ABSTRACT
We present a systematic methodology for exploring application partitioning and assignment together with platform architecture exploration. Streaming applications with multiple runtime modes are considered. The platform architecture is based on a domain specific MPSoC architecture template. We show results using complete inner modem physical layer processing of wireless applications, WLAN and LTE. We show that the proposed methodology obtains up to 30% energy improvement in energy with negligible area overheads as compared to straightforward mapping to one processor, while meeting performance constraints, for a multi-mode WLAN 11n system and single-mode LTE system.

Categories and Subject Descriptors
C.3 [Computer Systems Organization]: Special-Purpose and Application-based Systems—Real-time and embedded systems

Keywords
Streaming applications, Heterogeneous MPSoC, Partitioning, Mapping, Co-Design

1. INTRODUCTION
It is imperative in today’s growing constraint on low power design for embedded systems to go for heterogeneous MPSoC (Multiprocessor System-on-Chip) designs. Although heterogeneous MPSoCs have energy, performance and cost benefits, such platforms are increasingly becoming complex and often have irregular memory hierarchy organizations. With the advent of ASIP (Application Specific Instruction Set Processor) based MPSoC architectures, the complexity increases further as they have multiple heterogeneous configurable processors.

One of the key challenges in a heterogeneous MPSoC design is partitioning and assignment (P&A) of the application and the architecture instantiation of the underlying platform. Partitioning is the design decision for splitting the application in to several parts. Assignment is the step of binding the application elements to the system resources. Architecture instantiation involves exploring architecture configuration parameters, both at the platform and the individual processors level.

The problem of P&A is relatively simple when dealing with a given fixed platform and has been researched [1][2]. The architecture instantiation for an already partitioned application is also straightforward, where mostly application assignment and scheduling is explored [3][4]. However, the design of a MPSoC architecture and P&A are not orthogonal to each other and are coupled to each other. In the co-design space, [5][6] traditionally partitioning decisions were limited to hardware-software partitioning only, where the design space is limited. Most of the works in ASIP based MPSoC designs with P&A exploration [7][8][9] either do not consider complete application and focus on specific parts only or carry out limited exploration. Further, the problem of P&A and architecture exploration gets amplified exponentially with applications with multiple modes (runtime profiles) [10] or even more in case of multiple applications [11]. Thus, P&A exploration “together” with platform architecture instantiation for mapping multi-mode streaming applications, is still an unsolved problem in the state of the art.

The key contributions of this paper are: (a) Proposes systematic methodology for early exploration of the P&A “together” with platform architecture exploration, (b) Proposes systematic approach for ASIP based MPSoC platform instantiation (c) Targets complete real-life multi-mode streaming applications (complete inner modem processing of the physical layer of wireless applications, WLAN 11n [12] and LTE-3GPP [13]) (d) Achieves better than worst-case energy efficiency for platform instantiation to support multi-mode applications

The proposed methodology has been shown to achieve energy gains with negligible area overheads by carrying out fine grained P&A exploration by considering the static and runtime dynamism across and within the modes. The methodology generates multiple heterogeneous partitions such that the tasks assigned to a partition are well matched in complexity, parallelism, duty cycle and hardware requirements, and do not have conflicting requirements. This ensures energy efficiency while minimizing the area overheads. We achieve energy gains of up to 30% for WLAN and LTE, with negligible area overheads and while meeting performance constraints, as compared to straightforward mapping to single processor. Although, this paper uses wireless ap-
application drivers, the proposed methodology is generic and can be applied on other streaming multi-mode applications such as MPEG-2/MPEG-4, etc.

The paper is structured as follows: Section 2 gives an overview of the related work followed by description of the platform template in Section 3. Section 4 discusses details of the proposed methodology, followed by results and conclusions in Sections 5 and Section 6, respectively.

2. RELATED WORK

System scenarios [14][15] and use-case scenarios [16] based approaches have been proposed for mapping multi-mode applications, i.e. with runtime dynamism, on heterogeneous MPSoC architecture. However, these [3][4][17] approaches have the limitation in a way that they either do not explore partitioning of the application tasks on a heterogeneous architecture template or [2][16][18] assumed a fixed underlying architecture and have limited scope for platform architecture exploration. In contrast, in our work we explore P&A of multi-mode applications along with exploring the instantiation of the platform from an application domain specific template (refer Table 1 in Appendix F). Moreover, our work is complementary to the scenario based approaches. We do not focus on creating the scenarios but consider representative scenarios for the application(s) and carry out the P&A and MPSoC architecture exploration for mapping these scenarios on the same platform. Our approach can handle multiple scenarios and is completely compatible with scenario based approaches.

