开源芯片
开源H.265/H.264视频编码器

范益波 复旦大学

- 个人简历
- 1999—2003：浙江大学 本科
- 2003—2006：复旦大学 硕士
- 2006—2009：早稻田大学 博士
- 2010年7月—至今：复旦大学
- 专用集成电路与系统国家重点实验室 副教授
- **成立视频图像处理实验室（VIP Lab）**
- **创立OpenASIC开源芯片社区**
- **发布开源H.265/H.264视频编码器IP核**
- 主要从事图像视频芯片研究

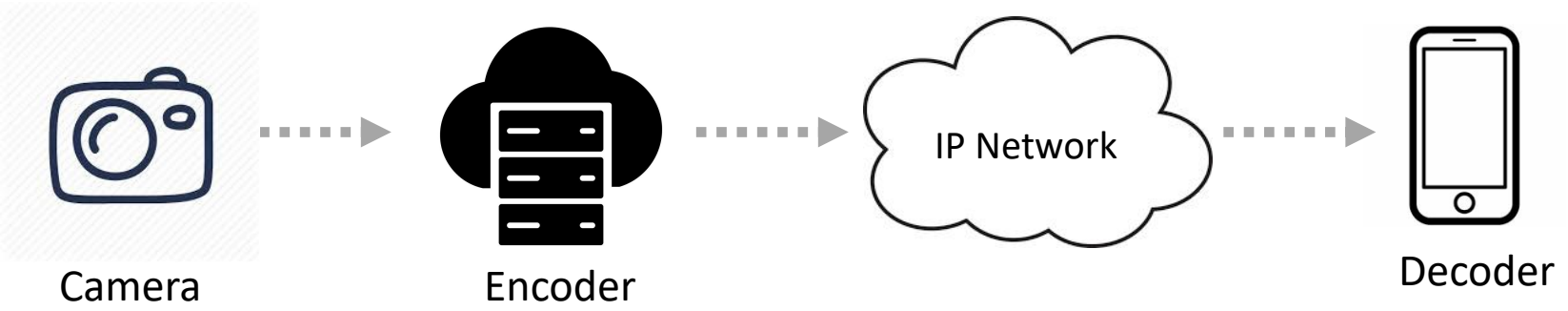
简介

主要从事图像处理、视频编解码、多媒体SoC系统的算法、硬件和芯片设计研究。作为课题负责人承担了国家自然科学基金、上海市科技创新项目、教育部博士点基金、专用集成电路与系统国家重点实验室面上项目等研究课题，并作为子课题负责人参与了国家863等重大重点项目的研究。以第一作者和通信作者在国内外重要的学术期刊与会议上发表（SCI/EI）论文80余篇，申请相关专利30多项。

目录

目录

视频传输过程

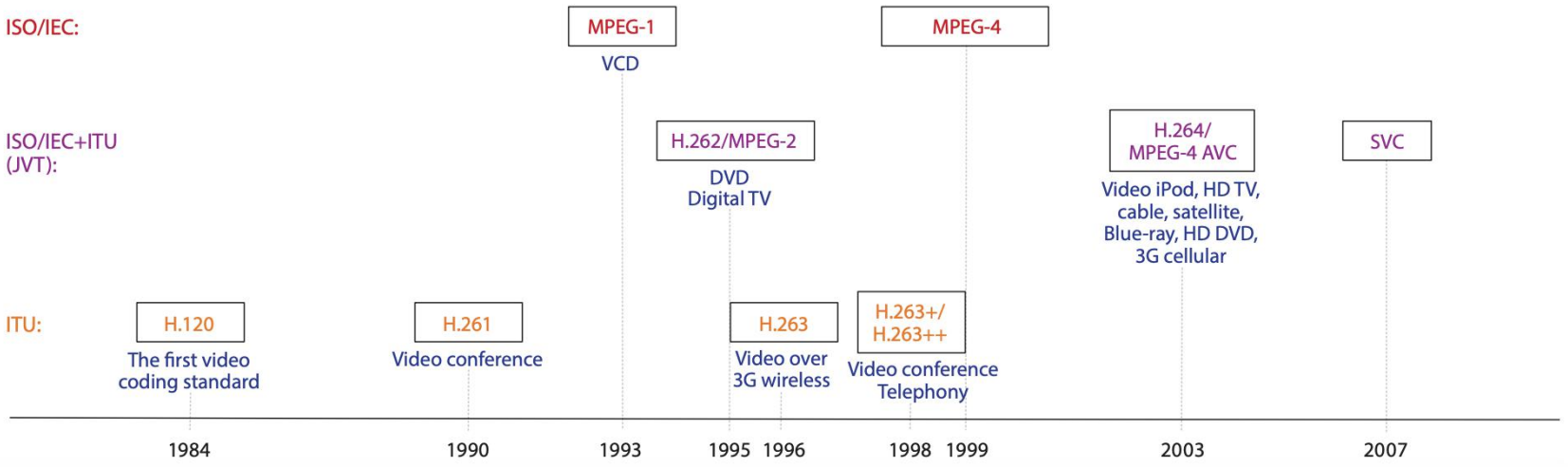


应用领域

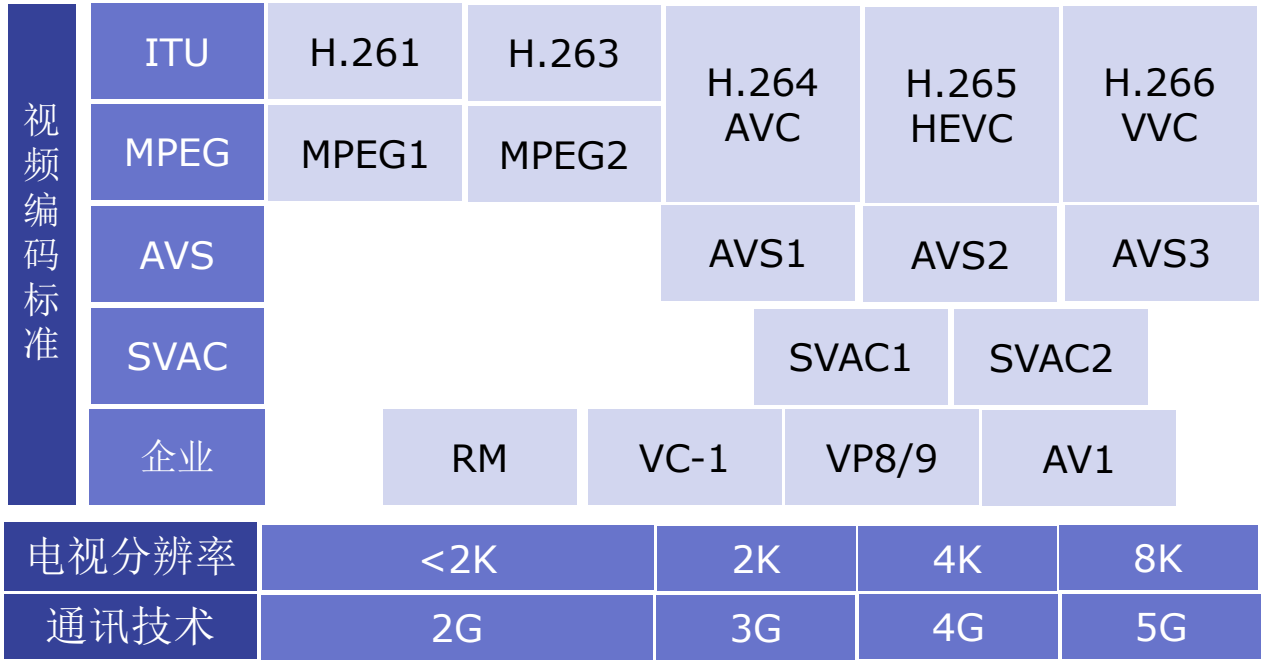




编码标准演进

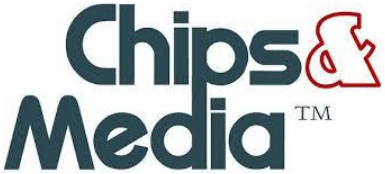

ISO/IEC:
MPEG-1
VCD
MPEG-4
ISO/IEC+ITU
(JVT):
H.262/MPEG-2
DVD
Digital TV
H.264/
MPEG-4 AVC
Video iPod, HD TV,
cable, satellite,
Blue-ray, HD DVD,
3G cellular
SVC
ITU:
H.120
The first video
coding standard
H.261
Video conference
H.263
Video over
3G wireless
H.263+/
H.263++
Video conference
Telephony
1984
1990
1993
1995
1996
1998
1999
2003
2007

