVS 视频解码中帧内预测模块的硬件化设计及 SoPC 验证

作者：刘家良 任怀鲁 指导教师：陈新华

(山东科技大学 信息科学与工程学院 青岛)

摘要： 本文通过研究 AVS 标准中帧内预测的实现算法，对帧内预测模块进行了划分，并根据各个模块的实现方法分别对其进行了硬件化设计。其中，在预测值计算模块设计中，提出了一种关键路径更短、占用资源更少的可重构运算单元，利于流水线设计，可以提高运行频率。并且，在参考样本管理方案中采用了一种环形 Ram 预加载方案，可以有效地提高了预测速度。借助于基于 Nios II 的 SoPC 系统，通过在 Altera 公司的 Cyclone II FPGA 平台上进行验证和测试，证明本设计的帧内预测模块可以正常工作在 100Mhz，解码速度提高了 19.4%。

关键字：帧内预测 AVS 视频编码标准 硬件加速 SoPC

Hardware Design of Intra-Prediction in AVS Video Decoder

Author: Jialiang Liu Huailu Ren Advisor: Xinhua Chen

ShanDong University Of Science and Technology

College Of Information Science And Engineering QingDao

ABSTRACT: This paper completes the division of Intra-prediction module by researching its implement method. What is more, this paper achieves the hardware design of Intra-prediction. In this design, a Processing Element(PE), whose critical path is short and which economizes hardware resource, is brought up. This PE contributes to pipeline design and can improve the running frequency. And, In the samples management module, we adopt a kind of Ring-Ram, which can pre-load samples and improves the prediction's speed. The verification and test in Cyclone II FPGA platform from Altera Cor indicates the intra-prediction module in this design can work well with 100MHz, and the decoding speed is increased by 19.4%.

Keywords: Intra-prediction AVS standards Hardware acceleration SoPC

引言

随着数字视频技术的迅速发展,视频技术已应用到国民经济和社会生活的各个领域。针对数字视频数据量大的特点,国际上先后制定出了先进的视频编解码标准,如 MPEG-2、MPEG-4、H.264 等。我国也于 2006 年制定出了具有我国自主知识产权的第二代音视频编码标准 AVS。AVS 具有专利费低,编码效率高,算法相对简洁,应用范围广等特点,使得 AVS 更具优势^[1]。

然而,由于这些高性能的视频编解码标准的出现,以及目前对高分辨率数字视频的广泛需求,对当前解决视频实时处理的硬件设备提出了更高的要求。而采用 FPGA 技术可以很容易地实现对视频的并行处理,并且,与专用于视频编解码的 ASIC 相比,基于 FPGA 实现的视频编解码设备具有高性能、低成本、易升级和开发周期短等特点,这些特点使得使用 FPGA 实现视频编解码成为一种趋势。

本论文所研究的课题是在山东科技大学嵌入式系统与集成电路设计实验室所承担的青岛市科技计划项目:“网络关键技术研究-基于 OR1200 嵌入式 SoC 网关集成电路的设计及 AVS 实现(编号:07-2-3-1-jch)”的研究背景下进行的。本设计采用 Altera 公司的 Cyclone II EP2C70F896C6N 作为验证平台。

1 帧内预测在图像压缩中的作用

帧内预测技术在视频压缩中对去除空间冗余信息起到了重要的作用。绝大多数图像都有一个共同的特征:内容缓变区域占据图像的绝大部分,相邻像素的像素值相差很小。根据预测编码理论,在图像传输或存储时,没有必要传输或存储图像的全部信息,而只需要传输或存储图像像素之间的差异即可,这样可以利用更少的位数来表示图像信息,实现了对图像的压缩。在图像压缩中,人们广泛的采用预测编码技术^[1]。

在进行图像压缩时,首先要对图像数据进行预测,得到预测数据,然后再与原始数据相减即可得到残差。相反,在进行图像复原时,再次进行预测得到预测图像,预测图像加上残差即可得到原始图像。

2 帧内预测在 AVS 编码标准中的实现技术

简单的说帧内预测就是利用已知的像素值去预测未知的像素值。在 AVS 编码

标准中，对图像的处理都是基于宏块进行的，每个宏块可进一步划分为 8×8 块，如图 2-1 所示，对于 4:2:0YUV 格式的图像的宏块可划分成 4 个亮度块和 2 个色度块。因此，AVS 标准所处理的最小单元就是 8×8 块¹。帧内预测正是利用前面已解码的块对当前未知的块进行预测的。

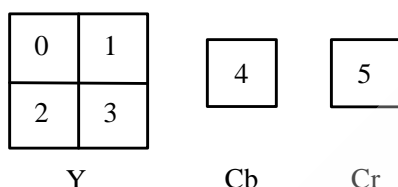


图 2-1 4:2:0YUV 格式的一个宏块的划分

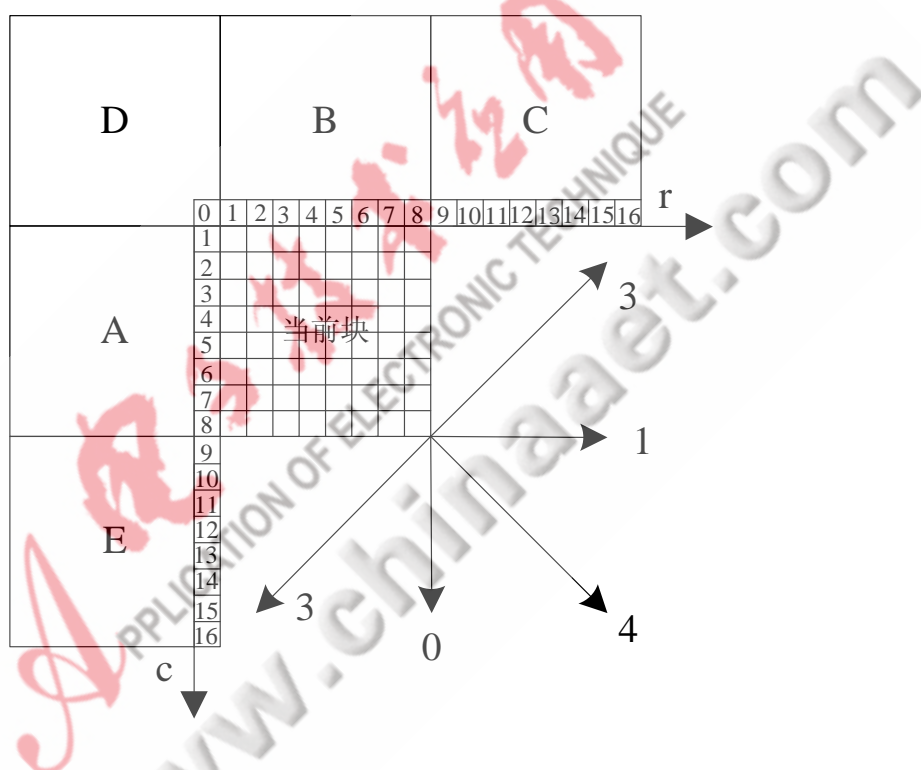


图 2-2 可利用的相邻块及参考样本，以及亮度预测模式的预测方向

如图 2-2 所示，在 AVS 标准中，若要对当前块进行预测则需要知道与其相邻的左(A)、上(B)、上左(C)、左下(E)和左上(D)块的参考样本值。AVS 标准是根据这些块的“可用性²”来获得参考样本的。获得参考样本后，要根据当前块与相邻块的相关性以及相邻块的存在性来决定采用什么样的预测模式来进行预测。AVS 标准中，亮度块采用了 5 种预测模式，色度块采用了 4 种预测模式，如表 2-1、2-2 所示。每一种预测模式所代表的预测方向如图 2-2 所示。根据预测模式

