An Implementation of Multiple-Standard Video Decoder on a Mixed-Grained Reconfigurable Computing Platform*

Leibo LIU†, Member, Dong WANG†††, Yingjie CHEN†, Min ZHU†, Nonmembers, Shouyi YIN†, Member, and Shaojun WEI†, Nonmember

SUMMARY This paper presents the design of a multiple-standard 1080 high definition (HD) video decoder on a mixed-grained reconfigurable computing platform integrating coarse-grained reconfigurable processing units (RPUs) and FPGAs. The proposed RPU, including 16 × 16 multi-functional processing elements (PES), is used to accelerate compute-intensive tasks in the video decoding. A soft-core-based microprocessor array is implemented on the FPGA and adopted to speed-up the dynamic reconfiguration of the RPU. Furthermore, a mail-box-based communication scheme is utilized to improve the communication efficiency between RPUs and FPGAs. By exploiting dynamic reconfiguration of the RPUs and static reconfiguration of the FPGAs, the proposed platform achieves scalable performances and cost trade-offs to support a variety of video coding standards, including MPEG-2, AVS, H.264, and HEVC. The measured results show that the proposed platform can support H.264 1080 HD video streams at up to 57 frames per second (fps) and HEVC 1080 HD video streams at up to 52fps under 250MHz, at the same time, it achieves a 3.6× performance gain over an industrial coarse-grained reconfigurable processor for H.264 decoding, and a 6.43× performance boost over a general purpose processor based implementation for HEVC decoding.

key words: algorithm mapping, coarse-grained reconfigurable array, field-programmable gate array, reconfigurable computing, video decoding

1. Introduction

Reconfigurable computing fabrics fall in between the instruction driven processor, such as general purpose processors (GPPs) and digital signal processors (DSPs), and hardware logic, such as application specific integrated circuits (ASICs). When dealing with word-level compute-intensive applications, coarse-grained reconfigurable computing fabrics have substantial advantages in performance and power over instruction driven processors, while still possessing a very high function flexibility compared with the hard-wired logic [1]. In contrast, fine-grained reconfigurable devices have more advantages in bit-level control-intensive computations. Most FPGAs provide low-level fine-grained parallelism with a high degree of flexibility but normally pay a power or area penalty. Concerning the capability of dynamic reconfiguration, although much research, such as dynamic partial reconfiguration [2], has been conducted in the past few years to improve the speed of reconfiguration process on FPGAs, it still could not achieve the same level of efficiency when compared with the coarse-grained reconfigurable processors.

The designs of the coarse-grained reconfigurable processor have been widely studied in the literature during the last decade. A dynamically reconfigurable processor named XPP-III, reported by Ganesan et al. [3] and Rossi et al. [4], is an example of high-performance commercial reconfigurable multimedia processor. It consists of 56 processing elements (PEs) and 8 very long instruction word (VLIW) RISC cores. The XPP-III architecture utilizes a fast configuration mechanism, which can dynamically switch preloaded configuration contexts in only one clock cycle. The implemented chip is capable of decoding 1080p main profile H.264 video streams at 24 frames per second (fps). ADRES, which combines a VLIW processor with a reconfigurable array of 8 × 8 function units, is presented by Suzuki et al. [5]. In this design, the reconfigurable array is used to accelerate the dataflow-like kernels, whereas the VLIW core executes the non-kernel code by exploiting instruction-level parallelism.

A common feature of these designs is that the coarse-grained reconfigurable processors require a general-purpose-like processor, such as RISC or VLIW, to implement the control tasks in the software. However, after the system is fabricated, the performance and power characteristics of such general-purpose-like processors are usually fixed and are not scalable due to the untailorable hardware architecture. A better scheme is to utilize the advantages of both fine-grained and coarse-grained reconfigurable hardware to build a mixed-grained reconfigurable computing platform. For instance, Sterpone [6] proposed an analytical model for analyzing the tradeoff between fine-grained processing tasks and coarse-grained tasks that should be implemented on different hardware architectures depending on the granularity of the architecture. The reconfigurable hardware was composed by mixed-grain reconfigurable cells that include 64-bit arithmetic logic units (ALUs), look-up tables (LUTs), and an efficient configuration and data memory architecture [6]. Experiments show that the mixed-grained processor was very scalable and it could efficiently extract the parallelism from streamed applications. A new reconfigurable cell (RC) based mixed-grain architecture [7] was proposed. This architecture could deliver a gate-level imple-
mentation of the reconfigurable logic unit (RLU) focusing on the ALU implementation. There are still several issues that have not been fully studied by previous works, i.e., the communication and parallelization schemes that should be adopted when mapping the target application on two types of hardware architectures with different granularities. Moreover, no actual applications have ever been implemented on such a mixed-grained system to prove the efficiency of these proposed optimization schemes.