标准演进趋势



视频编码标准、通讯标准、视频分辨率发展

视频编码IP现状



价格昂贵

无RTL源码

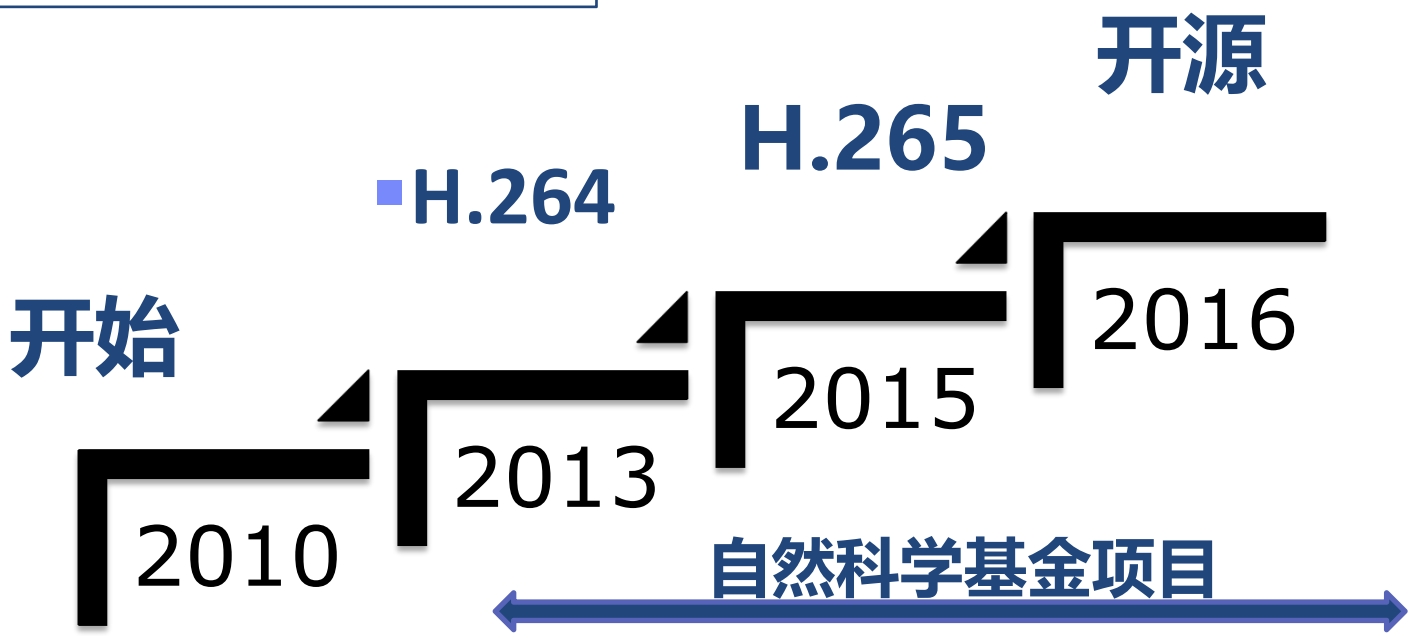
不开源

当前视频编码IP核提供商
（核心IP技术主要在国外）

开源项目由来

免费　RTL源码　可定制

性能较弱　麻雀虽小、五脏俱全

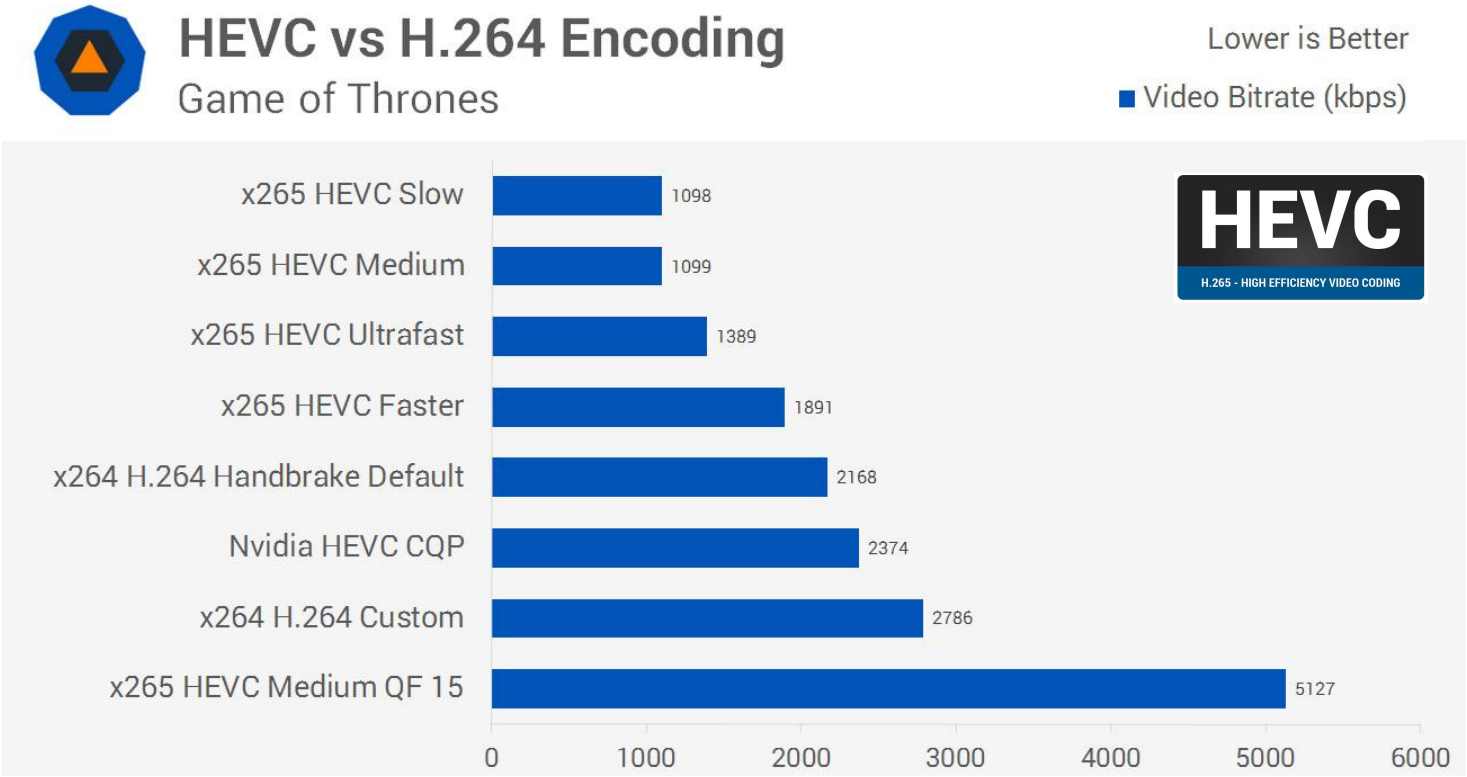


目标：HD/4K@30fps

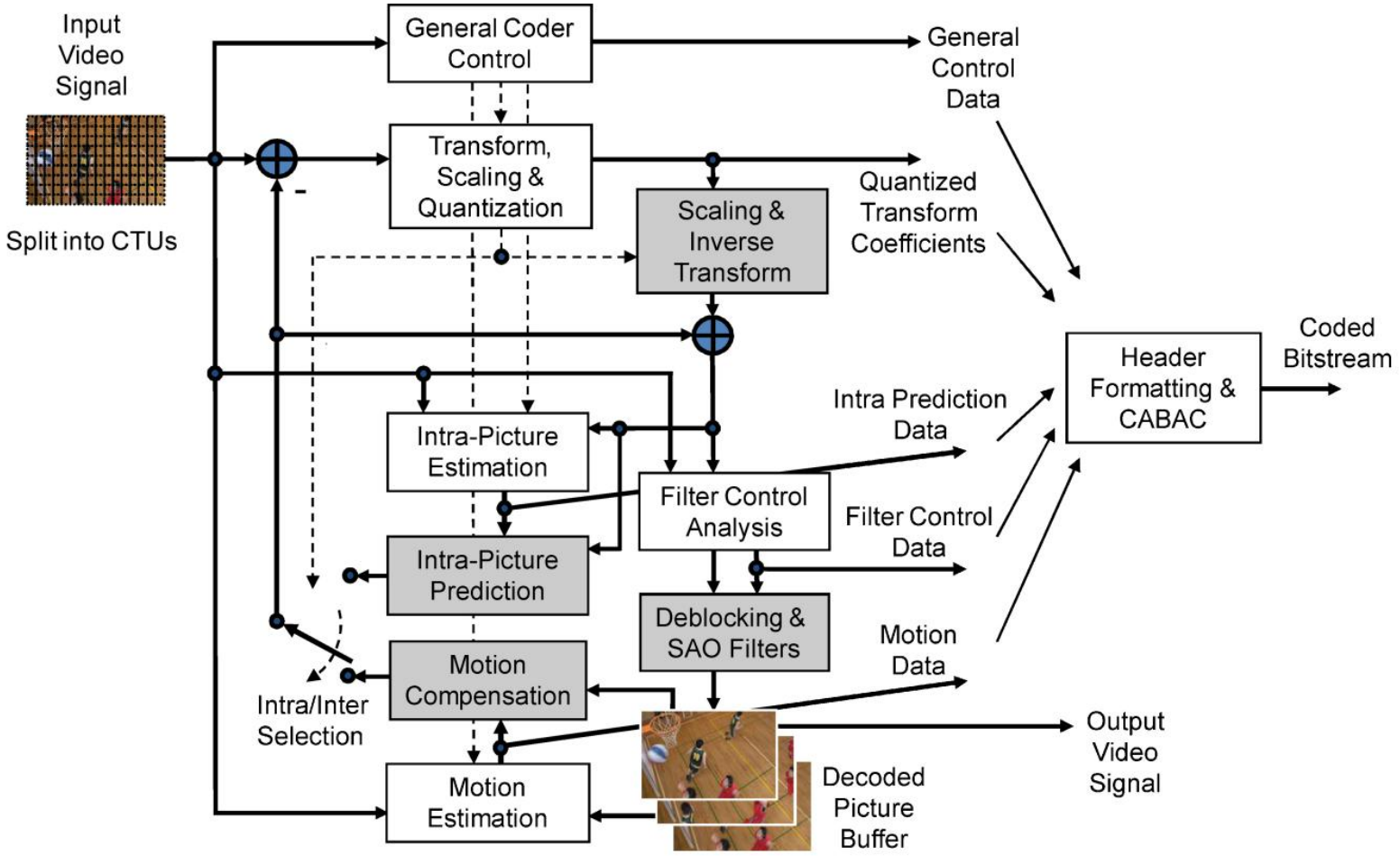
目录

H.265标准

In 2013, H.265/HEVC was released, aiming to reduce 50% bitrate

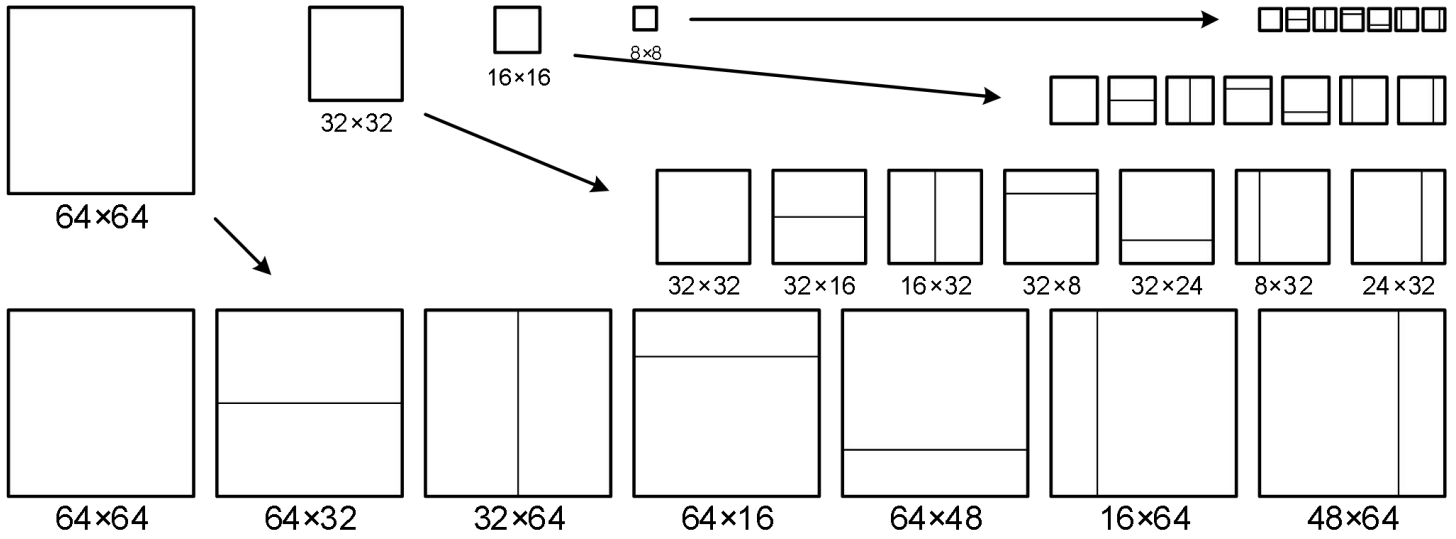


HEVC基本框架


Input Video Signal
Split into CTUs
General Coder Control
General Control Data
Transform, Scaling & Quantization
Quantized Transform Coefficients
Scaling & Inverse Transform
Header Formatting & CABAC
Coded Bitstream
Intra Prediction Data
Intra-Picture Estimation
Filter Control Analysis
Filter Control Data
Intra-Picture Prediction
Deblocking & SAO Filters
Motion Data
Motion Compensation
Intra/Inter Selection
Output Video Signal
Motion Estimation
Decoded Picture Buffer