¹ 下面提到的块都是指的 8×8 块。

² 如果某个图像样本所在的块“不存在”或此样本尚未解码，则此样本‘不可用’

进行预测计算的运算法则请参考文献[2]。

表 错误！文档中没有指定样式的文字。-1 8x8 亮度预测模式

亮度预测模式	名称
0	Intra_8x8_Vertical
1	Intra_8x8_Horizontal
2	Intra_8x8_DC
3	Intra_8x8_Down_Left
4	Intra_8x8_Down_Right

表 错误！文档中没有指定样式的文字。-2 8x8 色度预测模式

色度预测模式	名称
0	Intra_Chroma_DC
1	Intra_Chroma_Horizontal
2	Intra_Chroma_Vertical
3	Intra_Chroma_Plane

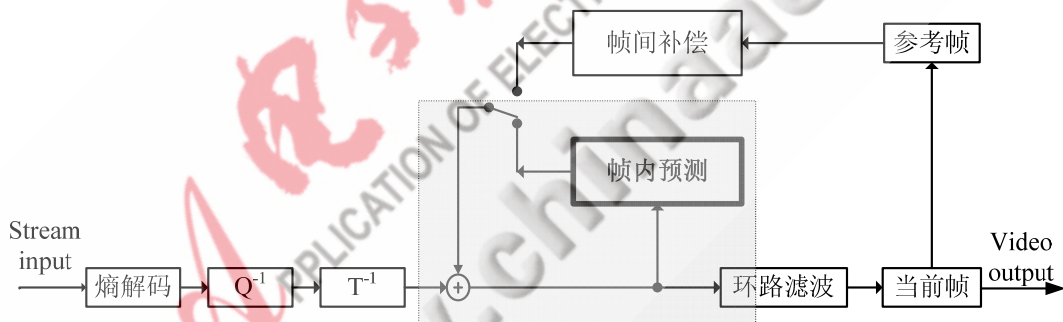


图 2-3 解码流程

基于 AVS 标准的视频解码流程如图 2-3 所示。从解码流程中可以看出，帧内预测的预测值要与 IQIT（反量化/反变换）后得到的残差值进行重构，重构后的结果再次进入帧内预测模块，作为其他模块预测的参考样本值。

3 帧内预测的模块划分及硬件化设计

根据帧内预测实现算法可知，若要对当前块进行预测，则需要获得相邻块的参考样本，再根据预测模式采用不同的预测法则对当前块进行预测，获得参考样本过程同样需要知道预测模式。因此，帧内预测模块可划分为模式判别模块、参考样本管理模块和预测值计算模块。如图 3-1 所示。本设计根据信号的需求关系，

按照先预测值计算模块，再参考样本管理模块，再模式判别模块的顺序进行的设计，这样可以容易定义各个模块的信号接口。

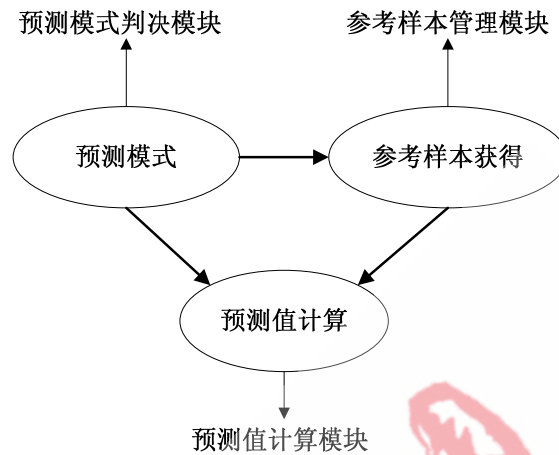


图 3-1 帧内预测模块划分

3.1 预测值计算模块

预测值计算模块是整个帧内预测模块的核心计算模块。预测值计算模块实现的功能是利用参考样本在不同的预测模式下采用不同的预测法则计算出当前帧的预测值。通过研究各个模式所对应的运算法则（请参考文献[2]），可以发现这些运算法则中有一个共同的特点：它们都有一个形如 $(a+2*b+c+2) \gg 2$ 的运算单元。这样在设计过程中就可以设计一个可重构在各种模式中的运算单元 PE(Processing Element)。在文献[3][4][5]中分别介绍了各自对 PE 的设计方法。通过对他们所设计 PE 的研究，发现他们所设计的 PE 在不同程度上存在着关键路径长，占用资源多等缺点。然而，PE 占据着预测值计算模块的关键路径，因此，减少 PE 的关键路径对提高帧内预测模块的运行频率起到了重要作用。本设计正是基于这种考虑重新对 PE 进行了设计。如图 3-2 所示，是本设计中所设计的 PE。此 PE 具有关键路径短，占用资源少等特点，有利于流水线设计。

此运算单元 PE 可重构到帧内预测的各个模式中。通过对帧内预测运算法则的研究，可以将运算法则根据预测模式分为以下三类情况：

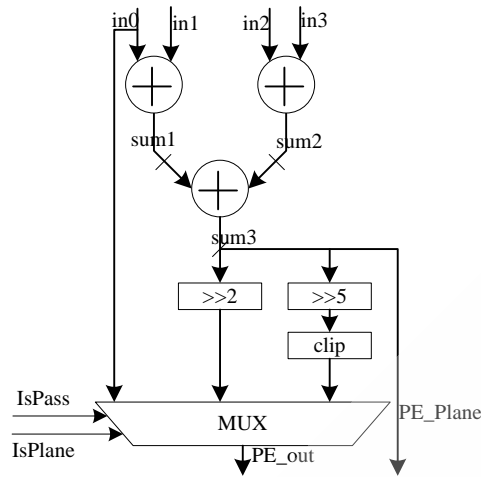


图 3-2 运算单元 PE(Processing Element)

- 1) 亮/色度的垂直、水平预测模式，Intra_8x8_Down_Right 和上/左参考样本不能同时可用的亮/色度 DC 模式。

在这些预测模式中，对一个像素进行预测只需要一个 PE 单元即可。因此，可以利用 8 个并行的 PE 单元并行计算，这样一个时钟即可计算出一行的预测值，8 个时钟即可计算完一个块。

- 2) Intra_8x8_Down_Left 和左、上参考样本同时可用的亮度色度 DC 模式。

在这些预测模式中，对一个像素进行预测则需要两个 PE 单元才能完成。为了不增加硬件资源，在对这些模式处理时，仍然采用了 8 个并行 PE 单元并行处理。本设计中采用了如图 3-3 所示的解决办法实现预测值得计算。

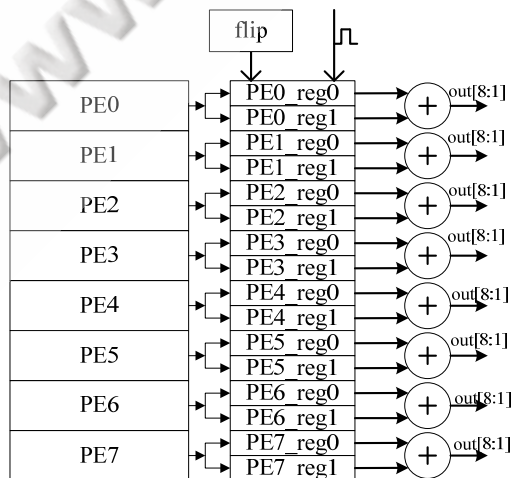


图 3-3 对需要两个 PE 的模式采用的设计结构

根据图 3-3 所示的设计结构，在第一个时钟是运算出运算法则中的一个 PE，然后寄存在寄存器组中。在第二个时钟时计算出另一个 PE，同样寄存在寄存器组中，然后经过加和移位得到计算值。根据流水线特点，两个时钟才能计算出一个像素的预测值。计算完一个块则需要 16 个时钟。