In this paper, we focus on designing a mixed-grained reconfigurable platform for multiple-standard video decoding. The main contributions of this work include: (1) proposing an energy-efficient VLSI array architecture of a coarse-grained reconfigurable processing unit (RPU) targeting compute-intensive multiple-standard video decoding; (2) proposing an efficient implementation of a soft-core-based microprocessor array on FPGA to speed-up the dynamic reconfiguration of the proposed RPU; (3) presenting a mail-box-based communication scheme that improves the communication efficiency between RPUs and FPGAs; (4) a multi-level macroblock (MB)-based parallel and block-based parallel techniques for mapping compute-intensive tasks on multiple RPUs. A reconfigurable computing platform, which integrates eight RPU processors and two FPGA fabrics, is implemented and tested with multiple video decoding applications including H.264, MPEG-2, AVS [8] and HEVC [9]. The experimental results show that the proposed flexible architecture can achieve considerable speed-ups for a wide range of applications.

2. Algorithm Analysis

The proposed reconfigurable computing platform targets multiple-standard video decoding which is normally regarded as one of the most challenging applications because video decoding consists of wide-ranging computation/processing sub-tasks with different characteristics. The decoding tasks can be divided into several categories of arithmetic computations and tasks, i.e., bit-level computations, compute-intensive data-parallel calculations, massive data accessing/storing operations, and word-level complex arithmetical operations. For these word-level compute-intensive tasks, the target implementation platform is the coarse-grained reconfigurable computing hardwires, while for those bit-level and control-intensive tasks, FPGA fabric is the target platform.

In this paper, three widely used standards (i.e., H.264, MPEG-2, and AVS) and the new standard, i.e., HEVC, are considered. Although these standards vary in many aspects because of the different targeted application scenarios, most of them could be generally summarized by a unified framework [9], [10]. All these video standards are based on a block-based hybrid coding architecture as depicted in Fig. 1. Entropy decoding (ED) kernel parses and decodes syntax elements. The decoded syntax elements are sent to different functional kernels in the decoding process. Inverse quantization (IQ) kernel carries out the scaling process of transform coefficients and then inverse transform (IT) kernel executes transformation process of scaled transform coefficients. At the same time, motion compensation (MC)/intra prediction (IP) kernel executes inter/intra-prediction calculation. Deblocking filter (DF) is used to reduce blocking artifacts resulted from the block-based processing, while sample adaptive offset (SAO) kernel is adopted to reduce the mean distortion between original and reconstructed samples by adding an offset to samples. In these kernels, ED is dominated by irregular non-arithmetic operations such as look-up table operations and bit-manipulation, and the inherent data dependence among sequentially decoded syntax elements makes ED very difficult to adopt parallel structure; whereas IQ, IT, MC, IP, DF, or SAO is dominated by arithmetic operations (e.g., addition, subtraction, multiplication, etc.) and is block-based word-level regular calculation processing a large number of data in a relatively uniform way. In a word, in a video decoding process, ED is thought as a control-intensive kernel, whereas IQ, IT, MC, IP, DF, and SAO are treated as compute-intensive kernels.

Other studies, such as parallelization of compute-intensive tasks in H.264 based on reconfigurable multimedia system [11], have also shown that the block-based word-level calculations comprise a majority of the decoding workloads in all the video decoding standards. For instance, an experiment of H.264 decoding is performed on a RISC processor to measure the workloads of MC, DF, and inverse quantization and transformation (IQT). The results show that the workloads account for more than 75% of the total workload. Moreover, during the execution of these sub-algorithms, the coefficients and the pixels are processed in parallel within each MB or sub-MB. And in MC and IDCT, the processing of each MB is also data-independent. MPEG-2, AVS, and even HEVC also show similar computational patterns.

Therefore, as shown in Fig. 1, in the proposed platform, coarse-grained reconfigurable processors are designed...
to support the execution of the sub-algorithms including IQ, IT, IP, MC, DF and SAO (for HEVC only) which are shown in green boxes, while the remaining ED tasks, which are shown in gray boxes, are mapped onto FPGAs. In the following sections, we will first focus on how to design an efficient coarse-grained reconfigurable hardware architecture which could efficiently implement the heavy-work-loaded block-based word-level decoding sub-tasks. Then the optimal communication scheme between coarse-grained processor and FPGAs which greatly improves the overall system parallelism and performances is proposed. Finally the optimization schemes which target mapping the control-intensive kernels on fined-grained FPGAs and mapping the compute-intensive kernels on coarse-grained processors are discussed.