设计难点

- 四叉树划分：模式多、划分复杂
  - 算法复杂，流水级设计困难
  - 访存带宽大，面积功耗大
  - 实时性要求高



设计目标

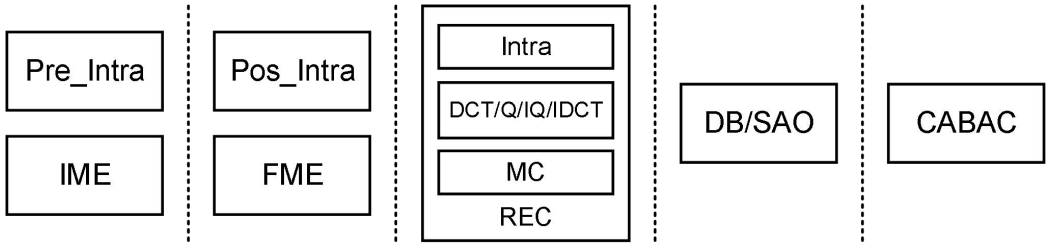
- 4K@30fps, 400 MHz, TSMC 65nm
- 一帧4K (3840x2160) 图像的LCU个数

$$3840 \times 2160 \div 64 \div 64 \approx 2025 \text{ LCU}$$

- 单个流水级的cycle数目

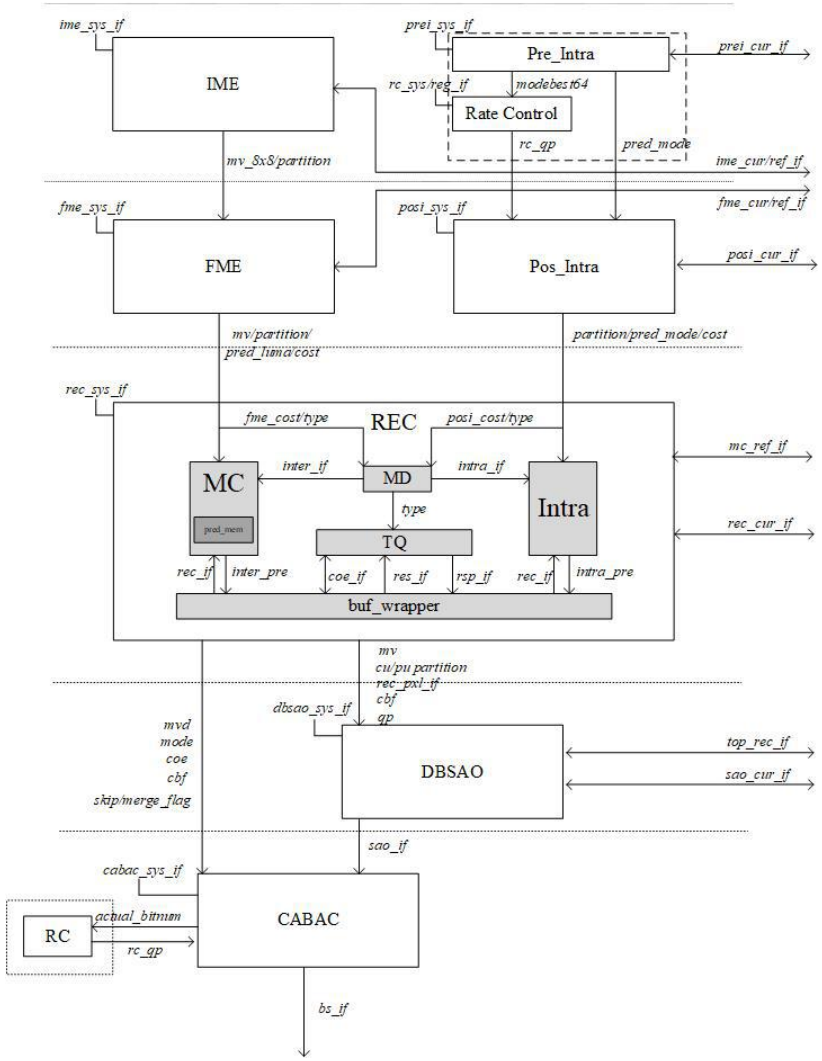
$$400 \times 10^6 \div 2025 \div 30 \approx 6584 \text{ cycle}$$

- 五级流水划分



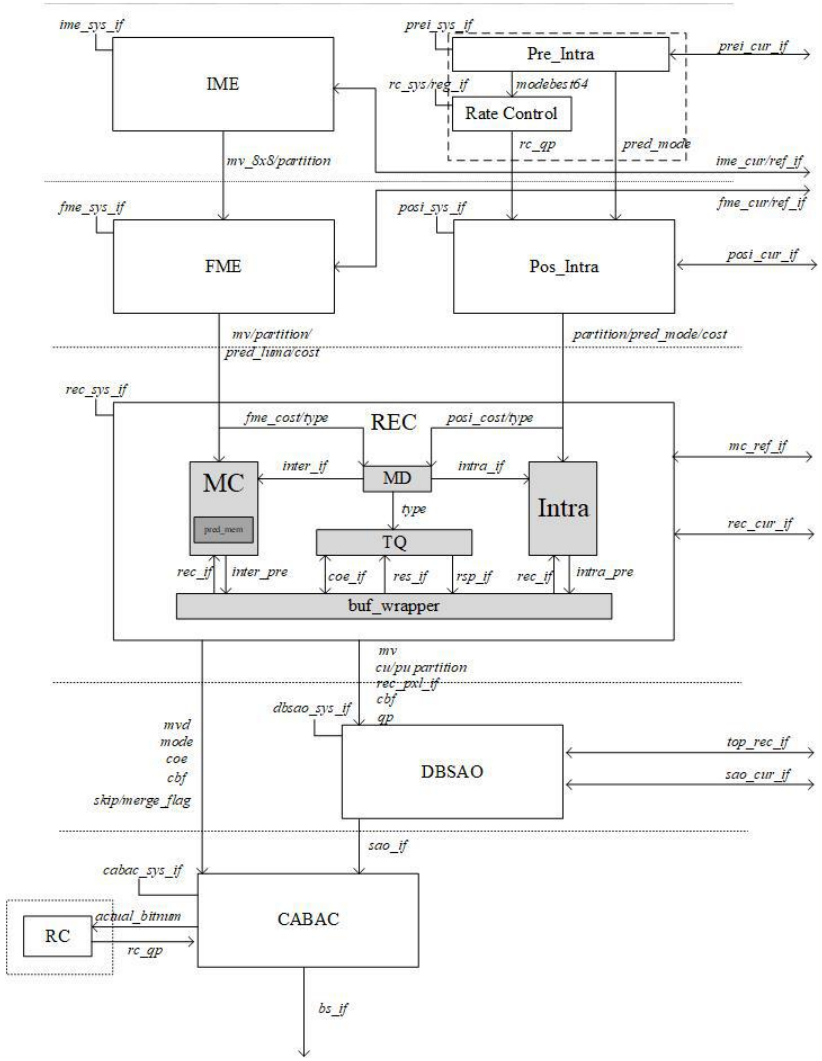
顶层架构

- Pre_Intra
  - 根据图像像素信息，对一个CTU内的各个大小的分块进行遍历，即依次对4×4，到8×8，16×16，32×32块，进行纹理分析，得到各个分块的预测模式
- Pos_Intra
  - 通过状态机控制对整个CTU中的块遍历，计算预测值与原始像素之间的失真，从下向上判断是否划分，从而在整个CTU中选择最优的块划信息



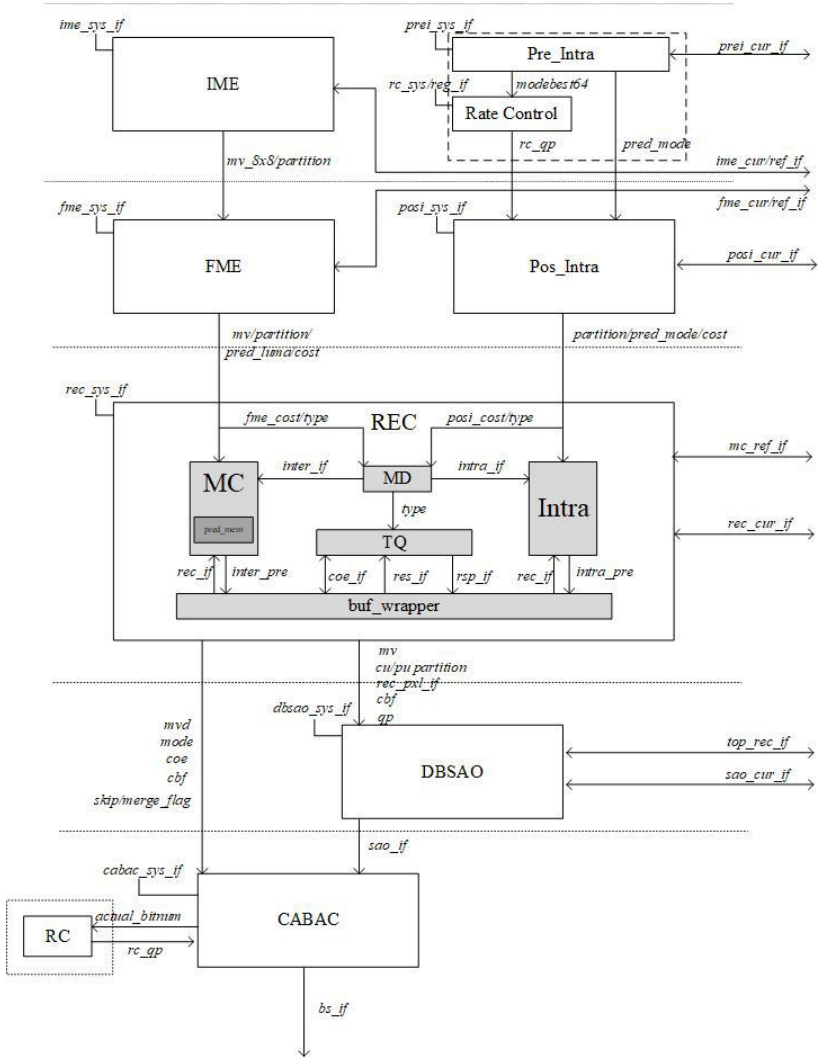
顶层架构

- IME
  - 整像素运动估计采用了一种可配置的搜索方法，通过软件配置搜索中心、搜索范围和搜索层次，从而提高应对不同场景的灵活性
- FME
  - 实际情况下，物体的运动是连续的，所以物体的偏移也不是整数像素的跳跃式运动。完成整个CTU的1/2像素搜索之后，在1/2像素运动向量的基础上进行1/4运动向量搜索，这样可以减少单次搜索次数，降低计算复杂度



顶层架构

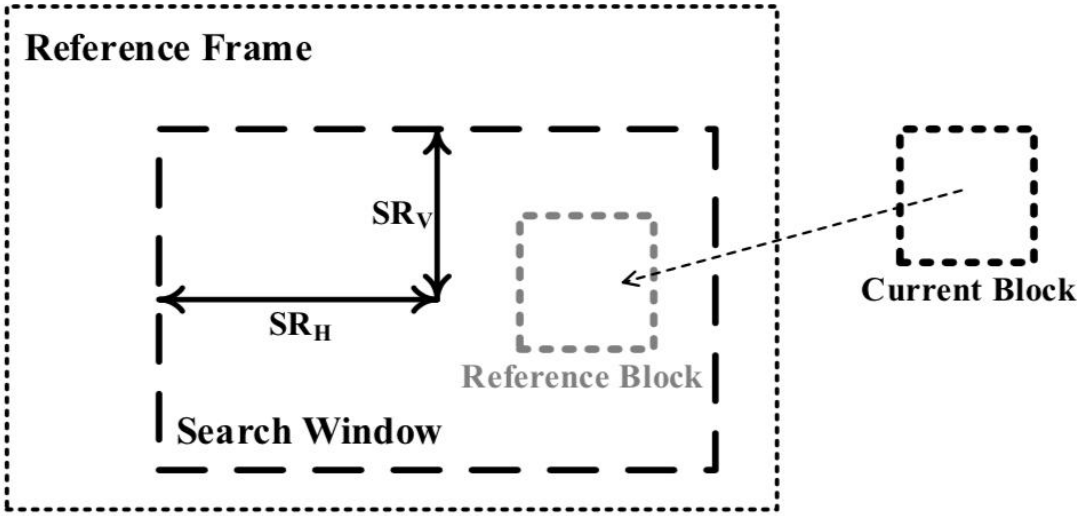
- REC
  - 根据帧内预测和帧间预测得到的模式和划分，预测像素与原始像素做差得到残差值，用于变化和量化，送至CABAC模块；并将经过反变换反量化之后得到的残差值与预测值相加得到重建像素值，输出到环路滤波模块
- DB/SAO
  - 环路滤波消除边缘的不连续，基于8×8块的串行处理，依次滤波整个CTU内的8×8块的垂直和水平边界
- CABAC
  - 由于熵编码计算过程较复杂，而且块与块之间存在严重的数据依赖，硬件处理速度较慢。为了加快熵编码处理速度，硬件实现中采用了4路并行的熵编码结构，每次能输出4 bit的码流数据