3) 色度块的 Intra_Chroma_Plane 模式。

对于色度块的 Intra_Chroma_Plane 模式，它的运算法则与上面提到的其他模式的运算法则有一定的区别，它的运算法则为：

$$\text{predMatrix}[x,y] = \text{clip1}((\text{ia}+(x-3)\times\text{ib}+(y-3)\times\text{ic}+16)\gg 5) \quad (x,y=0\sim 7)。$$

其中， $\text{ia} = (r[8]+c[8])\ll 4$, $\text{ib} = (17\times\text{ih}+16)\gg 5$, $\text{ic} = (17\times\text{iv}+16)\gg 5$,

$$\text{ih} = \sum_{i=0}^3 (i+1)\times (r[5+i]-r[3-i]) \quad , \quad \text{iv} = \sum_{i=0}^3 (i+1)\times (c[5+i]-c[3-i])$$

在文献 [3][4][5][6][7] 中都介绍了同一种解决方案，即将 $\text{clip1}((\text{ia}+(x-3)\times\text{ib}+(y-3)\times\text{ic}+16)$ 的运算进行了解析：

$$\text{clip1}((\text{ia}-3\text{ib}-3\text{ic}+16+x\times\text{ib}+y\times\text{ic}))\gg 5) \Rightarrow \text{clip1}((A+x\times\text{ib}+y\times\text{ic})\gg 5)$$

其中 $A = \text{ia} - 3\text{ib} - 3\text{ic} + 16$ 。对于一个确定的模块 $\text{ia}, \text{ib}, \text{ic}$ 都是确定的值。这样对一行的像素进行预测计算的表达式就可表示为：

$$\text{predMatrix}[0,0] = \text{clip1}((A + 0\times\text{ib} + j\times\text{ic})\gg 5);$$

$$\text{predMatrix}[1,0] = \text{clip1}((A + 1\times\text{ib} + j\times\text{ic})\gg 5);$$

$$\text{predMatrix}[2,0] = \text{clip1}((A + 2\times\text{ib} + j\times\text{ic})\gg 5);$$

.....

$$\text{predMatrix}[7,0] = \text{clip1}((A + 7\times\text{ib} + j\times\text{ic})\gg 5);$$

根据这些表达式的特点，在计算第 0 行时，需要计算 $A + i\times\text{ib}$ 。以后，在计算其他行时，依次在上一行的基础上加上 ic 实现。这些加和运算的过程可以利用 PE 单元的四输入加法运算来实现。为了更好的将 PE 单元应用到 Intra_Chroma_Plane 模式，在对 PE 单元进行设计时，加入了可适应于 Plane 模式的移位和 $\text{clip1} (0\leq X\leq 255)$ 操作，如图 3-2 所示。

在进行帧内预测值计算之前，关键是要计算出 A 、 ib 、 ic 的值。在本设计中专门设计了一个 Plane 值计算模块(Plane_calc)。当预测模式为 Intra_Chroma_Plane

模式时，会自动的启动这一模块。计算 A、ib、ic 的关键是计算 ih、iv，根据 ih、iv 的计算特点，本设计中采用了文献[3]所采用的设计结构进行设计的。如图 3-4 所示。

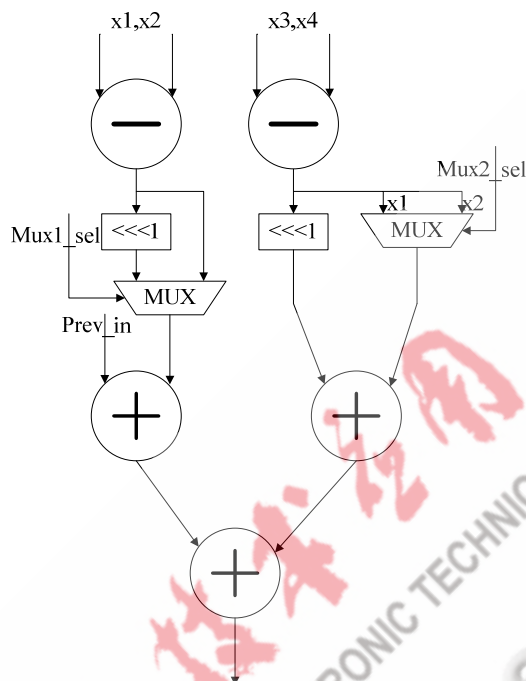


图 3-4 ih、iv 的计算结构图

在左半边 $\text{mux1_sel}=1$ 可以对 $x1$ 进行运算， $\text{mux1_sel}=0$ 可以对 $x2$ 的进行运算。在右半边 $\text{mux2_sel}=1$ 可以对 $x3=x1+x2$ 进行运算， $\text{mux2_sel}=0$ 可以对 $x4=x2+x2$ 进行运算。为了实现累加，将每一次的运算结果由 prev_in 再传回给运算单元。利用这个运算单元可以在两个时钟内就能计算完 $\text{ih}(\text{iv})$ 。

通过上面对各种模式的分析，可以确定本设计的 PE 可以充分的重构到每一种模式中。本设计利用 PE 所设计的预测值计算模块的整体设计结构如图 3-5 所示。

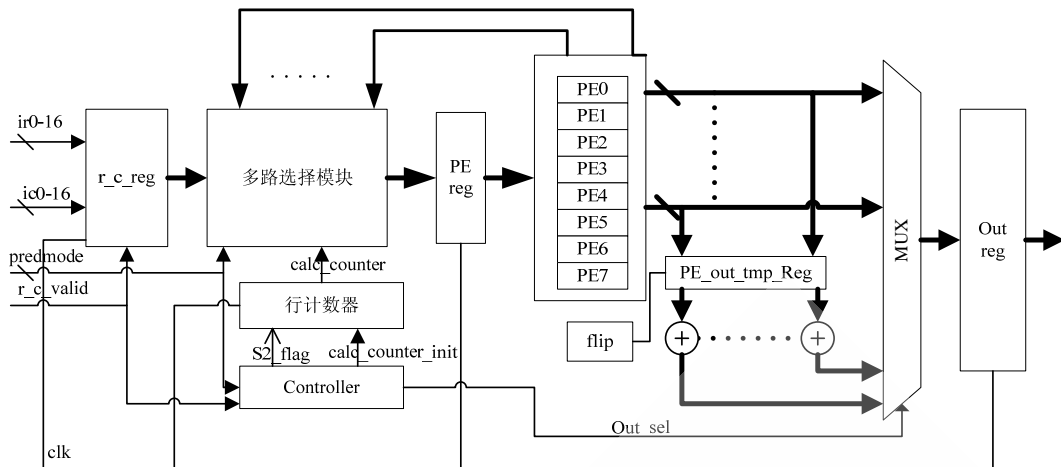


图 3-5 帧内预测值计算模块的设计结构图

如图 3-5 所示，预测值计算模块采用了三级流水线设计。由于在进行计算时每个 PE 在每一行的输入值都是变化的，图 3-5 中的多路选择模块就是解决每个 PE 的输入问题的，多路选择模块在预测模式和行计数器的控制下对参考样本数据进行选择输入到 PE 单元进行计算。对于需要两个时钟计算一行的模式，则需要自适应的加入一级加和和移位操作。