Hardware/software (HW/SW) partitioning has been researched well [5][6]. Here, the P&A decisions are relatively straightforward as the application has to be partitioned across generic processors (SW) or custom hardware blocks (HW). The design space here is much limited as compared to P&A exploration on ASIP based MPSoC architectures, which is the focus of our work.

Most of the industrial mappings of wireless standards on SDR (Software Defined Radio) platforms, carried out for complete receivers [9], typically neither explore P&A nor analyze the trade-offs in details. Besides this, PHY layer implementations mostly focus on on efficient implementations for the computationally dominant blocks, such as MIMO detectors and FFT [7][8].

In this paper, we extend our previous work [19] which focused on P&A exploration for single mode and static behaviour of the application. Here, we present a methodology to handle dynamic runtime behavior of different types of multi-mode streaming applications. The dynamism of the input applications is assumed to be well-defined at the design time.

To the best of our knowledge, our work is the first work that systematically explores, early in the design flow, P&A and architecture together for mapping of multi-mode application(s) on a MPSoC platform. We are also not aware of any work which can achieve energy efficient mapping with negligible area overheads for such multi-mode applications, besides minimizing energy overheads for the lower complexity modes when mapped with worst case complexity modes on the same platform.

3. ARCHITECTURE TEMPLATE

We consider application domain specific architecture templates, where the template has been designed and optimized for applications of a particular domain. Figure 1 shows the high level architecture template considered in this work. This template is assumed to be designed for streaming applications like wireless standards and multimedia. The figure shows both the platform level and the processor level template. At the platform level there are multiple processors, all of which share a unified instruction and data level-2 (L1+D L2) memory. The communication between the processors is point-to-point using DMA (Direct Memory Access) transfers. The number of processors, size of L2 memory and the bus width are configurable.

At the processor level, each processor consists of scalar and vector datapaths (DP), and local instruction and data memories. We assume single issue for scalar as well as vector datapaths. For the targeted multi-mode streaming applications, which are inherently data-parallel, we can get reasonably accurate estimates of area and energy by considering data-level parallelism (DLP) only. Although we do not consider instruction-level parallelism (ILP), the results of the methodology are valid even if ILP is considered. The pipeline depths for the scalar and the vector datapaths are 3 and 7, respectively. These have been obtained based on targeted application requirements at the targeted 40nm process technology. The local instruction (IMEM) and data (DMEM) memories are software-controlled, separated (scalar and vector) and hierarchical (i.e. each consisting of two levels, L1 and L0). The L0 instruction memory is essentially a loop buffer whereas L0 data memory is a Register File (RF). The datapath of the processor consists of computational operators specific to application domain requirements. For each processor, the memory sizes, the scalar and the vector word size, and the type and number of compute operators can be configured. We mostly focus on the vector datapath as it is the more dominant part of the design in terms of area, power and other metrics, especially for the application domains considered.

Although this template is basic and high level, it captures the essential elements and provides configuration parameters also provided in other academic templates such as SODA [20] and commercial ones such as Tensilica [21], CEVA [22].

4. PROPOSED METHODOLOGY

In this section, we discuss the proposed methodology for carrying out the P&A exploration, a high level view of which is shown in Fig. 2. The methodology explores the complete design space and generates a set of P&A schemes under given constraints (including performance). It estimates area and energy for each P&A scheme.

4.1 Application Characterization

Streaming applications such as wireless communication (WLAN, LTE), multimedia (MPEG-2, MPEG-4), etc. sup-
port multiple modes (runtime profiles) and exhibit runtime dynamism. We handle this dynamism in the input applications in two steps. Firstly, the multiple modes of the application are clustered into a number of system scenarios [14]. Each scenario is modeled as a separate control and dataflow graph (CDFG) $G_s$. Secondly, for each scenario, if different types of the input data (such as preamble and payload symbols in case of WLAN or I, P, B type frames in case of MPEG-2) results in a different control and data flow of the execution, each such execution flow is considered as a separate control path graph (CPG) in $G_s$. Refer to Appendix D for an illustration. The execution of these CPGs is considered to be mutually exclusive in time. Splitting $G_s$ into multiple CPGs abstracts away the heterogeneity in the control flow of $G_s$ while preserving the data flow dependency, besides making the proposed methodology generalized enough to handle multiple modes of one or more applications.