3. The Proposed Architecture

The proposed mixed-grained reconfigurable computing platform is shown in Fig. 2. It consists of a main FPGA as the major controlling engine, several coarse-grained reconfigurable processors and a small FPGA which connects all the processors as a router. Our work focuses on the utilization of the main FPGA. In the following sections, we will introduce separately the design of the coarse-grained processor and the implementation of an efficient soft-core implemented on the main FPGA that controls the whole system and the dynamic reconfiguration of the coarse-grained processors.

3.1 Coarse-Grained Reconfigurable Processor

The proposed architecture of the coarse-grained processor, which is referred as RPU, is shown in Fig. 3. It consists of five major parts: the PE array (PEA), the Input Control logic, the Output Control logic, the Inner Buffering memory, and the Context Storing and Controlling Logic. The former four parts make up of the data processing path, and the latter one forms the configuration path. The operation of the RPU is driven by the input context. When the context streams are pushed into the context storage module, the controlling logic first translates and reorganizes the contexts, and then configures the input control module or the inner buffering module to prepare the data (fetching from external memory or loading from inner buffering memory). At the same time, the PEA is reconfigured to the desired functional structure. Once the data are prepared, the data stream is sent to the PEA and processed. In the end, the calculated results will be exported from PEA to the output control module (to write back to external memory) or inner buffer module according to the functions configured by the contexts.

The reconfigurable PEA is organized in four groups referred as the reconfigurable cell array (RCA), and each RCA contains $8 \times 8$ reconfigurable PEs as illustrated in Fig. 4 (a). Functions of an RCA change with the context switch. Each RCA can work independently thus providing a higher level of parallelism to improve the RPU throughput. Moreover, any RCA can also be turned off by switching off its clock to maintain very low power consumption when desired. Each PE with 16-bit data-width contains an ALU, a group of input (A\_REG and B\_REG), output (C\_REG), temporary result (T\_REG) registers, and a route block, as illustrated in Fig. 4 (b). The functionality of each PE is summarized in Table 1. There are 26 different operators, which can perform logical operations, such as NOT, AND, OR, XOR, etc., and arithmetic operations, such as addition, multiplication, shift, comparison, saturation, absolute, etc.
### Table 1 Function summary of the ALU.

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>No.</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A×B</td>
<td>14</td>
<td>A×E(0,B)</td>
</tr>
<tr>
<td>1</td>
<td>A×B</td>
<td>15</td>
<td>A×E(∞,B)</td>
</tr>
<tr>
<td>2</td>
<td>A×B(3,0)</td>
<td>16</td>
<td>A×B</td>
</tr>
<tr>
<td>3</td>
<td>A×B[1,0]</td>
<td>17</td>
<td>(A×B)ₙ₉₆₅₁₆</td>
</tr>
<tr>
<td>4</td>
<td>(A×(1&lt;&lt;B)&gt;&gt;1))×B</td>
<td>18</td>
<td>(A×B)ₙ₉₆₅₁₆</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>19</td>
<td>B×A</td>
</tr>
<tr>
<td>6</td>
<td>A×B</td>
<td>20</td>
<td>B×A×0?</td>
</tr>
<tr>
<td>7</td>
<td>A×B</td>
<td>21</td>
<td>B×A×0?</td>
</tr>
<tr>
<td>8</td>
<td>A×B</td>
<td>22</td>
<td>T=1?×B×A×B\text{-}A</td>
</tr>
<tr>
<td>9</td>
<td>¬(A×B)</td>
<td>23</td>
<td>A×E(B, x)</td>
</tr>
<tr>
<td>10</td>
<td>[A×B]</td>
<td>24</td>
<td>(A×B)&gt;&gt;T</td>
</tr>
<tr>
<td>11</td>
<td>A×B×0?</td>
<td>25</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>A×B×0?</td>
<td>26</td>
<td>NOP</td>
</tr>
</tbody>
</table>

### 3.2 Control Logic on FPGAs

In this work, both computation and controlling tasks are implemented on FPGAs. The controlling task is the configuration process of the RPUs, including configuration context generation, context packing, RPU synchronization, etc. The computation task is the implementation of ED kernels as shown in Fig. 1. In this section, we first present how the control logic is designed on the FPGA and then introduce the implementation of ED in the following section.

The reconfiguration and scheduling process of all the RPUs are supervised by a scalable soft-core-based microcontroller implemented on the FPGA, which is referred to as the RPU microcontroller (RMC). There are three pipelined operations performed on RMC for the reconfiguration and scheduling process: 1) the configuration flow is parsed and the corresponding configuration data is located; 2) if the configuration data is cached in RMC, they are directly sent to RPUs; 3) if not, an extra pre-fetching operation which is responsible for pre-loading corresponding configuration data from external memory, will be carried out between the former two operations.