3.2 参考样本管理模块

参考样本管理模块的主要功能有两个：一、为预测值计算模块提供参考样本；二、如何管理当前块的重构值以便更好的为其他块的预测提供参考样本。

AVS 编解码是以宏块为单位按照光栅扫描顺序进行的。如图 3-6 所示，通过分析一个块的周围环境可以发现，对于每个块的最左边的一列重构数据可以很快应用到以后块的预测中。对于 blk0 和 blk1 的最下面那一行的重构数据也可以很快的应用到以后块的预测中去。然而，对于 blk2、blk3、blk4 和 blk5 最下一行的重构数据则要到预测到下一行宏块的 blk0、blk1、blk4 和 blk5 时才用到。若是对所有的重构数据都采用寄存器存储，虽然参考样本加载速度快，但是对于高清视频的解码器来说就会占用大量的 FPGA 资源。因此，本设计采用折中方法。对于很快就被应用到其他块预测的重构数据存储于寄存器中，而对于不能很快应用到预测中的重构数据则存储在 ram 中。这样既节省了 FPGA 资源，同时也不会影响速度。

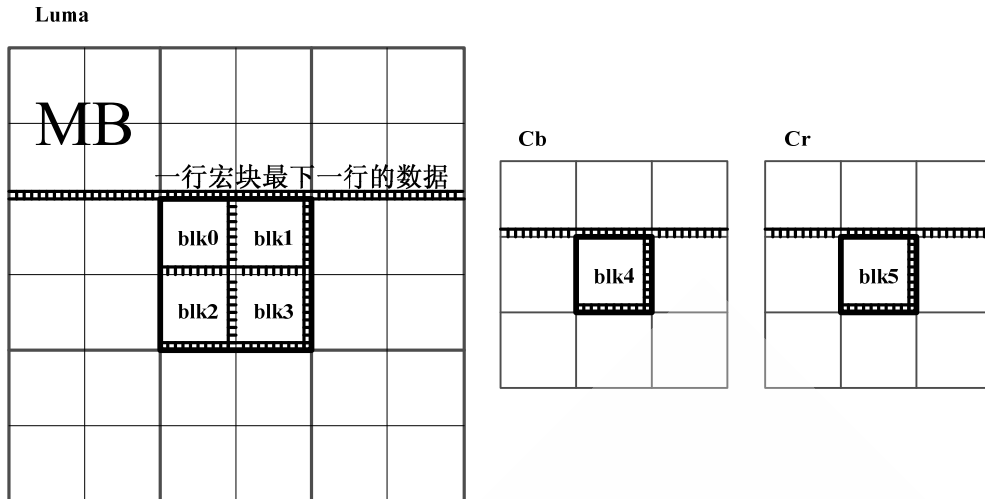


图 3-6 一幅图像以宏块为单位的划分

如图 3-7 所示，本设计采用了分层次样本管理方案。对于很快应用到以后块的重构数据存储到寄存器中，对于 blk2、blk3、blk4 和 blk5 最下一行的重构数据则需要存储到片上 ram 中去。

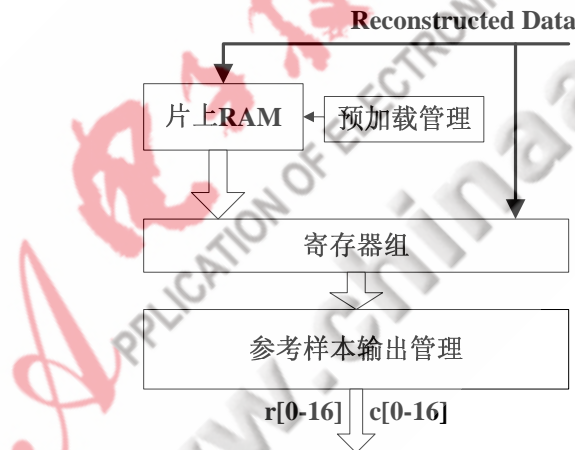


图 3-7 分层次样本管理方案

如 3-6 所示，若要对 blk0、blk1、blk4 和 blk5 块进行预测时，需要从片上 ram 加载参考样本，这样会消耗一定的加载时间，为了提高预测速度，本设计中采用了预加载管理方法。如图 3-8 所示，对于 blk0、blk1、blk4 和 blk5 分别设置了两个参考样本预加载寄存器。每当 blk0、blk1、blk4 和 blk5 中的一个进行预测时取走了参考样本后，预加载寄存器就立即加载下一个相应块的参考样本，这样当预测下一个相应块时可以直接加载，而节省时间。当一行的数据加载完之后，预加载器会自动的循环到片上 ram 的开始位置加载下一行宏块的参考样本。实现

了一种类似循环的参考样本加载方式，如图 3-8 所示。

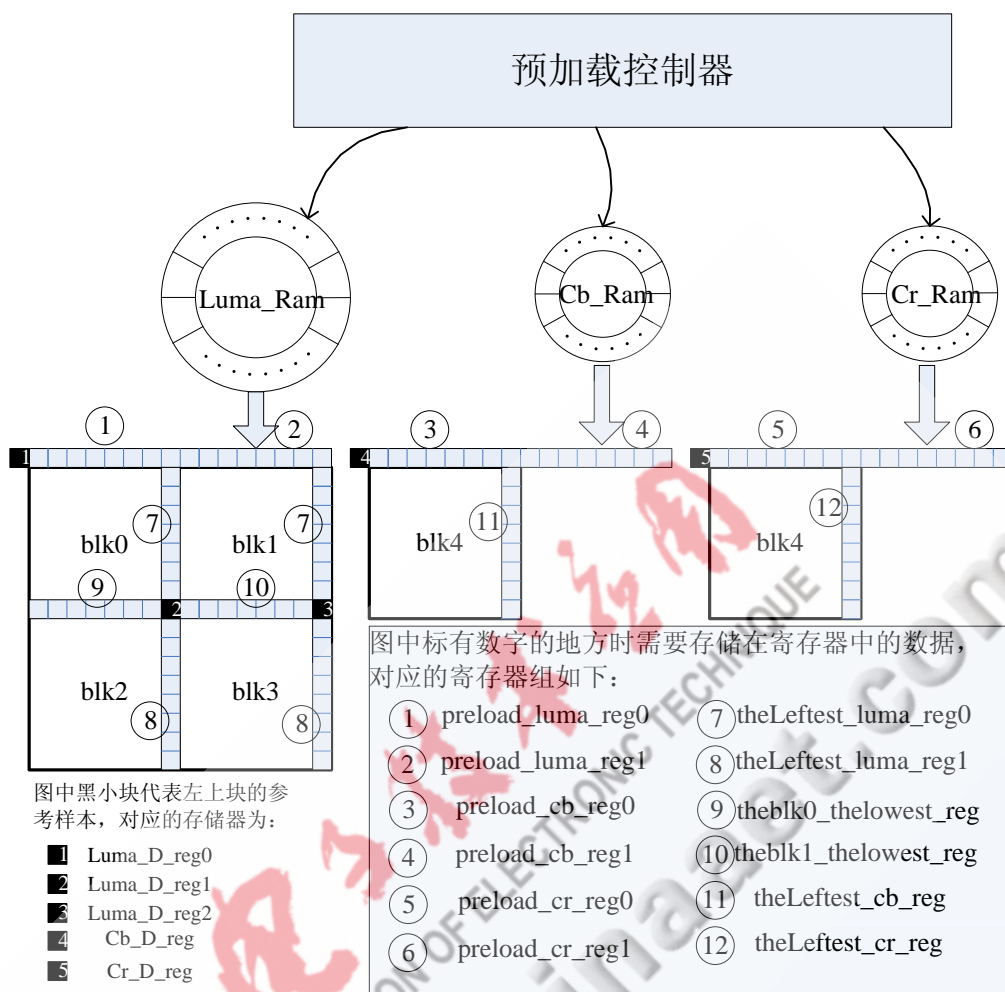


图 3-8 参考样本管理方案

3.3 模式判别模块

对于解码端来说，一个块的预测模式可以通过编码码流中的语法元素和上、左相邻块的预测模式来决定。帧内预测模式算法相对简单，但是若要采用纯组合电路来实现势必造成关键路径较长，为了提高系统运行频率，以及更好实现同步，在对模式判别中采用了有限状态机来指示整个预测流程，如图 3-9 所示。