目录

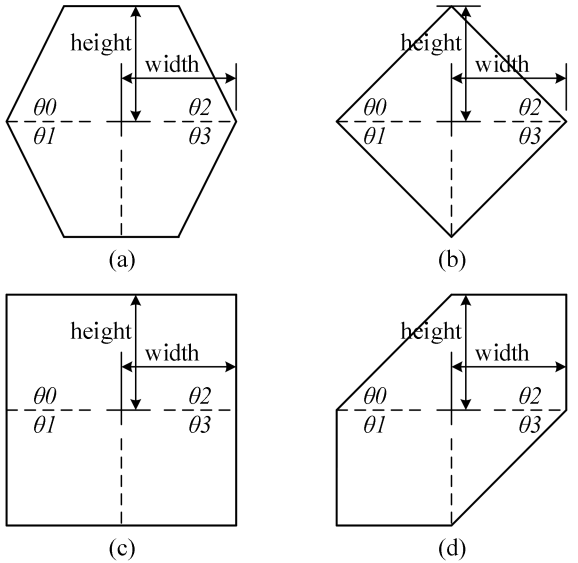
整像素运动估计 IME *[IEEE-T CSVT]*

- 对于当前帧的所有块，在参考帧中搜索最匹配的块
  - 参考窗大小
  - 搜索中心
  - 候选块搜索次序



整像素运动估计 IME

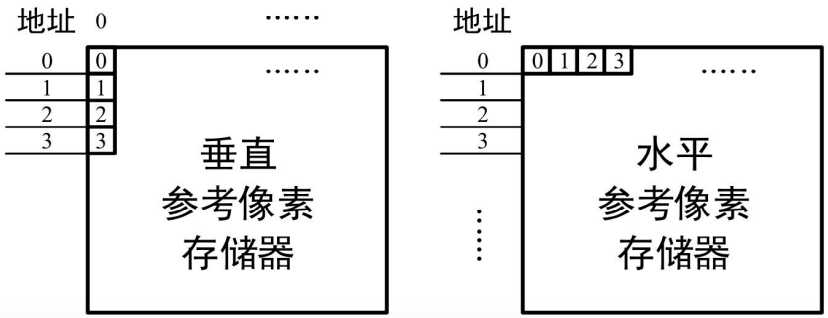
- 微代码可重构
  - 在帧间编码的过程中，需要确定搜索窗在参考帧中的位置和范围。在我们的硬件设计中，这两个参数以及一些搜索模式的参数均可以通过软件配置
    - 搜索窗中心位置
    - 搜索范围
    - 搜索窗形状
    - 搜索层级

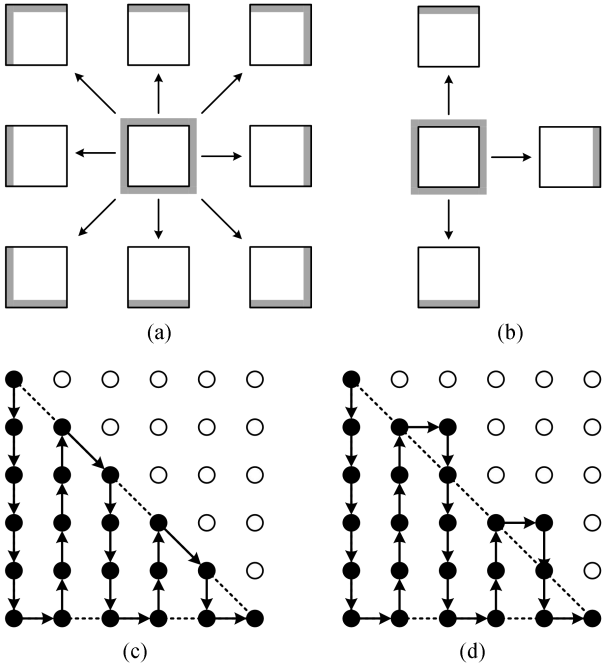


  - 冗余的搜索带来不必要的带宽和功耗，不够充分的搜索影响编码效果
    - 根据编码器的应用场景，调整运动估计的各个参数（高速公路监控 教室监控）
    - 软件统计之前若干帧的搜索结果，相应计算出下一帧的搜索位置和搜索范围，再传递给硬件编码器（无人机拍摄）

整像素运动估计 IME

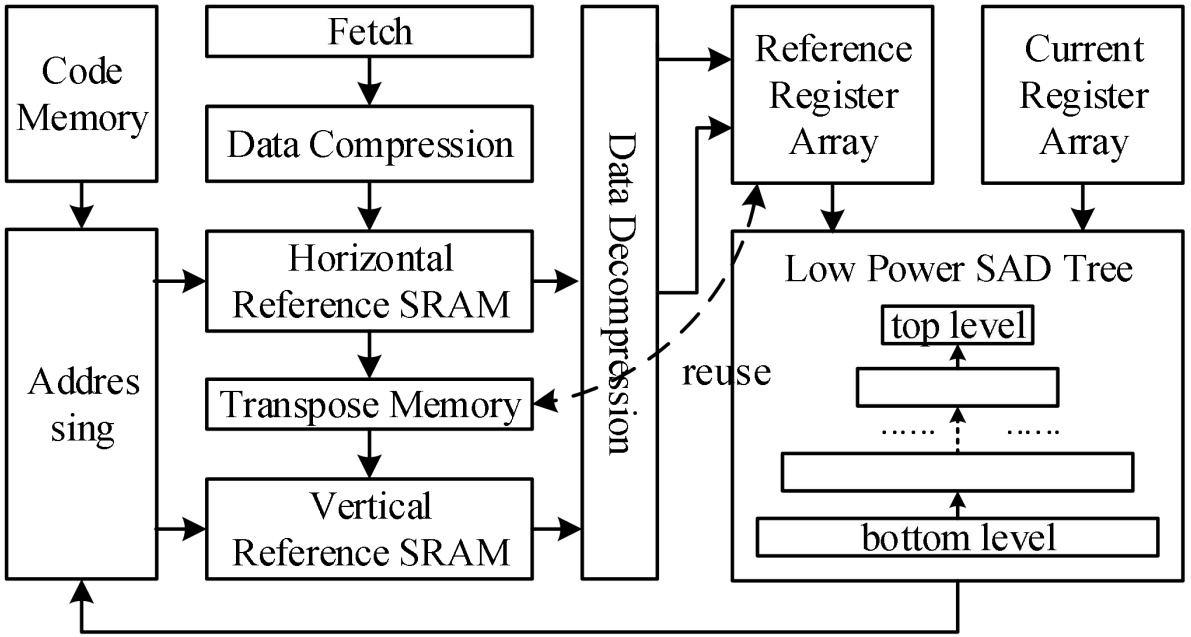
- 搜索点快速更新
  - H-V Reference SRAMs
    - Horizontal Reference SRAM
    - Veritical Reference SRAM
  - 在所有的运动方向，参考像素阵列的更新只需要一个周期





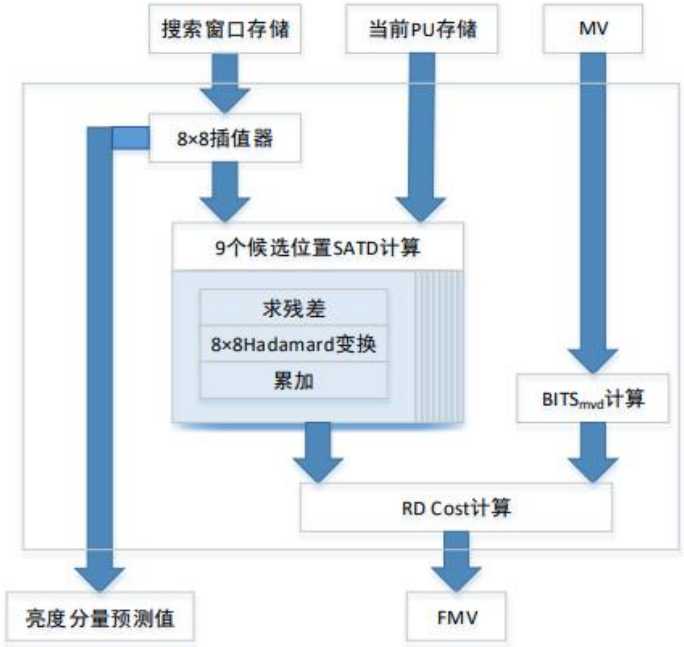
整像素运动估计 IME

- 硬件架构
  - Reference SRAM: 6.75 KB



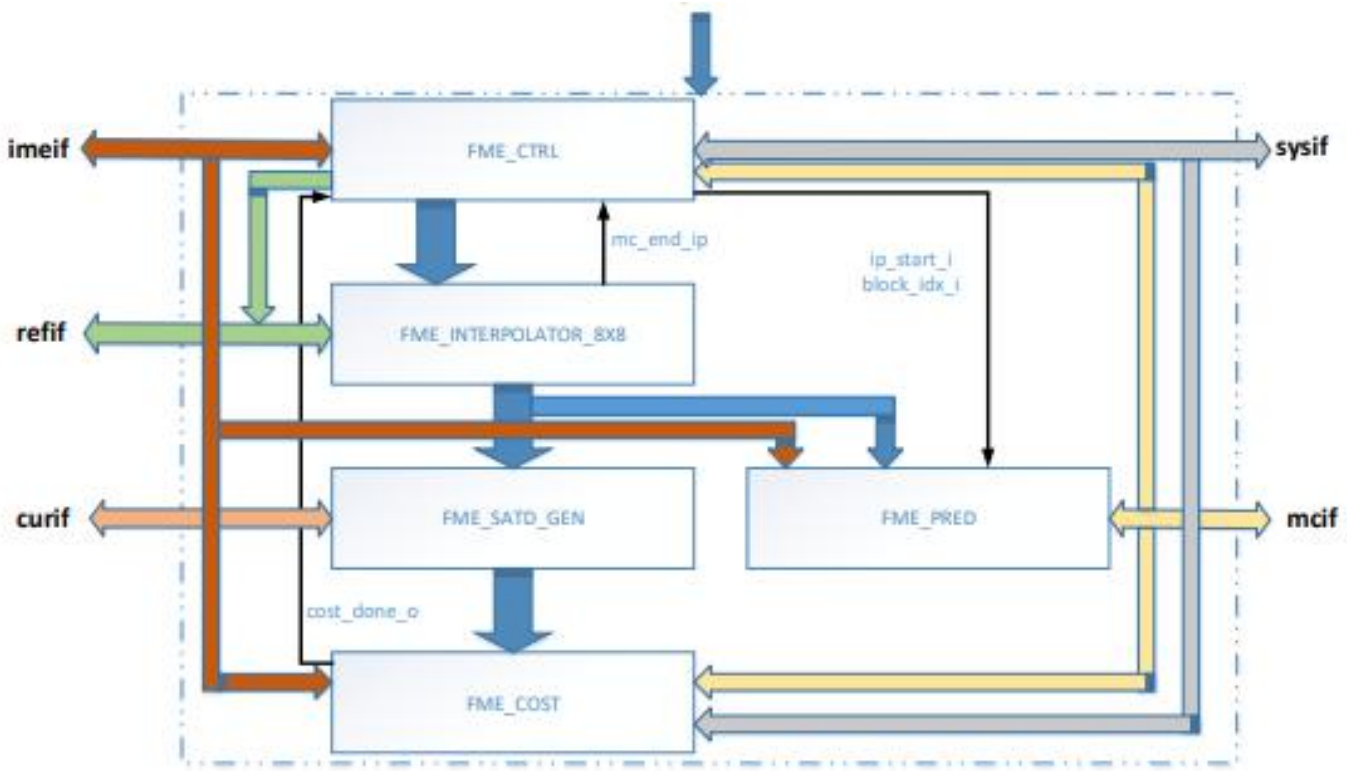
分像素运动估计 FME

- 硬件过程
  - 从存储器中获取重建像素值，以及原始像素值
  - 从上一级流水线IME获取到IMV的信息
  - 对划分后的若干个8×8单元进行亮度分量的插值计算并计算SATD值，并计算编码MV所需的比特数，求和得到RDCost
  - 选出最佳亚像素匹配块，得到对应的FMV值，以及对应的亮度分量预测值