The proposed RMC contains a $1 \times M$ soft microprocessor (μP) element array, a $1 \times 2$ bitstream processor array (SPA), and a specific cache called Context Group Control Unit (CGCU), as shown in Fig. 5. The SPA is a reconfigurable bitstream decoder, and consists of two bitstream processor elements (SPEs) in order to process data fields efficiently. The μP element array is responsible for carrying out control-intensive tasks including the supervision of the reconfiguration process. The CGCU is responsible for the configuration data pre-loading and distributing, including three kinds of sub-operations: pre-loading configuration data from external memory, caching configuration data in each μP, and sending configuration data to RPUs.

The configuration context is generated and then sent to RPUs by the RMC. The configuration context generation process is as follows: firstly, the μP element array parses the input options of system control flow which are results of entropy decoding tasks, such as intra prediction modes and motion vectors, and then locates the configuration data; secondly, CGCU prepares the configuration data (i.e., pre-loading configuration data from external memory), caches configuration data in RMC, and then organizes and sends configuration data to RPUs in a specified format (i.e., configuration packing).

To support the multiple-standard video decoding algorithms, eight soft microprocessor cores are implemented on the main FPGA. Since the proposed architecture is a soft-core based architecture, a big advantage that it is highly scalable to meet different performance and power requirements. As shown in Fig. 6 (a), an optimized mail-box-based RPU-FPGA interface is designed to improve the communication efficiency between the two types of computation engines. Communication messages are sent and received through these mail boxes. Each mail box provides an output interrupt signal which is connected to its associated RPU. By using the proposed architecture, the eight RPUs can be arranged in parallel mode or sequential mode to support different application execution needs.

To improve the system flexibility and efficiency, a mail-box-based RPU-FPGA communication scheme is proposed. Figure 6 (b) shows the control-flow of the μP elements. Each μP element is designed to work in two successive modes. The first mode is mail-driven mode, in which stage μP element can be woken up by a mail message sent by the master controller. The master controller can be either an RPU or another μP element. As soon as the μP element responses to the mail message, the μP element enters the raw config-

![Fig. 5 The proposed RPU microcontroller.](image)

![Fig. 6 The proposed mail-box-based RPU-FPGA interface (a) and communication scheme, (b) between the main FPGA and RPUs.](image)
uration package (RCP)-FIFO-driven mode. In this mode, the µP element checks the data FIFO’s status, and parses the RPU configuration contexts. In the output state, µP element writes top-level contexts into CGCU for one RPU’s processing mission. After the tasks are finished, the µP element returns back to the mail-driven mode and is ready for the next mail message. The µP elements, which are in the waiting status, would be switched off for power efficiency. The mail-box-based communication scheme allows the access of each RPU to any µP element, and the access of different µP elements among each other. Therefore, controlling tasks can be efficiently pipelined with processing tasks, and a controlling task can be accelerated on several µP elements, which allow a significant improvement of the system performance and flexibility.

4. Algorithms Mapping

As been analyzed in Sect. 2, the word-level compute-intensive decoding tasks are mapped on RPUs, while the bit-level control-intensive decoding tasks and the RPU controlling tasks are mapped on FPGAs. Among the four standards, HEVC is most complicated and heavy work-loaded one, therefore, we will take HEVC as an example to present the proposed algorithm mapping schemes.

4.1 Implementing ED on FPGA

A multiple-standard entropy decoder architecture for a multi-standard video (i.e., MPEG-2, AVS, H.264, and HEVC) decoding is proposed in this work as depicted in Fig. 7 (a). The MPEG2-VLD module is used for the decoding of variable length coding (VLC) in MPEG-2, CA2D-VLD module for the decoding of context-based adaptive two-dimensional variable length coding (CA-2D-VLC) in AVS, CAVLD for the decoding of context-based adaptive variable length coding (CAVLC) in H.264 decoding, CABAD module is designed for the decoding of context-based adaptive binary arithmetic coding (CABAC) in H.264 and HEVC, and UVLD for the decoding of universal variable length coding (UVLC), such as Exp-Golomb codes and unsigned integers, in MPEG-2, AVS, H.264 and HEVC. The MEM_ACC module reads bitstream data and stores them into a FIFO, FIFO_ctrl reads bitstream data from MEM_ACC, and then provides them for the aforementioned ED modules, and it also performs other operations (e.g., shift operations for input bitstream, identifying start codes, etc.), And the Result_FIFO module is a buffer of parsed results. Several architectural optimizations have been conducted in the proposed ED engine to improve the system performance and reduce circuit area.