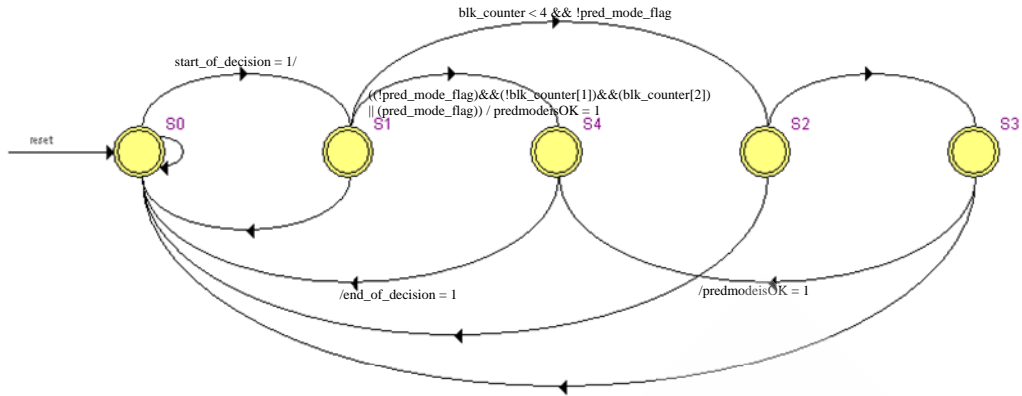


图 3-9 模式判决有限状态转换图

模式判决的流程：当模块接到 `start_of_decision` 信号后，进入状态 S1。在 S1 状态，要判断当前块是色度块还是亮度块，若是亮度块，还要判断 `pred_mode_flag` 标志。若是色度块或 `pred_mode_flag = 1`，那么预测模式直接由编码码流中的语法元素来决定。否则，就会进入 S2 状态。在 S2 状态，要根据上、左边相邻块的预测模式得到 `predIntraPredMode` 值，然后自动转入状态 S3。在 S3 要根据编码码流中的 `Intra_luma_pred_mode` 的值和 `predIntraPredMode` 的值进行获得预测模式。在 S4 状态中，要对预测模式进行编码，这样可以在预测值计算模块中不用再对预测模式进行处理了。

4 帧内预测模块的总体设计

在对上一节中的所有模块进行设计时都设置了交互信号，因此利用一个有限状态机就可以很容易地对他们进行控制。本设计所设计的帧内预测模块的总体结构图如图 4-1 所示。

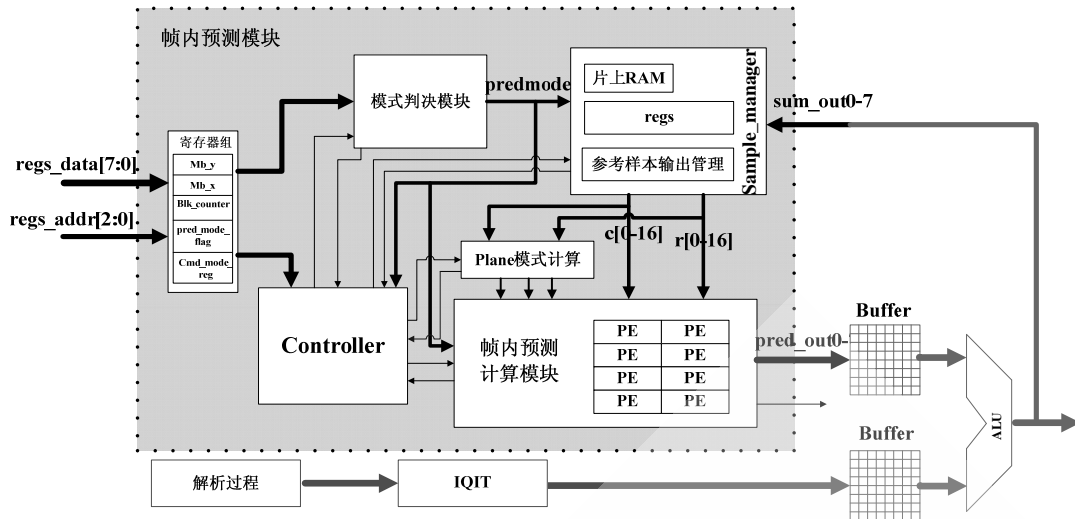
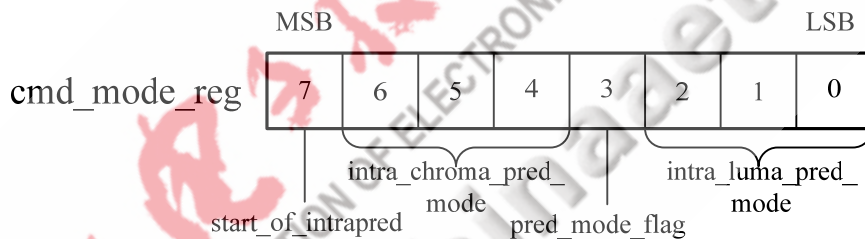


图 4-1 帧内预测总体结构图

如图 4-1 所示，在帧内预测模块中，通过一个控制器实现了对所有模块的控制。在帧内预测模块的实现上增加了一组寄存器组，减小输入信号，便于与其他总线进行交互。其中，cmd_mode_reg 寄存器的每一位的介绍如下：



其中，intra_luma_pred_mode、pred_mode_flag、intra_chroma_pred_mode 是码流中的语法元素，start_of_intrapred 是帧内预测模块的启动信号，此信号会维持一个时钟周期后自动清零。

本设计采用 Verilog HDL 语言对帧内预测模块的硬件进行了描述，利用 Altera 公司提供的 Quartus II 9.0 软件进行综合和分析。图 4-2 是由 Quartus II 9.0 软件综合生成的帧内预测顶层模块的符号图(Symbol)。表 4-1 对帧内预测模块的输入输出接口进行了说明。表 4-2 对帧内预测模块中寄存器地址进行说明。

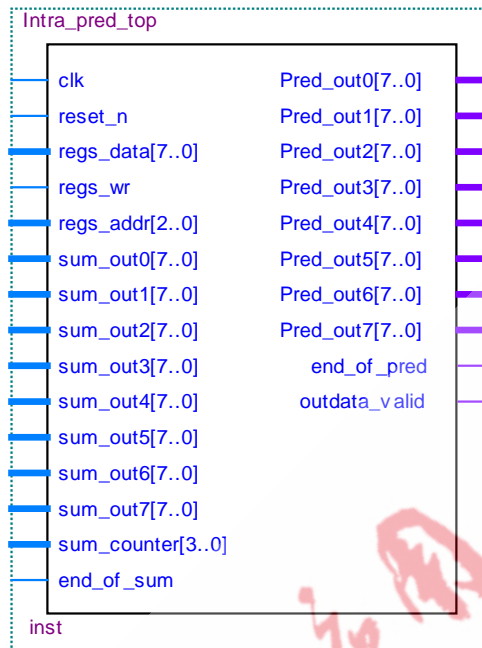


图 4-2 帧内预测顶层模块 symbol

表 4-1 帧内预测模块的输入输出接口定义

Port	Width	Direction	Description
Clk	1	Input	系统时钟
reset_n	1	Input	系统复位，低电平有效
regs_data	7	Input	寄存器组输入数据
regs_wr	1	Input	写寄存器组有效
regs_addr	3	Input	寄存器地址
end_of_sum	1	Input	一个块的重构结束信号
sum_out0-7	7	Input	重构数据输入，一次输入一列
sum_counter	4	Input	指示当前输入的 sum_out0-7 是第几列数据
Pred_out0-7	7	Output	预测数据输出，一次输出一行
end_of_pred	1	Output	一块预测结束
outdata_valid	1	Output	输出数据有效