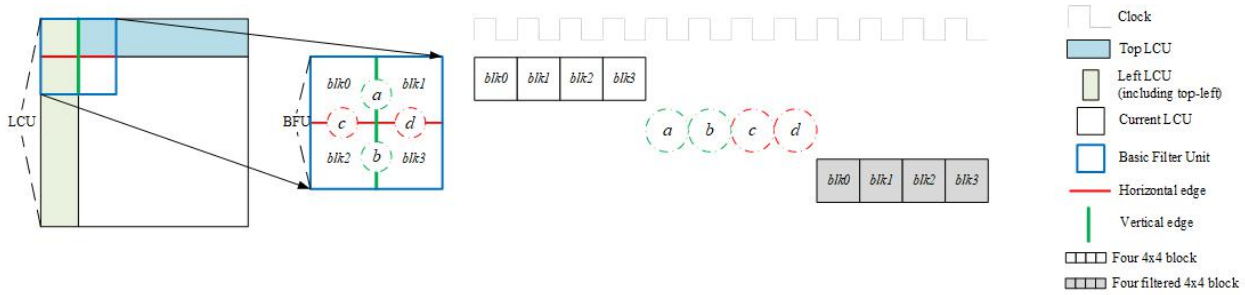
分像素运动估计 FME *[IEICE T]*

- 硬件架构

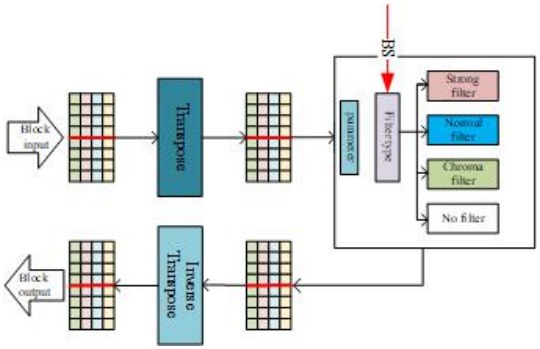


环路滤波 *[IEEE-T MM]*

- 硬件架构
  - 在HEVC中去方块滤波的处理中，HEVC的数据依赖只存在于8x8块内，因此，可以把8x8块大小作为DB的处理单元，这样能保证吞吐率的情况下，不需要片上SRAM，节省大量芯片面积
  - 每个周期处理一个4x4的块边界，处理整个LCU需要384个cycle

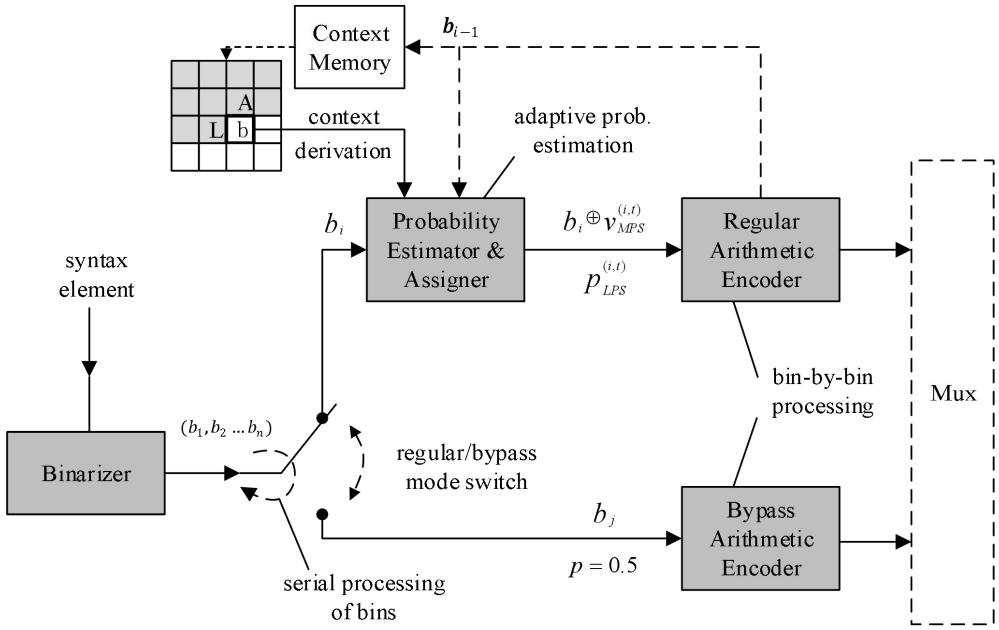


  - 滤波引擎：
    - strong filter、normal filter、chroma filter、no filter.



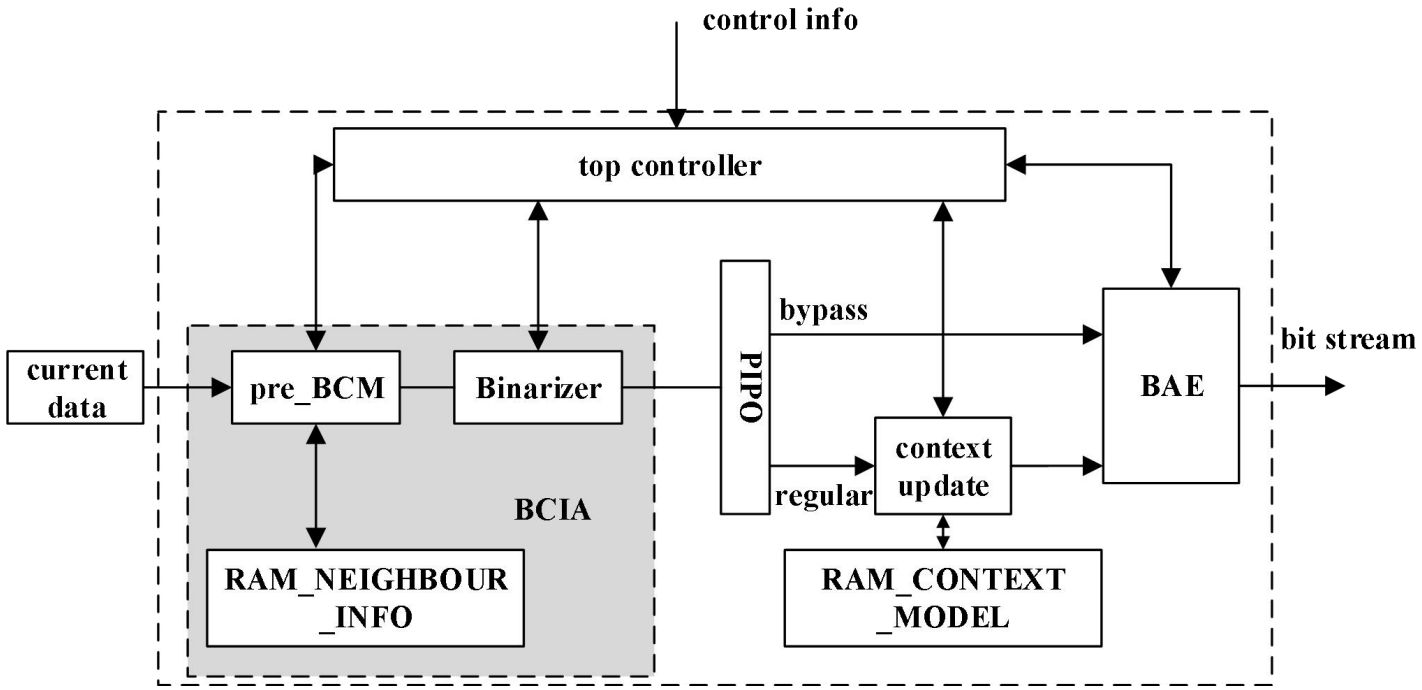
基于上下文的自适应二进制算术编码 CABAC

- 结构分析
  - 模块的输入是前级产生的数据如残差和运动向量差等，二值化模块将语法元素转换为一串二进制符号
  - 常规模式，通过上下文模型来自适应估计并更新二进制符号的概率
  - 旁路模式，不需要进行概率更新，BAE过程不需要进行区间的递归划分



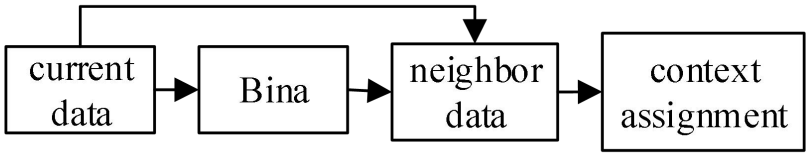
基于上下文的自适应二进制算术编码 CABAC

- 硬件结构
  - 完整的CABAC应该包括数据的获取，二值化和上下文索引的分配 (BCI)，上下文更新和BAE模块

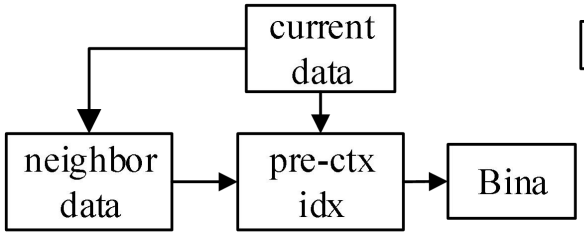


基于上下文的自适应二进制算术编码 CABAC

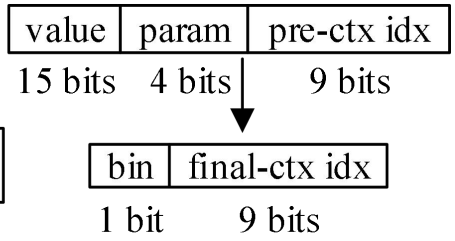
- BCIA模块的优化
  - HEVC标准中上下文索引的获得可以分为两个部分，一个是索引基准值的获取，第二部分是索引偏移量值的获取
  - 设计过程中可以将这两个过程分离，这样可以简化BCIA模块的时序，更进一步的，由于一部分二值化在索引基准值确定的过程中已经确定，因此二值化部分可以进行简化，从而减小并行化的消耗



(a)



(b)