CABAC has the best compression rate and highest computational complexity in these ED algorithms. In this paper, the proposed architecture and optimization schemes of CABAD are introduced. The block diagram of CABAD is illustrated in Fig. 7 (b). The CABAC decoding task can be configured either for H.264 or for HEVC by the controller module. And the decoding engine module, which is the core part of CABAD, is designed to implement arithmetic decoding. As shown in Fig. 7 (c), four optimization schemes are proposed in the arithmetic decoding process to shorten the critical paths as follows: (1) speculative computations provide the parallelism between the update of the two probable paths (i.e., the most probable and the least probable paths) and the bin decision; (2) a dedicated look-up table replaces the traditional iteration shift scheme when implementing the renormalization process of the least probable path; (3) logic balance scheme is adopted to decrease the critical path by dividing a table into two tables in the case that the subinterval of least probable symbol (i.e., R_{LPS}) is derived by a fixed 2-D table, indexed by the two variables, i.e., range on the critical path and the probability state.
(pState) on the non-critical path; (4) reordering scheme is adopted to provide the parallelism of a subtraction operation and a look-up table operation of $R_{LPS}$ in the reordered calculation of $off = (offset - range) + R_{LPS}$ when performing the calculation, i.e., $off = offset - (range - R_{LPS})$, on the least probable path. This engine outputs one bin per cycle, so the critical path optimization schemes extremely enhance CABAC decoding efficiency.

4.2 Mapping Schemes of Compute-Intensive Decoding Tasks on an RPU

The block-based word-level compute-intensive decoding tasks have a large effect on the whole decoding performance as described in Sect. 2. Hence, how to effectively map the compute-intensive decoding tasks on RPUs is important. In this sub-section, the features of compute-intensive decoding tasks (i.e., MC, IP, IT, DF, SAO) will be given firstly, and then, optimization and mapping schemes will be proposed.

4.2.1 Feature Analysis

The main features of MC sub-algorithm are: (1) data pattern is block-based, therefore, block-level parallelism can be utilized to improve performance and throughput; (2) in the same predicted block, different predicted samples share the same calculation pattern, and calculations of predicted samples are independent from each other. Therefore, the block-based calculation pattern in MC sub-algorithm can be seen as a loop calculation without data dependency between adjacent loops; (3) HEVC supports variable prediction block sizes, ranging from $64 \times 64$ down to $8 \times 4/4 \times 8$.

In HEVC, the actual IP prediction process is performed on a transform unit (TU) basis. A TU includes a luma transform block (TB) and two chroma TBs. There is no data dependency among the calculations of different predicted samples in the prediction process of a luma/chroma TB. Moreover, there is no data dependency among the prediction processes of luma and two chroma TBs in a TU. These features provide possibility to perform IP calculations in parallel, i.e., sample-level parallelism of IP calculation of a TB on an RCA and block-level parallelism of IP calculations of luma and chroma TBs on an RPU.

In HEVC, IT may use $N$-point IDCT ($N$ from 4 to 32) or IDST4 $\times$ 4 used for intra $4 \times 4$ luma TB. The IT process includes column and row transform sub-processes. The 1-D transform of each column/row is independent from each other with the same calculation pattern, so the 1-D column/row transform sub-process also can be seen as a loop calculation without data dependency between adjacent loops.

In HEVC, the in-loop filter process includes two stages: DF process and then SAO filter process. HEVC in-loop filter is carried out based on a picture rather than a coding tree unit (CTU) similar to an MB in AVS, MPEG-2 and H.264. A CTU includes a luma coding tree block (CTB) and two chroma CTBs. Therefore there is a high latency between the in-loop filter and other decoding modules at CTU level in an HEVC decoder. And a high latency also exists between the DF and SAO when performing HEVC in-loop filter at CTU level, because the SAO of the first CTU in a picture can only be carried out after completing the DF of the second CTU in the second row CTUs of a picture to ensure right referencing neighboring samples for the edge offset (i.e., EO, one of two types of SAO modes).

The filtering process of DF in HEVC is performed on a grid of $8 \times 8$ samples, both for luma and chroma components. And the filtering calculation of a vertical/horizontal edge can be seen two segment-based independent filtering calculation processes, i.e., the filtering calculation processes of segment 1 and segment 2, as illustrated in Fig. 8. Therefore, block-level parallelism can be used in DF mapping. And the filtering calculation of process of a segment is a loop calculation without data dependency between adjacent loops. SAO is performed on a CTU after the completion of the DF process. SAO aims to improve the accuracy of the reconstruction of the original signal amplitudes by adding an offset value to each sample adaptively. The SAO filtering calculation process of a CTB can be seen as a loop calculation without data dependency between adjacent loops.