表 4-2 帧内预测模块中寄存器地址对应表

寄存器	地址	说明
blk_counter	000	当前宏块中块的索引
Mb_x	001	当前宏块水平位置
Mb_y	010	当前宏块垂直位置
cmd_mode_reg	011	cmd_mode_reg 寄存器

帧内预测模块的资源利用情况，如图 4-3 所示。通过简单的时序分析证明本设计的帧内预测可以工作在 106.29MHz。

```
Fitter Status                Successful - Tue Aug 11 10:44:52 2009
Quartus II Version          9.0 Build 132 02/25/2009 SJ Full Version
Revision Name               Intrapred_top
Top-level Entity Name      Intrapred_top
Family                     Cyclone II
Device                     EP2C70F896C6
Timing Models              Final
Total logic elements       5,767 / 68,416 ( 8 % )
  Total combinational functions 5,507 / 68,416 ( 8 % )
  Dedicated logic registers  2,169 / 68,416 ( 3 % )
Total registers            2169
Total pins                 165 / 622 ( 27 % )
Total virtual pins         0
Total memory bits          16,384 / 1,152,000 ( 1 % )
Embedded Multiplier 9-bit elements 0 / 300 ( 0 % )
Total PLLs                 0 / 4 ( 0 % )

Info: Clock "clk" has Internal fmax of 106.29 MHz between source register "
```

图 4-3 帧内预测模块的综合报告及简单的时序分析

5 帧内预测模块的系统测试

5.1 测试方案介绍

本设计采用的是自顶向下和自下而上的混合设计方法。在帧内预测模块设计过程中，严格地按照设计一个模块验证一个模块的原则来进行的。但是，帧内预测的过程是一幅图像前后相互联系的过程。局部的测试很难判断出帧内预测模块是否可以应用到整个解码系统中去。从帧内预测模块的输入输出数据来看，若是将帧内预测模块作为单独的模块加到 SoPC 系统中应用于解码，势必会造成数据量访问频繁，影响解码速度。通过分析解码流程，可以将帧内预测模块与 IQIT 模块和 Sum(重构)模块组合在一起，然后再通过 Avalon Slave Interface 挂到 SoPC 系统上，这样就可以避免因访问外设频繁而造成的速度下降问题。如图 5-1 所示。

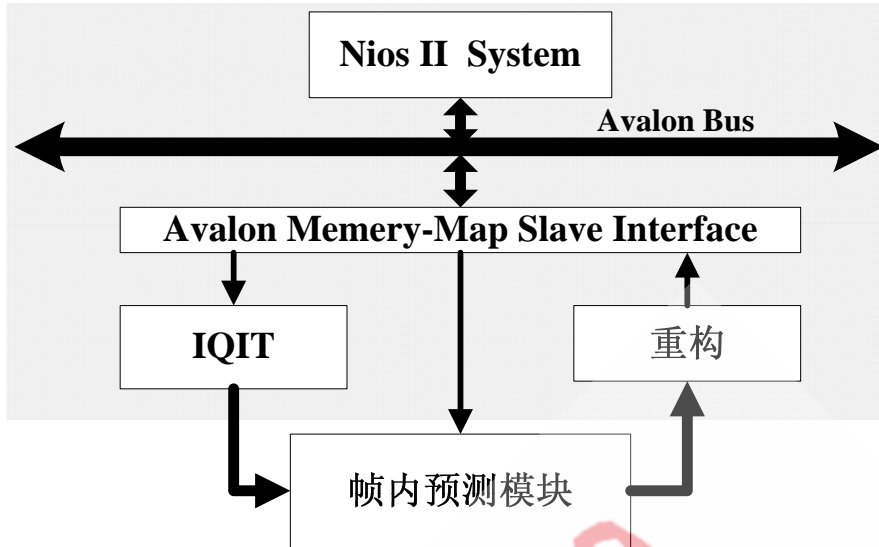


图 5-1 帧内预测模块的系统测试方案

5.2 测试环境介绍

测试硬件平台为台湾友晶公司生产的基于 Altera 公司的 Cyclone II EP2C70F896C6N 的 DE2-70 开发平台。由 Nios II 搭建的 SoPC 系统如图 5-2 所示。Nios II 核采用的是 Fast 型，系统的存储空间为 32M 的 SDRAM，系统的工作频率为 100Mhz。

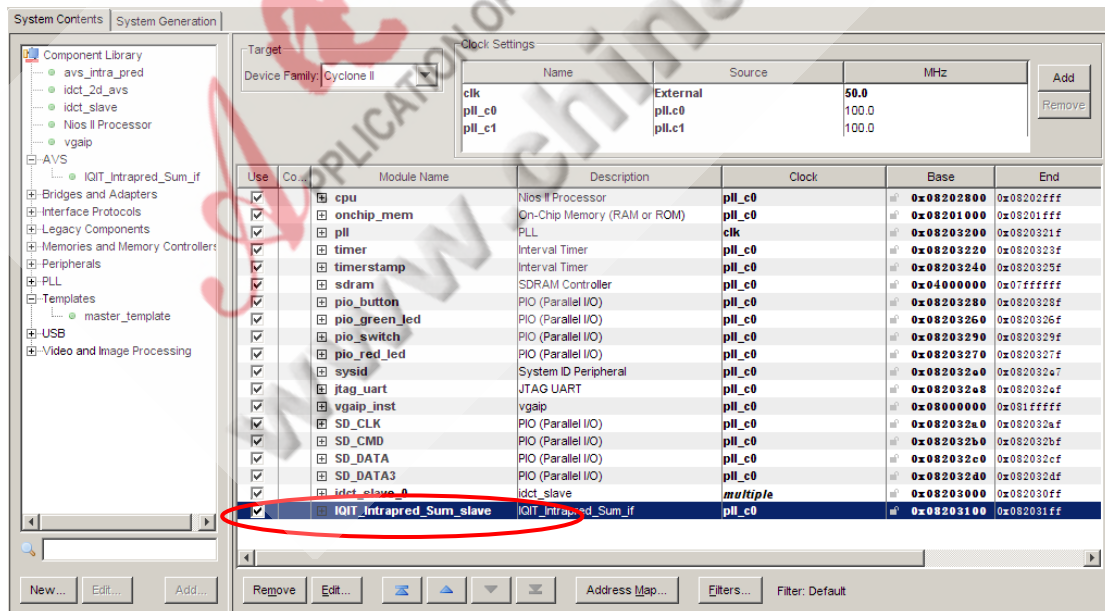


图 5-2 测试所搭建的 SOPC 系统

测试软件及码流是由北京联合信源发布的 AVS1-P2 解码软件 P2 版和相应的测试码流（<http://cosoft.org.cn/projects/avsdec>）。在进行测试时需要