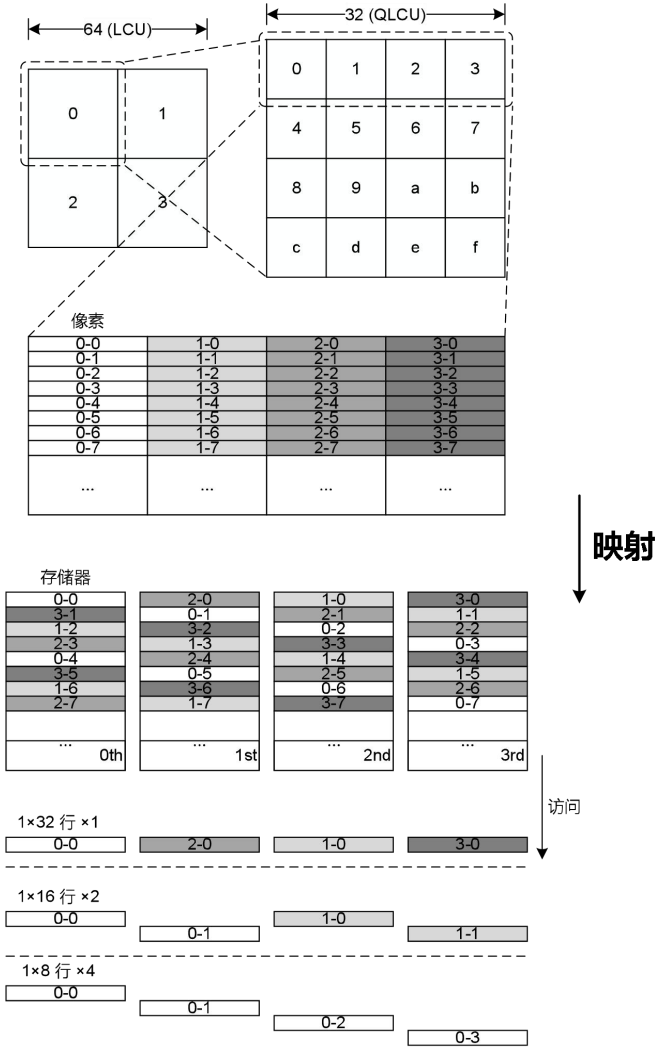


(c)

DCT变换 -- 并发存储器 *[IEEE-T VLSI]*

- 不同模块读取像素的方式不同
  - 预测模块：1×32，行格式
  - 像素读入：1×16，行格式
  - 去方块化、熵编码：4×4，块格式
- 并发存储器基于四个Bank实现
  - 每个QLCU被分割成16块8×8像素块(nBlock)
  - 每个8×8像素块被分割成8个1×8块(nRow)

```
switch(nRow%4)
    case 0: bank = (nBank + 0)%4
    case 1: bank = (nBank + 2)%4
    case 2: bank = (nBank + 1)%4
    case 3: bank = (nBank + 3)%4
end
addr = 32×nQLCU + 8×floor(nRow/4) + nBlock
```



并发存储器

- 所有左右相邻的4个1×8块都被映射到了不同的Bank中
  - 不管是1×32，1×16还是1×8都能够没有冲突地被访问
- 所有上下相邻的2个1×16块和相邻的4个1×8块也被映射到了不同的Bank中
  - 不管是2×16，还是4×8都能够没有冲突地被访问
- 光栅方式和并发方式下访问原始像素所需的周期

PU大小	4	8	16	32
光栅存储	4	8	16	32
并发存储	1	2	8	32

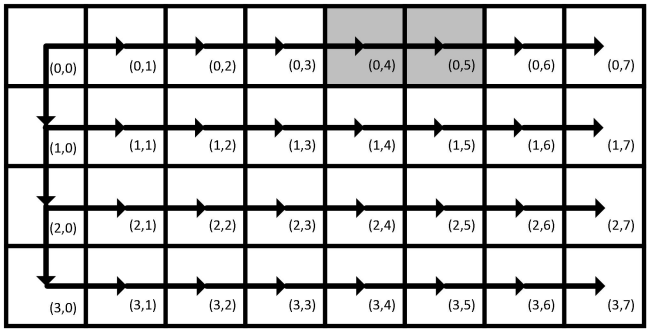
- 光栅方式和并发方式下存储原始像素所需的比特数和功耗

比较项目	比特数	功耗 (mW)	频率 (MHz)
光栅存储	32768	16.6	500
行列存储	32768	18.8	500

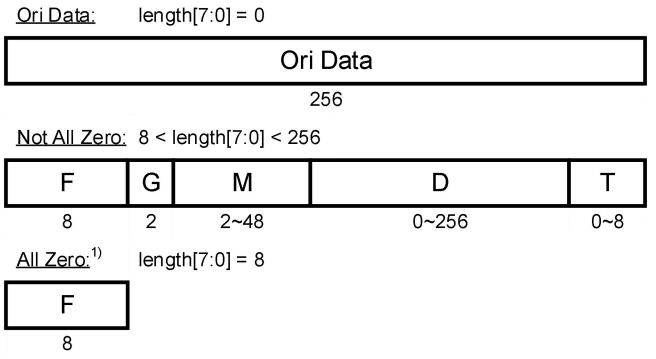
参考帧压缩 *[IEEE-T CSVT]*

- DPCM预测
  - 预测块的大小为8x4

$$Res_{m,n} = \begin{cases} P_{m,n} - P_{m-1,n} & (n = 0) \\ P_{m,n} - P_{m,n-1} & (n \neq 0) \end{cases}$$



- 压缩后的码流格式
  - 最终码流有三种格式：Ori data, Not All Zero, All zero
  - 解压缩时通过码流长度判读格式



参考帧压缩

- 小值优化的半定长编码

Table 1. Semi-fixed length coding

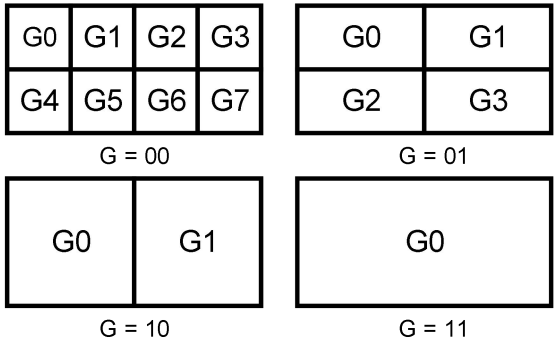
MAV[1)]		0	1	2	3~4	5~8	9~16	17~32	>=33
Coding Mode		0	1	2	3	4	5	6	7
D	0		0	00	000	0000	00000	000000	Original Pixel Values
	±1		1	S1	SS1	SSS1	SSSS1	SSSSS1	
	±2			10	S10	SS10	SSS10	SSSS10	
	±3				SS̲1	SSS̲1	SSSS̲1	SSSSS̲1	
	±4				100	S100	SS100	SSS100	
	±5					SS̲S1	SSS̲S1	SSSS̲S1	
	…								
	±8					1000	S1000	SS1000	
	…								
	±16						10000	S10000	
	…								
	±32							100000	

1) MAV: maximum absolute value.
2) "S" is the sign of residuals, "S̲" is the logic negation of residuals.

Table 2. Coding mode for residuals

Coding Mode	0	1	2	3
M	00	01	10	110
Coding Mode	4	5	6	7
M	1100	11110	111110	111111

Fig. 1. Regrouping modes



变换/反变换模块 *[IEEE-T CSII]*

- 变换和反变换矩阵都可以通过以下方式进行拆解

$$A_N = P_N \times \begin{bmatrix} A_{N/2} & 0 \\ 0 & R_{N/2} \end{bmatrix} \times B_N$$

- A_N代表N×N大小的正变换，P_N和B_N代表置换矩阵和蝶形变换矩阵

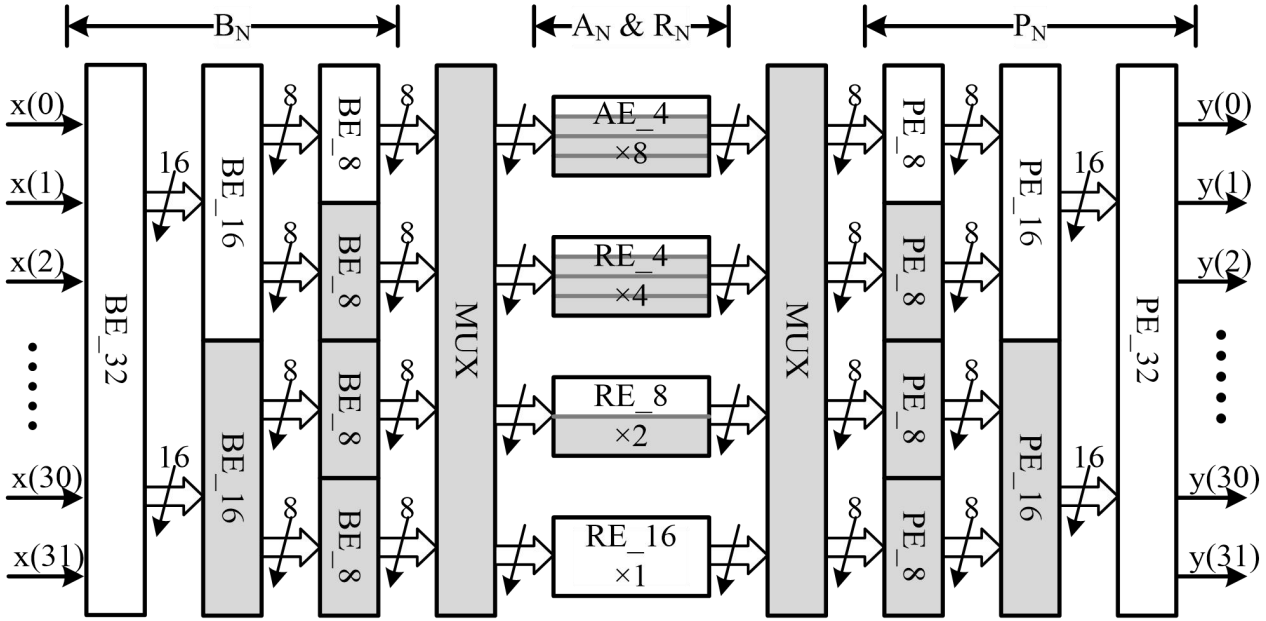
$$P_N(i,j) = \begin{cases} 1 & ; i = 2j \text{ or } i = 2j - N/2 \times 2 + 1 \\ 0 & ; else \end{cases}$$

$$B_N = \begin{bmatrix} I_{N/2} & \widetilde{I_{N/2}} \\ \widetilde{I_{N/2}} & -I_{N/2} \end{bmatrix}$$

- I_N和$\widetilde{I_N}$分别代表单位矩阵和反三角单位矩阵

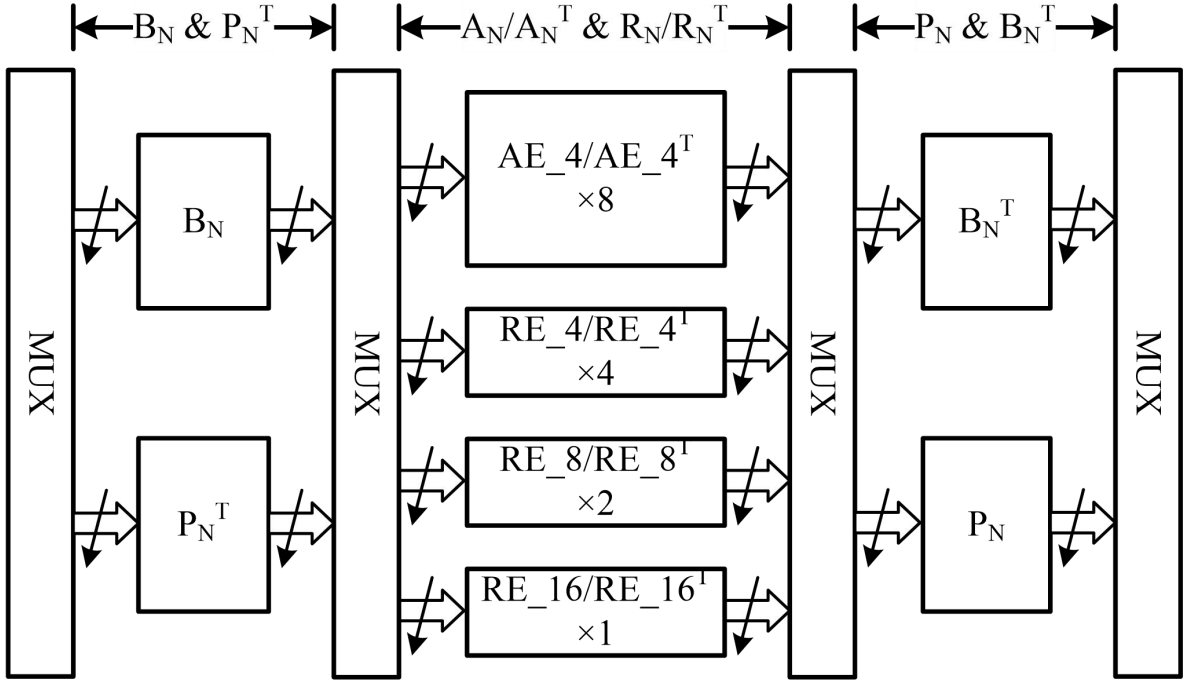
变换/反变换模块

- 对于不同大小的DCT运算，一维DCT复用硬件资源
  - BE_32、BE_16和BE_8分别代表B32、B16和B8的运算引擎
  - PE_32、PE_16、PE_8、RE_16、RE_8、RE_4和AE_4也代表了各自对应的运算引擎
- 恒定32像素每周期的1D-DCT结构