The number of CPGs created will depend on the granularity of the nodes in the CDFG and the granularity of the input data. In the most simple case, the entire CDFG can be considered as a single CPG which will result in worst-case over-dimensioned system. To keep the number of CPGs manageable, we consider each task (node) in $G_s$ to be a functional step in the application. A task is also considered as the lowest granularity of the partitioned application during the P&A exploration. The application is partitioned into these tasks based on the communication and computation complexities, and to minimize communication overhead. It is also assumed that the inter-task data transfers are blocked-transfer and at a symbol or frame level.

Each $G_s$ is characterized separately by internally profiling each of its tasks to estimate its compute and data-memory related complexities for the processing of one unit of input data, which is a symbol or a frame in case of WLAN/LTE or MPEG-2/MPEG-4, respectively. The profiling is carried out on a simple template architecture with no parallelism. It is an one time effort for each input $G_s$.

Each task-level operation is mapped to one or more operators supported in the platform template. The set of different platform operators required and the number of times each of them is executed is calculated for each task to estimate its compute complexity. The memory sizes and the data-memory access complexity for each task are estimated using the sizes of the input, output and the intermediate data, required for the processing of one symbol/frame by the task. The inter-task data transfers are considered as the communication complexity of the tasks. The data transfers between the datapath and the memories (L1-D and RF) are considered as the data-memory access complexity of the tasks.

Using these estimates, each task in the CPG is annotated with a 5-tuple $\langle \Omega_n, \epsilon_n, \Psi_n, \phi_n, \rho_n \rangle$, referred to as the characterization tuple, where $\Omega_n$, $\epsilon_n$ and $\Psi_n$ denote task complexity per iteration, number of iterations and the set of required hardware resources, respectively, $\phi_n$ and $\rho_n$ denote the available degree of data-level parallelism at the application and the task level, respectively. Thus, $\epsilon_n$ is an upper bound on $\rho_n$. All the tasks in the same CPG will have the same values of $\epsilon_n$ and $\phi_n$.

It may be noted that in this case, $\Omega_n$ is calculated assuming blocked transfer of data between the tasks. The methodology can be easily extended for the streaming transfer of data between the tasks by recalculating $\Omega_n$ based on the size of the data chunk that is received before the task starts processing.

### 4.2 Platform Template Characterization

In this step, the datapath operators, RF and L1-D memory components of the platform are characterized. Physical synthesis and hand-annotated gate level simulations are carried out to build a component level area and energy library. This library is used to derive an empirical cost model, using regression analysis, to estimate area and energy for different platform components at different configurations. This step is also one time effort. It has to be repeated only if a new template is considered, or if new components are added to the existing template.

### 4.3 P&A Schemes Derivation

Here, we discuss the algorithm for systematically deriving all the P&A schemes (Algorithm 1). This step receives as input the set of all CPGs derived for all the modes of the input application. It may be noted that the proposed P&A exploration is complete under the assumptions that the tasks in the CPGs are not further partitioned and that only blocked data transfer is considered between the tasks.

These streaming applications offer parallelism at task (TLP), data- (DLP) and instruction-level (ILP) and at different granularity levels. The proposed P&A algorithm explores TLP by pipelining the execution of the tasks in the CPGs. The extent of pipelining explored depends on the latency constraints imposed by the application specifications (refer Appendix E). Pipelining decisions have major impact on the platform configuration, cycle budget allocation across different processors in the platform, etc.

Coarse-grained DLP is explored at the task level by partitioning input data streams and processing them in parallel. Fine-grained DLP is explored at the processor level using vectorization/SIMD (Single Instruction Multiple Data). Although ILP is not explored, it does not restrict the effectiveness of the proposed methodology, as discussed in Section 3. The P&A algorithm iterates over different cases of pipelining (referred to as pipeline schemes) and derives P&A schemes for each of these cases, as described below. The algorithm has been implemented as a tool and it’s time complexity is $O(n \cdot p \cdot s)$, where $n$ is the total number of nodes in the input CPGs, $p$ is the number of pipeline stages in a scheme and $s$ is the number of pipeline schemes being explored.

#### 4.3.1 Taskgroup Creation (Lines 2-4)

For each CPG, create_taskgroups() divides its tasks into disjoint sets called task groups based on the tasks’ characterization information. The objective is to create task...
groups with balanced complexity while preserving the data flow dependency among the tasks, and also to ensure that tasks with similar degree of parallelism are grouped together. This impacts the number of processors instantiated in the platform and configuration of each processor.