McIdctRecOneMacroBlock 函数中的 inv_transform_B8 函数、ReconB8 函数以及亮度色度帧内预测函数进行修改(下面只给出了亮度，色度也是相似的)。

```

IOWR(IDCT_INTRAPRED_SUM_BASE,MB_Y,MbY);//写Mb_y寄存器
IOWR(IDCT_INTRAPRED_SUM_BASE,MB_X,MbX);//写Mb_x寄存器
//亮度预测，变换，重构
for(block=0; block<4; block++)
{
    LumaResidual_offset = pLumaResidual+block*64;
    //写blk_counter寄存器
    IOWR(IDCT_INTRAPRED_SUM_BASE,BLK_INDEX,block);

    cmd_mode_reg = 0x80 | (pMbInfo[dwMbIndex].bPredModeFlag[block]<<3)|
                    (pMbInfo[dwMbIndex].iIntraLumaPredMode[block]);
    //写start信号和predmode语法元素
    IOWR(IDCT_INTRAPRED_SUM_BASE,CMD_MODE,cmd_mode_reg);
    //写残差数据
    if(pMbInfo[dwMbIndex].dwCbp & (1<<block))
    {
        IOWR(IDCT_INTRAPRED_SUM_BASE,CBP,1);

        for(y=0;y<8;y++)
        {
            offset = LumaResidual_offset + y*8;
            for(x=0;x<8;x++)
            {
                IOWR(IDCT_INTRAPRED_SUM_BASE,IDCT_INPUT +
                    x,*(offset + x));
            }
        }
    }
    else
    {
        IOWR(IDCT_INTRAPRED_SUM_BASE,CBP,0);
    }
    Luma_blk_base = (pbTopLeftY+(imgY)*iImgWidth+imgX);
    //读出重构数据
    for(y=0;y<8;y++)
    {
        Luma_blk_base_offset = Luma_blk_base + y * iImgWidth;
        for(x=0;x<8;x++)
        {
            *(Luma_blk_base_offset + x)=
                IORD(IDCT_INTRAPRED_SUM_BASE,y*8 + x);
        }
    }
}

```

```

    }
}
} //for(block=0;block<4;block++)

```

5.3 测试结果

利用 Nios IDE 的文件系统可以将解码后的 rgb 数据通过文件方式写到 PC 机上。将解码后的 rgb 数据经过简单的处理转换成 bmp 格式的文件，如图 5-4、5-5 所示。通过与 PC 机上软件的运行结果进行对比可以证明本设计的帧内预测模块可以完成帧内预测功能。

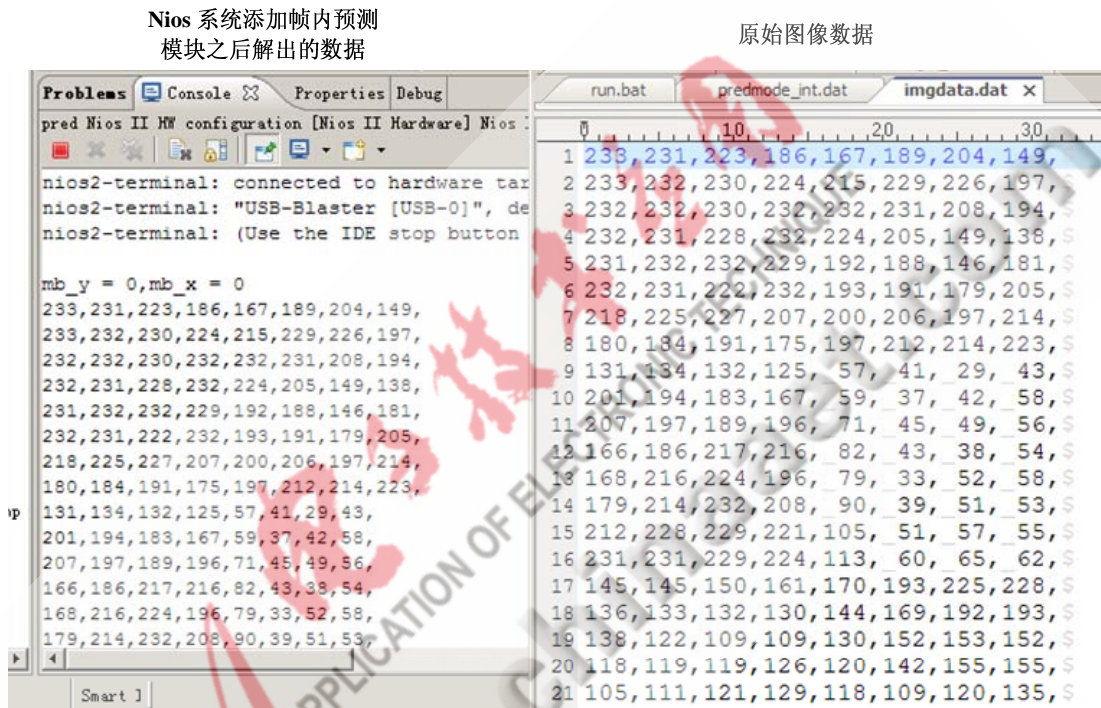


图 5-3 NiosII 系统运行解码输出数据与 PC 机软件解码输出数据对比

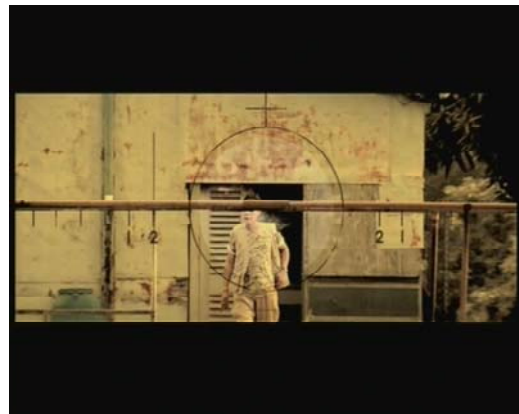
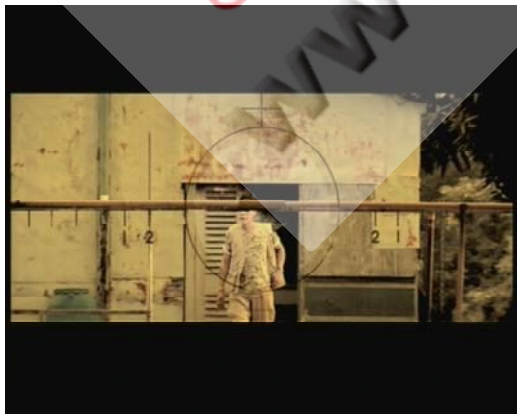
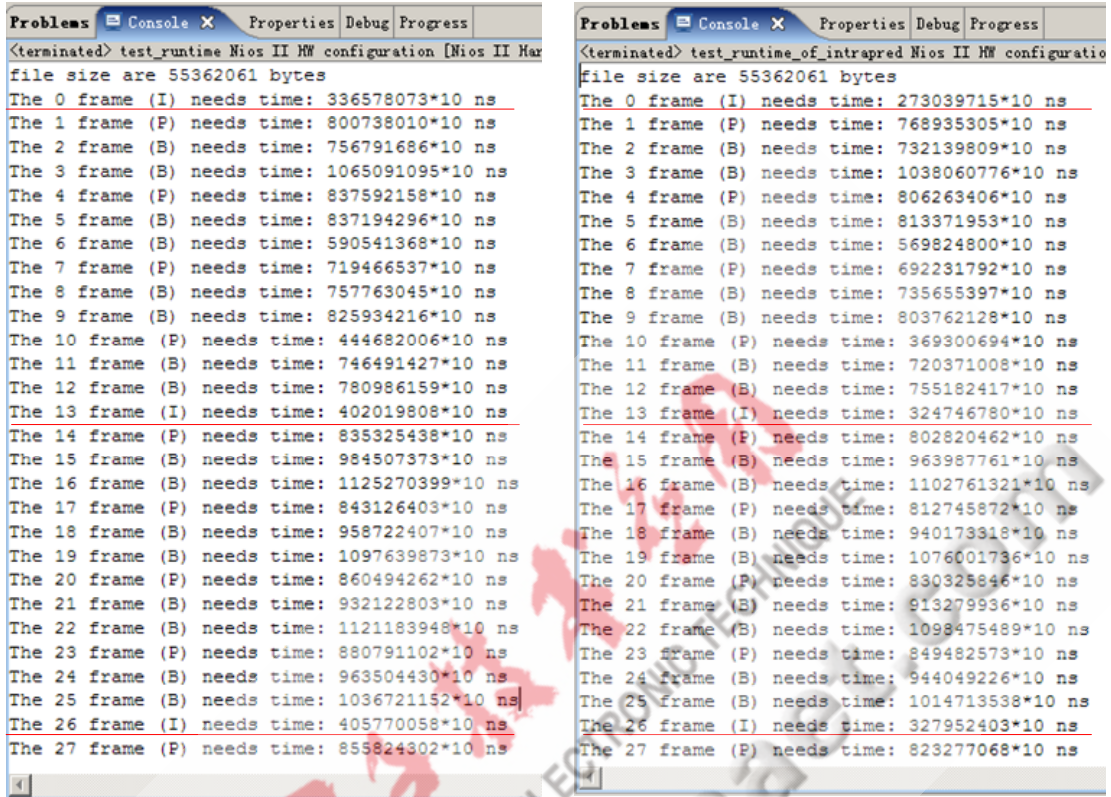