![Fig. 8](image-url) The proposed CTU-based HEVC in-loop filter implementation scheme.
4.2.2 The Proposed Optimization and Mapping Schemes

**Block-level parallelism for compute-intensive decoding tasks.** Block-level parallelism can be used for the mapping of compute-intensive decoding tasks on the RPU to improve the performance and throughput, as described in 4.2.1 subsection.

**Loop execution pipeline (LEP) technique for loop calculations.** Loop calculations exist abundantly in compute-intensive decoding tasks (e.g., MC, DF, SAO, etc.). Therefore, the optimization for loop calculations is an important scheme to improve the decoding performance. The LEP technique is presented to improve the efficiency of an \(N\)-time loop calculation when no data dependency exists between adjacent loops. For an \(N\)-time loop calculation without using a pipeline technique, the total calculation time is expressed as \((t_{in} + t_c + t_o) \times N\), where \(t_{in}\) is the cycle number of fetching data from input FIFO into the RCA; \(t_c\) is the cycle number of completing one-time loop calculation; \(t_o\) is the cycle number of taking the calculation results from the RCA into output FIFO. For the \(N\)-time loop using the LEP technique, the total calculation time becomes \((t_{in} + t_c + t_o) + (N - 1) \times G\), where \(G\) is a configuration parameter called loop gap, i.e., the cycle number of start-time difference between adjacent loops, which ensures the results of loop calculations can be output correctly from the RCA into output FIFO. For the \(N\)-time loop using the LEP technique, the total calculation time becomes \((t_{in} + t_c + t_o) + (N - 1) \times G\). The performances of the key sub-algorithms in MPEG-2, AVS, H.264 and HEVC are measured and listed in Fig. 9. In order to compare the performance of these sub-algorithms fairly, the CTU size in HEVC is set to \(16 \times 16\), equal to the size of an MB in MPEG-2, AVS, and H.264, when making the experiments of HEVC sub-algorithms on an RPU, so the unit cycles/MB is used on the vertical axis in Fig. 9. And the complexity of key sub-algorithms in these four video standards can be seen more clearly. The complexity of the mapped MBs increases in the order of MPEG-2, AVS, H.264 and HEVC, the main reason is that the increase in the

**Repartition scheme for in-loop filter mapping at CTU** level by coupling DF and SAO. DF in HEVC is performed based on a picture, namely, the vertical edges in a picture are filtered firstly, followed by the horizontal edges. In order to make sure that the filtered results of the DF at CTU level are in accordance with those based on a picture, a part of the horizontal edges of the most right \(8 \times 8\) blocks of the left CTU, i.e., a horizontal boundary (indicated by purple lines in Fig. 8) of two neighboring \(4 \times 4\) blocks, will be filtered after completing the filtering of all the vertical edges of the current CTU, similarly, a part of the horizontal edges of the most right \(8 \times 8\) blocks of the current CTU will be filtered after completing the filtering of all the vertical edges of the right CTU. Hence, it is reasonable and desirable to adopt a scheme of shift-left-four-column and shift-top-four-row which makes the size of data block to be equal to the size of a CTU and avoids RPU to fetch redundant data from an external memory, as depicted in Fig. 8. The source data of SAO are from the results of the DF. Hence, in order to carry out HEVC in-loop filter at CTU level and couple the DF and SAO, the current filtered data block need shift left one column and shift top one row again after performing the DF, and data blocks to be processed by the SAO are indicated as dashed boxes in Fig. 8. Therefore, the proposed HEVC in-loop filter can be implemented and pipelined along with other decoding modules at CTU level in an HEVC decoder by using this repartition scheme, reducing the latency between the in-loop filter and other decoding modules in an HEVC decoder from the decoding time of a frame to the decoding time of a CTU and coupling the DF and SAO easily to implement in-loop filter at CTU level.

Fig. 9  Performance comparison of the key sub-algorithms in MPEG-2, AVS, H.264 and HEVC.
tap number of the used filters in these four video standards makes the MC sub-algorithms more complex. The complexity of the IDCT $8 \times 8$ sub-algorithms is mainly decided by the matrix coefficients used for matrix multiplications in the four video standards. There is no in-loop filter process in MPEG-2; and the in-loop filter of AVS or H.264 only includes DF process, whereas that of HEVC is composed of two parts, i.e., DF first and then SAO. Over AVS using up to four tap filters, the more complex up to five tap filters are used for the DF in H.264; moreover, the DF in H.264 is performed on a grid of $4 \times 4$ rather than $8 \times 8$ samples, and this further makes its computational complexity higher than the DF in AVS; compare to H.264, the complexity of HEVC in-loop filter is slightly higher than one of H.264 in-loop filter because of the introduction of SAO, although the DF in HEVC has lower computational complexity over the DF in H.264.