Each task group (tg) is annotated with a 6-tuple \( < \Omega_g, \epsilon_g, \Psi_g, \phi_g, \rho_g, \kappa_g > \) obtained based on the characterization tuple of the tasks (n) grouped in tg. \( \Omega_g \) denotes the total complexity of tg and is also used as a measure of latency as the hardware is not yet defined. This is a reasonable assumption since we assume blocked data transfers between the tasks. \( \Psi_g \) is the union of hardware resources requirement across all the tasks in tg. \( \rho_g \) is the maximum degree of parallelization of tg, which is limited by the task with the minimum degree of parallelization (\( \rho_n \)) in tg. \( \kappa_g \) denotes the degree of hardware parallelism required for the task to complete its processing in the allocated cycle budget. \( \epsilon_g = \epsilon_n \) and \( \phi_g = \phi_n \) as these are same for all the tasks in a CPG.

4.3.2 Task Cluster Creation (Line 5)

\( \text{create_taskclusters()} \) merges the task groups (tg) across all the CGPs and creates a set of task clusters (\( = \) number of pipeline stages). Each task cluster (tc) is assigned to a separate stage. Task groups of each CPG are assigned a cycle budget based on the latency of the pipeline stage and \( \kappa_g \) for each task group is calculated. Typically, each tc is a candidate for being mapped on a single processor as well as being partitioned further for exploring coarse-grain DLP. Each partition (tsc) across all the CPGs and creates a set of task clusters (\( = \) number of pipeline stages). Each task cluster (tc) is assigned to a separate processor. Thus, this is a very crucial step in ensuring handling of inter- and intra-mode dynamism of the application in an energy and area efficient manner.

Each task cluster (tc) is also annotated with a 5-tuple \( < \Psi_{tc}, \rho_{tc_{\text{min}}}, \rho_{tc_{\text{max}}}, Z_{tc}, TG_{tc} > \) based on the task groups assigned to it. \( \Psi_{tc} \) is the union of hardware resources required across all the tg in the tc. \( \rho_{tc_{\text{min}}} \) and \( \rho_{tc_{\text{max}}} \) are the minimum and the maximum degree of parallelism, respectively, for the tc. \( Z_{tc} \) is the set of unique 2-tuple \( < \epsilon_{tg}, \kappa_{tg} > \) across all the tg and \( TG_{tc} \) is the set indicating all the tg clustered in the task cluster.

4.3.3 Task Sub-Cluster Creation (Line 7)

Although, \( \text{create_tasksubclusters()} \) attempts to create task clusters with homogeneous task groups, due to the data flow dependency constraints, it is possible that a task cluster may contain task groups having significant variance in the available DLP (\( \phi_g \)), frequency of execution (\( \epsilon_g \)) and the required hardware parallelism (\( \kappa_g \)). The objective of \( \text{create_tasksubclusters()} \) is to analyze each task cluster, and if it contains task groups with large variation in \( \phi_g, \epsilon_g, \kappa_g \) values, the task cluster is split into one or more task sub-clusters based on the clustering of \( \phi_g, \epsilon_g, \kappa_g \) values. Like, task clusters, each task sub-cluster is also a candidate for being mapped on a single processor as well as being partitioned further for exploring coarse-grain DLP. Each sub-cluster (tsc) is also annotated with a 5-tuple \( < \Psi_{tsc}, \rho_{tsc_{\text{min}}}, \rho_{tsc_{\text{max}}}, Z_{tsc}, TG_{tsc} > \).

4.3.4 Partition Creation (Line 8)

\( \text{create_partitions()} \) creates one or more partitions for each task cluster or task sub-cluster (in case the cluster has been split into sub-clusters), as shown in Algorithm 2. It creates partitions at three granularity levels (denoted by Level 1, 2, 3a/3b in Algorithm 2) to cover the complete P&A space. Partition is assigned to a separate processor. Thus, the number of partitions in a P&A scheme decides the platform configuration. If the platform template imposes constraints on the maximum number of processors, this can be considered in Algorithm 2 to limit the partitioning choices.

Level 3 partitions are created to explore coarse-grained DLP. The maximum number of level 3 partitions (k) created depends on the maximum value of the degree of parallelism (\( \rho_{tsc_{\text{max}}} \)) of the task cluster, the application characteristics and platform template constraints. Each partition (P) is also annotated with a 3-tuple \( < \Upsilon_P, \kappa_P, \Psi_P > \). \( \Upsilon_P \) is a set of 2-tuple \( < \epsilon_g, \phi > \) where \( \epsilon_g \) is the task group assigned to the partition and \( \phi \) denotes the number of input data streams processed by this \( \epsilon_g \). \( \kappa_P \) is the required degree of hardware parallelism and \( \Psi_P \) is the set of the hardware resources required for the partition.