图 5-4Nios 系统解码输出数据转换到 bmp 图 5-5PC 机解码输出数据转换到 bmp

5.4 性能分析

数据统计³:



Frame Type	Needs Time (ns)
The 0 frame (I)	336578073*10 ns
The 1 frame (P)	800738010*10 ns
The 2 frame (B)	756791686*10 ns
The 3 frame (B)	1065091095*10 ns
The 4 frame (P)	837592158*10 ns
The 5 frame (B)	837194296*10 ns
The 6 frame (B)	590541368*10 ns
The 7 frame (P)	719466537*10 ns
The 8 frame (B)	757763045*10 ns
The 9 frame (B)	825934216*10 ns
The 10 frame (P)	444682006*10 ns
The 11 frame (B)	746491427*10 ns
The 12 frame (B)	780986159*10 ns
The 13 frame (I)	402019808*10 ns
The 14 frame (P)	835325438*10 ns
The 15 frame (B)	984507373*10 ns
The 16 frame (B)	1125270399*10 ns
The 17 frame (P)	843126403*10 ns
The 18 frame (B)	958722407*10 ns
The 19 frame (B)	1097639873*10 ns
The 20 frame (P)	860494262*10 ns
The 21 frame (B)	932122803*10 ns
The 22 frame (B)	1121183948*10 ns
The 23 frame (P)	880791102*10 ns
The 24 frame (B)	963504430*10 ns
The 25 frame (B)	1036721152*10 ns
The 26 frame (I)	405770058*10 ns
The 27 frame (P)	855824302*10 ns

Frame Type	Needs Time (ns)
The 0 frame (I)	273039715*10 ns
The 1 frame (P)	768935305*10 ns
The 2 frame (B)	732139809*10 ns
The 3 frame (B)	1038060776*10 ns
The 4 frame (P)	806263406*10 ns
The 5 frame (B)	813371953*10 ns
The 6 frame (B)	569824800*10 ns
The 7 frame (P)	692231792*10 ns
The 8 frame (B)	735655397*10 ns
The 9 frame (B)	803762128*10 ns
The 10 frame (P)	369300694*10 ns
The 11 frame (B)	720371008*10 ns
The 12 frame (B)	755182417*10 ns
The 13 frame (I)	324746780*10 ns
The 14 frame (P)	802820462*10 ns
The 15 frame (B)	963987761*10 ns
The 16 frame (B)	1102761321*10 ns
The 17 frame (P)	812745872*10 ns
The 18 frame (B)	940173318*10 ns
The 19 frame (B)	1076001736*10 ns
The 20 frame (P)	830328446*10 ns
The 21 frame (B)	913279936*10 ns
The 22 frame (B)	1098475489*10 ns
The 23 frame (P)	849482573*10 ns
The 24 frame (B)	944049226*10 ns
The 25 frame (B)	1014713538*10 ns
The 26 frame (I)	327952403*10 ns
The 27 frame (P)	823277068*10 ns

图 5-6 纯软件解码一帧数据所需时间 图 5-7 加入帧内预测硬件模块后所需时间

帧内预测主要用于 I 帧的压缩，因此，在解码 I 帧图像时，帧内预测模块的利用率较高，因此帧内预测硬件模块在 I 帧解码时起到的作用较大。如图 5-6、5-7 的对比，软件解码一个 I 帧平均需要 3.8146s，而加入帧内预测硬件模块后，解码一个 I 帧则平均需要 3.0751s，提高了 19.4% 的速度。P 帧和 B 帧主要采用的是帧间预测，但是 P、B 帧内会有一些宏块采用的是帧内预测，所以，帧内预测也相应的提高了 P、B 帧的解码速度。

6 总结

本设计对帧内预测算法进行了分析研究，同时还分析了目前他人在帧内预测硬件化方面的研究成果。本人在这些基础上，对帧内预测的硬件化重新进行了设计，并提出来了自己新的设计思想。本设计重新对运算单元 PE 进行了设计，具

³ 在前面的测试方案中提到是 IDCT、Intrapred 和重构三者一起进行测试的，但是，这无法说明帧内预测模块的速度问题，因此在做性能分析时，只加了 Intrapred 模块，因数据访问频繁可能会影响到解码速度。

有关键路径短, 占用资源少等特点。并且, 在设计过程中尽量考虑到了流水线设计, 使得整个帧内预测模块简单易控制。为了提高帧内预测的预测速度, 本设计中模拟了一种环形 ram, 实现预加参考样本功能, 提高了预测速度。

通过在基于 Nios II 的 SoPC 系统上测试, 可以证明帧内预测模块能够实现预测功能, 并且对 AVS 标准的视频解码速度的提高起到积极作用。

参考文献

- [1] 陈亮. AVS 先进编码技术研究[D], 湖北武汉: 华中科技大学, 2006
- [2] 数字音视频编码技术标准工作组. 信息技术先进音视频编码 第 2 部分: 视频 (GB/T 20090.2-2006), 中国国家标准化管理委员会, 2006
- [3] 姜伟, 王祖强. AVS 解码器自适应帧内预测的硬件实现[J], 计算机工程与应用, 2008, 44(36): 81-84
- [4] 王争, 刘佩林. AVS 帧内预测算法及其解码器的硬件实现[J], 计算机工程与应用, 2006, 19:80-83
- [5] 付丰华, 李凤亭. 并行结构的 AVS 帧内预测编码器[J], 计算机工程与设计, 2009, 30(5): 1140-1143
- [6] 纪洪芝. H.264 及 AVS 解码中帧内预测的硬件设计和 ASIC 实现[D], 陕西西安: 西安电子科技大学, 2007
- [7] 李镛的. AVS 视频编解码标准中预测编解码技术的硬件设计与实现[J], 电脑知识与技术, 2008, 3(6):1310-1311
- [8] Huang Yu-wen, Hsieh Bing-yu, Chen Tung-chien, et al. Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2005, 15(3): 378-401.

原创性声明

本人郑重声明, 此处提交的论文《AVS 视频解码中帧内预测模块的硬件化设计及 SoPC 验证》, 是本人在导师指导下, 在山东科技大学期间进行研究工作所取得的成果。据本人所知, 论文中除已注明部分外不包含他人已发表或撰写过的研究成果。本声明的法律效果将完全由本人承担。

作者签字: 刘家良 任怀鲁

日期: 2009 年 8 月 21 日