5. Implementation Results

5.1 Chip and Platform Implementations

To fully test and verify the functionality and performance of the proposed RPU architecture, we implement the reconfigurable processor core RPU as a single chip named after CHAMELEON. As shown in the die micrograph of Fig. 10, the CHAMELEON prototype chip is implemented on a 6.5 mm $\times$ 6.2 mm die by using TSMC 65 nm low power one-poly eight-metal (LP1P8M) CMOS process. Since many test I/O pins are intentionally implemented, the final die of CHAMELEON is pad-limited, resulting in a big die size compared with the 5.4 mm $\times$ 3.1 mm RPU core.

The proposed mixed-grained reconfigurable computing platform is finally implemented as shown in Fig. 11. The hardware platform includes one large scale FPGA (Stratix IV EP4SE820F43C3) and one small scale FPGA (Cyclone III EP3C120F780C7) on the motherboard, one large scale FPGA (Stratix II EP2S90F1020C3), eight selectable RPUs and a storage disk on the daughterboard. The router FPGA is designed to connect the eight RPUs on the daughterboard and also be used as connector with the FPGAs on the motherboard, so that data flows from the large FPGAs or other RPUs can be routed and streamed to any desired RPUs for processing. Such architecture increases the flexibility of the whole system. The small FPGA using 921 logic elements on the motherboard is only used to configure the two large FPGAs when the system powers up. It does not implement any computing functional blocks in the system.

To support multiple-standard video decoding, a scalable decoding engine is implemented on the proposed reconfigurable computing platform. The functional block diagram of the decoding engine is shown in Fig. 12. The decoding engine utilizes two identical RPU chips. It also contains an RMC, a master microcontroller (MMC), a direct memory access controller (DMAC), an external memory interface (EMI) and an entropy decoder, which are implemented by the two large FPGAs. The RMC contains eight $\mu$P elements as analyzed in Sect. 3.2. An MMC is used to initialize the RMC and other peripheral circuits, which include an interrupt control module, a DMAC controller, a 128 Kbytes program memory and a dedicated 64-bit EMI, through a
Table 2 1080 HD decoding performance with two RPU chips.

<table>
<thead>
<tr>
<th>Standard</th>
<th>Profile &amp; level</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-2</td>
<td>Main profile @ high level</td>
<td>73 fps</td>
</tr>
<tr>
<td>AVS</td>
<td>Right profile @ level 6.0</td>
<td>68 fps</td>
</tr>
<tr>
<td>H.264</td>
<td>High profile @ level 4.0</td>
<td>57 fps</td>
</tr>
<tr>
<td>HEVC</td>
<td>Main profile @ level 4.0</td>
<td>52 fps</td>
</tr>
</tbody>
</table>

Table 3 Comparisons when performing H.264 decoding.

<table>
<thead>
<tr>
<th>Item</th>
<th>This work</th>
<th>[3]</th>
<th>[12]</th>
<th>[13]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology (nm)</td>
<td></td>
<td>65 (RPU1)</td>
<td>90 (Stratix II)</td>
<td>40 (Stratix IV)</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td></td>
<td>250</td>
<td>450</td>
<td>333</td>
</tr>
<tr>
<td>Area</td>
<td>Two RPU’s: 80.6mm²</td>
<td>75mm²</td>
<td>210mm²</td>
<td>N/A</td>
</tr>
<tr>
<td>Video format</td>
<td></td>
<td>1920×1080</td>
<td>1920×1080</td>
<td>1920×1080</td>
</tr>
<tr>
<td>Performance (fps)</td>
<td></td>
<td>57</td>
<td>24</td>
<td>30×</td>
</tr>
<tr>
<td>Normalized performance</td>
<td></td>
<td>0.23</td>
<td>0.05</td>
<td>0.09</td>
</tr>
<tr>
<td>Normalized performance comparison (this work / other work)</td>
<td>1×</td>
<td>4.6×</td>
<td>2.56×</td>
<td>1.28×</td>
</tr>
</tbody>
</table>

Table 4 Comparison of HEVC decoder on different platforms.