变换/反变换模块

- 一维IDCT复用一维DCT的部分硬件资源
  - 只复用了的A_N和R_N模块，这是因为A_N和R_N模块占用了较多的面积，复用产生的效益较大。除此之外，由于B_N和B_N^T分别处于正变换的第一级和反变换的最后一级，复用可能导致十分棘手的时序问题
- 恒定32像素每周期的1D-DCT/IDCT结构



目录

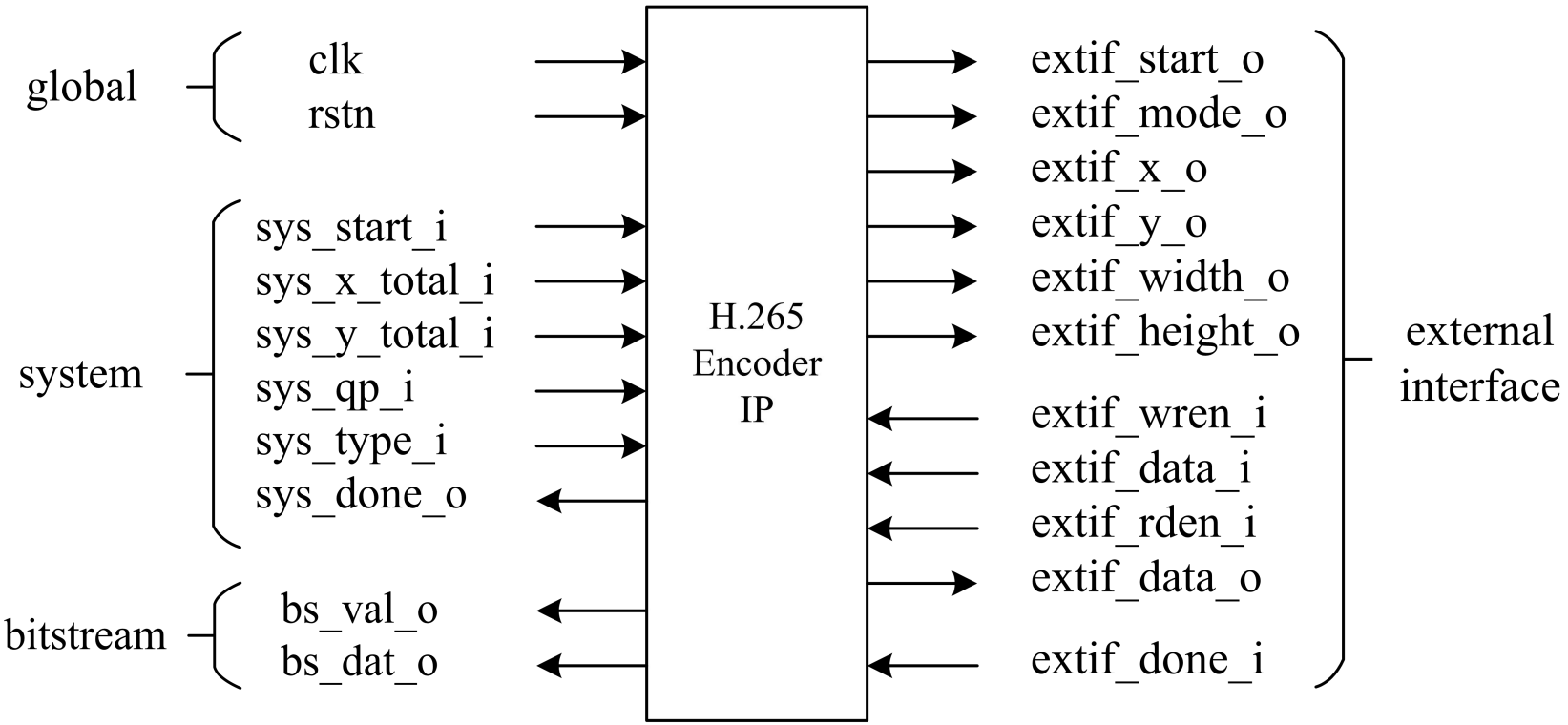
开源地址

http://openasic.org

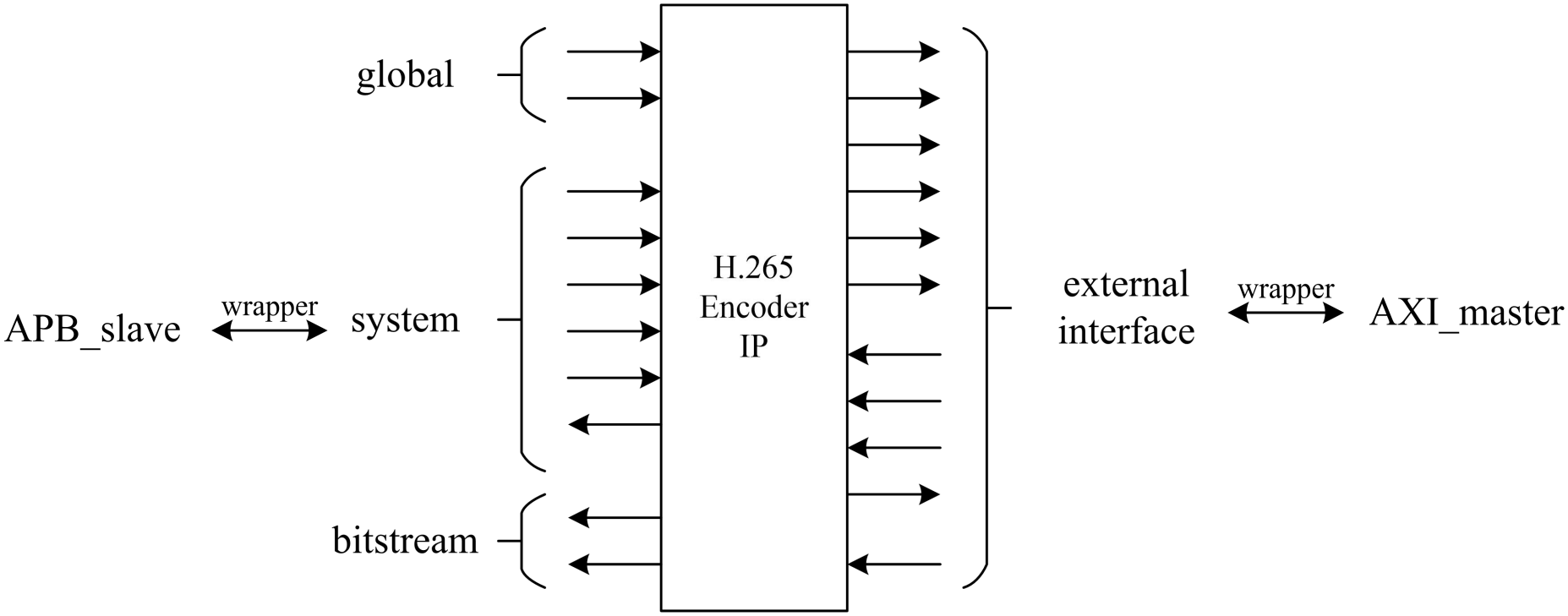
Feature List

- HEVC/H.265 Main Profile
- YUV 4:2:0
- Bit depth 8
- 4K@30fps, 400MHz
- GOP: I/P
- CTU: 64x64
- CU: 8x8~64x64
- PU: 4x4~64x64
- TU: 4x4/8x8/16x16/32x32
- 1/4 Sub-pixel interpolation
- Search range: 64
- All 35 Intra prediction mode
- CABAC
- Deblocking Filter
- SAO（Sample Adaptive Offset）
- Rate control: CBR/VBR（Software-based)
- Rate control: CTU level（Hardware-based)
- SKIP/MERGE
- Intra CTU in Inter frame

Interface


global
clk
rstn
system
sys_start_i
sys_x_total_i
sys_y_total_i
sys_qp_i
sys_type_i
sys_done_o
bitstream
bs_val_o
bs_dat_o
H.265
Encoder
IP
extif_start_o
extif_mode_o
extif_x_o
extif_y_o
extif_width_o
extif_height_o
extif_wren_i
extif_data_i
extif_rden_i
extif_data_o
extif_done_i
external
interface

Integration



综合结果

- DC综合
  - TSMC 65nm, 400 MHz

Module	Logic Gate Count
enc_top	**2064K**
ime	313K
fme	325K
prei	71K
posi	138K
rec	764K
dbsao	198K
cabac	125K
fetch	103K

综合结果

- FPGA综合
  - Xilinx xvcu9pflgb2104-2L，100MHz

Module	LUT	LUTRAM	FF	BRAM	DSP
enc_top	**329559**	**1080**	**116220**	**2149.5**	**725**
ime	40973	51	25707	256	0
fme	34746	19	17235	259.5	225
prei	3681	63	5300	0	45
posi	18270	40	7511	2	138
rec	133752	837	31437	1091.5	316
dbsao	21225	56	17167	1	1
cabac	25568	14	6581	0	0
fetch	42220	0	2099	539.5	0

文件目录

- **h265enc_v2.0** - main folder
  - **lib** - behavioral memory model
  - **rtl** - rtl for the h265 encoder
    - **cabac** – rtl for the cabac
    - **db** – rtl for the DB and SAO
    - **fetch** – rtl for the data exchange with the external memory
    - **fme** – rtl for the fractional motion estimation
    - **ime** – rtl for the integer motion estimation
    - **mem** – memory used in the h265 encoder
    - **posi** – rtl for the pos-intra, CU partition decision
    - **prei** – rtl for the pre-intra, CU mode decision
    - **rec** – rtl for the reconstruction loop, DCT/Q/IQ/IDCT
    - **top** – rtl for the h265 encoder top
    - **enc_defines.v** – defines for the h265 encoder
  - 转下页

文件目录

- **h265enc_v2.0** - main folder
  - 接上页
  - **sim** - simulation files for the h265 encoder
    - **top_testbench** - TV and TB for the h265 encoder top
  - **sw** - software reference model for the simulation
    - **f265** – linux version
    - **f265.exe** – windows version
    - **f265_encode.cfg** – config file for the software reference model