<table>
<thead>
<tr>
<th>Item</th>
<th>This Work</th>
<th>[14]</th>
<th>[15]</th>
<th>[16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology (nm)</td>
<td></td>
<td>65(RPU)</td>
<td>90 (Stratix II)</td>
<td>40 (Stratix IV)</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td></td>
<td>250</td>
<td>3100</td>
<td>3500</td>
</tr>
<tr>
<td>Area</td>
<td>Two RPU’s: 80.6mm²</td>
<td>1406.25 mm²</td>
<td>1406.25 mm²</td>
<td>3.17 mm²</td>
</tr>
<tr>
<td>Video format</td>
<td></td>
<td>1920×1080</td>
<td>1920×1080</td>
<td>1920×1080</td>
</tr>
<tr>
<td>Performance (fps)</td>
<td></td>
<td>52</td>
<td>102</td>
<td>about 98</td>
</tr>
<tr>
<td>Normalized performance</td>
<td></td>
<td>0.028</td>
<td>0.033</td>
<td>0.028</td>
</tr>
<tr>
<td>Normalized performance comparison (this work / other work)</td>
<td>1×</td>
<td>6.30×</td>
<td>7.43×</td>
<td>1.33×</td>
</tr>
</tbody>
</table>

The performance is for 24-core configuration.

5.2 Performance Evaluation and Comparison
To evaluate the performance of the proposed reconfigurable computing platform, we implemented the four-standard (i.e., MPEG-2, AVS, H.264, and HEVC) 1080 high definition (HD) video decoding on the proposed platform, and both of the two RPU’s shown in Fig. 12 are used, running with an operating frequency of 250 MHz. The measured performances in terms of decoding frame rate are reported in Table 2. For each video format, fifty-four encoded video streams are tested and the average decoding speeds are recorded.

More specifically, we compare the H.264 decoding performance with the XPP-III coarse-grained reconfigurable processor [3], a state-of-the-art many-core processor [12] and a dedicated multi-format video codec chip [13] in Table 3. From the measured data, one can clearly see that the performance of the proposed platform is 4.6×, 2.56×, and 1.28× that of the XPP-III processor, many-core and ASIC designs, respectively. The normalized performance is derived by Eq. (1).

\[ \text{Normalized performance} = \frac{\text{Performance}}{\text{Frequency}} \]  

(1)

Table 4 compares the performance of the proposed platform performing HEVC decoding with other schemes. The decoding throughput achieved by this work reaches up to 1920 × 1080 with 52 fps under a 250MHz working frequency. Compared with other works, normalized performance of this work is 6.3×, 7.43× and 1.33× that of the two GPP-based schemes ([14] and [15]) and one ASIC scheme [16], respectively.

6. Conclusion
This paper has presented a mixed-grained reconfigurable platform for multiple standard video decoding applications. Eight coarse-grained reconfigurable processing units (i.e., eight RPU’s) and two FPGA fabrics can be dynamically utilized in this platform. Thanks to the proposed flexible reconfigurable PEA architecture, the proposed platform achieves a 3.6× speedup for H.264 HD decoding compared with the XPP-III processor. The proposed platform can also perform HEVC HD decoding with 52fps under 250MHz. The measured performance is 6.3× and 7.43× that of the two GPP-based implementations, respectively.

References
[4] D. Rossi, F. Campi, S. Spolzino, S. Pucillo, and R. Guerrieri,


Leibo Liu received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree in Institute of Microelectronics, Tsinghua University, in 2004. He now serves as an associate professor in Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing and VLSI DSP.

Dong Wang received the PhD and MS degrees in electronic engineering in 2010 and 2006 from Xi’an Jiaotong University, China. He has been a visiting student in the Computer Science Department of University of California Los Angeles during 2009 - 2010. He is currently working in Beijing Jiaotong University at the Institute of Information Science. His research interests include computer arithmetic for reconfigurable and high performance computing architectures for embedded applications.

Yingjie Chen received the M.S. degree in Institute of Microelectronics from Tsinghua University, Beijing, China, in 2009. She currently serves as a research fellow in Institute of Microelectronics, Tsinghua University. Her research interests include multimedia computing and reconfigurable computing.

Min Zhu received the B.S. degree from the Department of Micro & Nano Electronics, Tsinghua University, Beijing, China, in 2006, where he is currently working toward the Ph.D. degree in the Institute of Microelectronics. His research interests include reconfigurable computing and multimedia processing.

Shouyi Yin received the B.S., M.S. and Ph.D. degree in Electronic Engineering from Tsinghua University, China, in 2000, 2002 and 2005 respectively. He has worked in Imperial College London as a research associate. Currently, he is with Institute of Microelectronics at Tsinghua University as an associate professor. His research interests include mobile computing, wireless communications and SoC design. Dr. Yin has published more than 20 refereed papers, and served as TPC member or reviewers for the international key conferences and leading journals.
Shaojun Wei was born in Beijing, China in 1958. He received Ph.D. degree from Faculte Polytechnique de Mons, Belgium, in 1991. He became a professor in Institute of Microelectronics of Tsinghua University in 1995. He is a senior member of Chinese Institute of Electronics (CIE). His main research interests include VLSI SoC design, EDA methodology, and communication ASIC design.