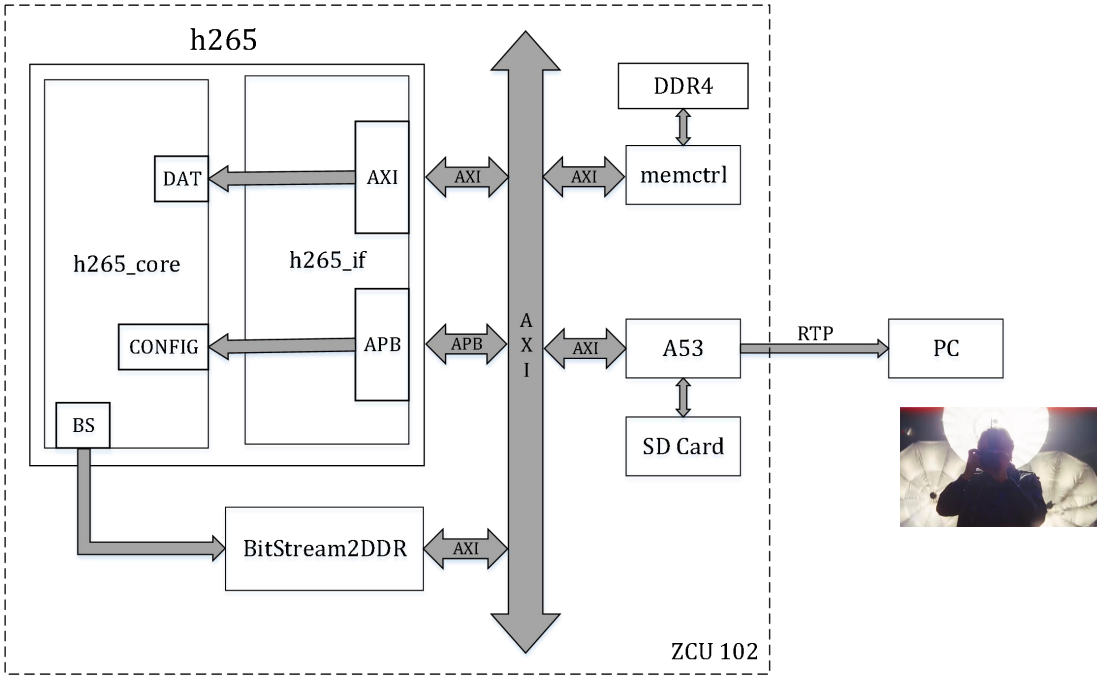
仿真教程

- 修改"f265_encoder.cfg"文件里的各项参数
  - 测试序列位置，图像尺寸，QP，GOP
- 运行 "f265 -c f265_encode.cfg"
  - 生成测试文件
- 复制测试文件到"sim/top_testbench/tv "
- 修改"tb_enc_top.v "的各项参数
  - 确保各项参数与" f265_encoder.cfg "一致
- 在"sim/top_testbench"文件夹运行"make sim"启动仿真
  - 如果仿真结果与测试文件不一致，报错并中断仿真

目录

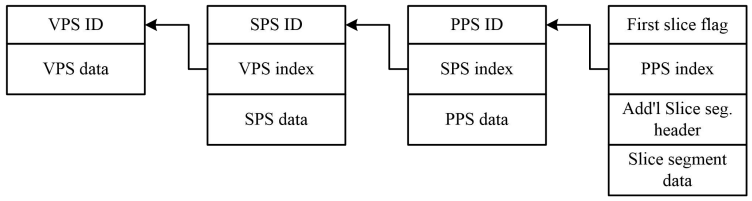
平台架构

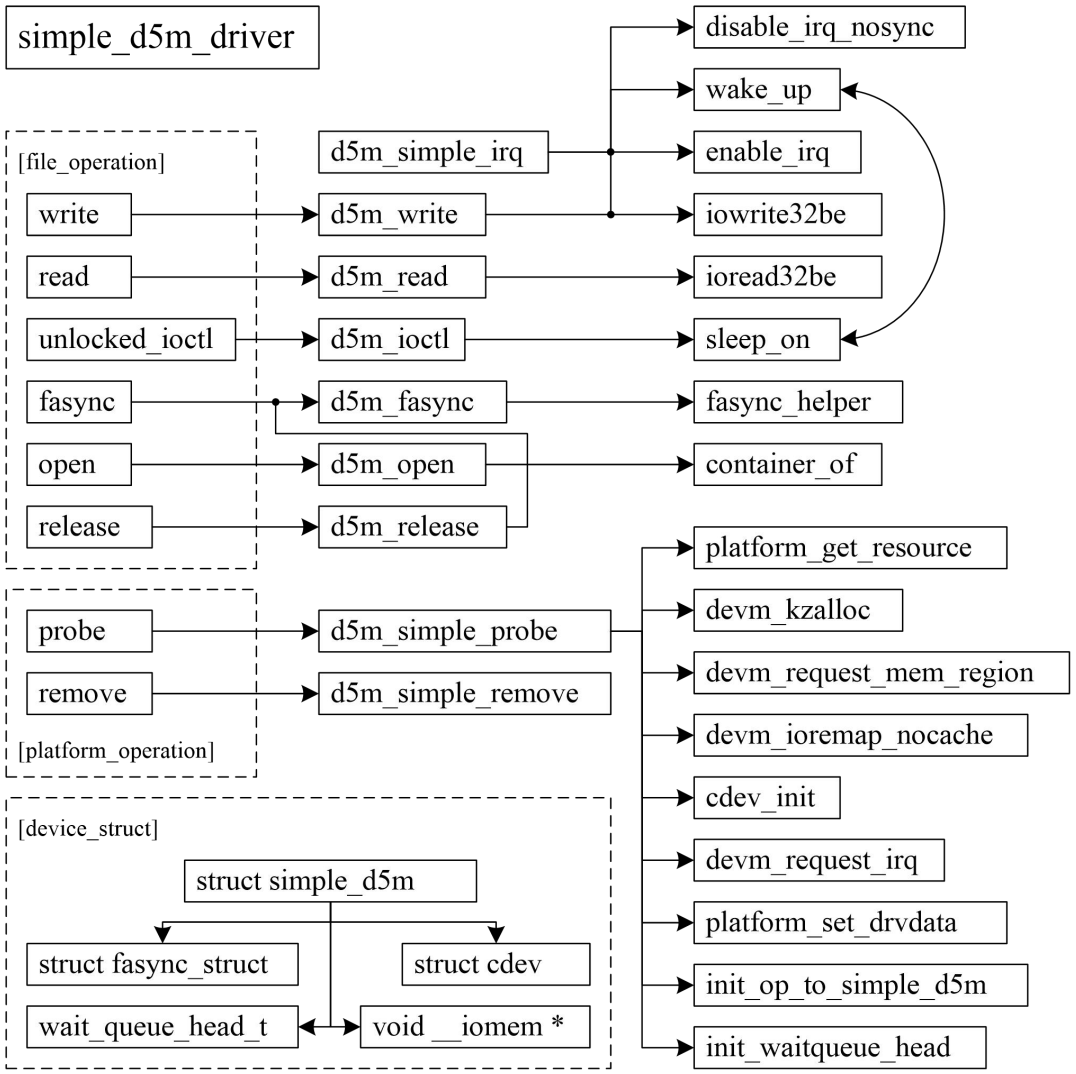
- CPU将视频序列从SD卡读出，写入内存
- HEVC编码器从内存读取视频序列，编码后将码流写入内存
- CPU从内存中读取码流，进行RTP打包并通过以太网发送
- PC接收网络包并解码播放



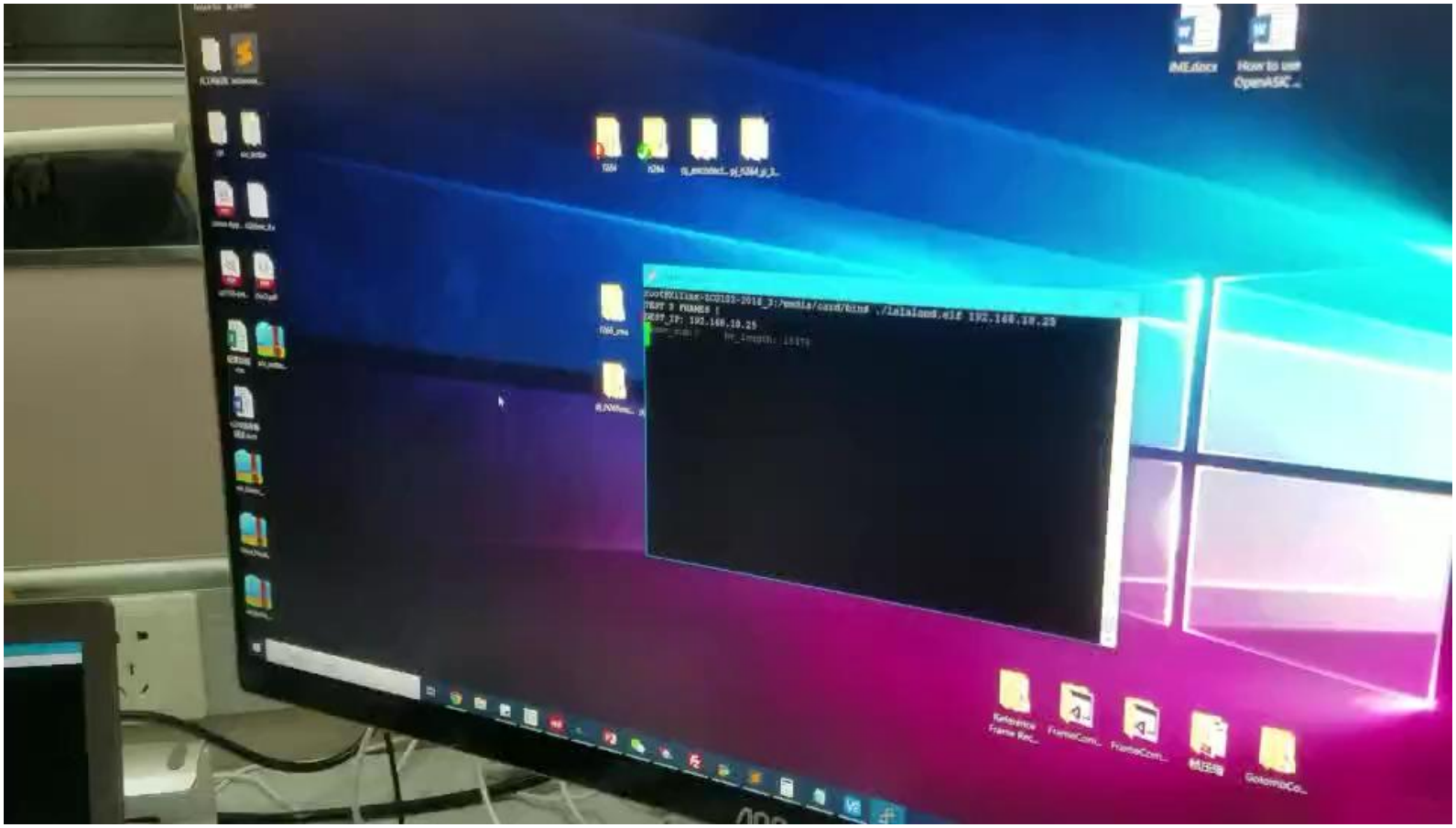
软件驱动

- Linux移植
- H265驱动软件
- 码率控制
- 摄像头驱动软件
- RTSP推流
- APP程序





效果演示



目录

工作进展和展望

- 编码性能优化
  - BD-Rate
  - vs Verisilion, Chips&Media
- 开发基于FPGA的HEVC视频编码器
  - 4K@30fps 200MHz
- 开发面向下一代的VVC视频编码器
  - Maximum Size: 128*128
  - Quadtree with binary tree and ternary tree
  - …
- 开发基于OpenPOWER的异构计算8K x265视频编码器

THANK YOU

范益波