4.3.5 P&A Schemes Creation (Line 10)

Partitioning at each level, in the previous step, can be considered as separate intra-taskcluster P&A scheme. These schemes are combined across the task clusters to obtain the complete set of P&A schemes for each case of pipelining. Schemes across all the pipelining cases form the complete set of P&A schemes for the input application.

4.4 Platform Instantiation

Platform is instantiated for each P&A scheme separately. The methodology configures the number of processors at the
platform level (= number of partitions in the P&A scheme, as mentioned in Section 4.3.4). At the processor level, it configures parameters for the type and number of compute operators, vector word size and data memory sizes (RF and L1). The methodology currently assumes a fixed communication structure and ignores level-2 memory hierarchy as it is not crucial for streaming applications being considered.

The entire platform for a P&A scheme is instantiated by instantiating each of its individual processors. A processor is instantiated based on the characterization tuple of the partition assigned to it. The degree of parallelization (PF) required to meet the latency requirements is considered equal to the partition’s κPF value. Since we consider DLP only, PF implies the word size for the datapath and the width of the RF and the L1-D memory. The set of compute operators specified by ΨPF are instantiated at a SIMD width equal to PF. The required size of L1-D memory for the processor is considered equal to the maximum of the size of L1-D estimated for each task mapped on the processor. The depth of the RF has to be configured based on the application profiling and currently has to be provided by the designer.

4.5 P&A Scheme Characterization

The total area and energy of a P&A scheme are estimated by individually characterizing each processor in the scheme. This is done using processor’s configuration parameters, tasks mapped to the processor and template’s area and energy models.

The total area of a processor is estimated by calculating areas of datapath and the memories (RF and L1-D) separately. The datapath area is calculated based on the compute operators instantiated in the processor and their SIMD factor, whereas areas of the RF and L1-D memory are calculated using their sizes. For energy estimation, the compute, data-memory access and communication energies are calculated separately. These are calculated based on the set of tasks mapped to the processor, the amount of input data each task processes and the processor configuration. We do not discuss characterization in details due to space constraints in the paper.

5. RESULTS

5.1 Experimental Setup

We have considered a uniform quantization of 16 bits for all data variables. The word size and the frequency considered for the platform template are 16-bits and 700 MHz, respectively. The template has been characterized at 40nm G process technology. The datapath operators are implemented using commercial standard cells and L1-D memory using commercial memory generators. The register file is D flip-flop based. Physical synthesis back-annotated gate level simulations have been performed to obtain area and energy numbers for these components at different bit widths.

5.2 Application Drivers

We have used complete inner modem of the physical layer of wireless applications, WLAN 11n [12] and LTE-3GPP [13], as the application drivers (refer Appendices A and B). These have in-house developed industry quality code base. These are fairly large applications with several thousand lines of code. We consider two modes of WLAN 11n, i.e. 2 × 2 40MHz and 4 × 4 40MHz both with QAM-64 modulation, and single mode of LTE, i.e. 2 × 2 20MHz with QAM-256 modulation.

We carry out single-mode and multi-mode P&A exploration for LTE and WLAN, respectively. In each case, we explore area-energy trade-offs across different P&A schemes while meeting the required performance constraints. The 2 × 2 and the 4 × 4 modes of WLAN are considered as two different scenarios, where the 2 × 2 mode is an average complexity scenario whereas the 4 × 4 mode is a worst-case complexity scenario. It is also assumed that 2 × 2 mode is the more frequently occurring scenario for WLAN.

We chose WLAN 11n and LTE because they have very different characteristics and provide very different constraints and opportunities for P&A exploration (refer Appendix C). As mentioned in Section 1, although this paper uses wireless application drivers, the proposed methodology is generic and can be applied on other streaming multi-mode applications such as MPEG-2/MPEG-4, etc.

5.3 P&A Exploration Results

Design Space Coverage: In case of WLAN 11n, the methodology derives 18 P&A schemes with number of processors ranging from 1 to 11 by considering both the 4 × 4 and the 2 × 2 modes together. Whereas in case of the single mode of LTE, 29 P&A schemes are derived with number of processors ranging from 1 to 18. The design space obtained in each case is complete under the given constraints and the assumptions that the tasks (nodes) of the input CDFG are carried out considering either fixed architecture or a given assumption that the tasks (nodes) of the input CDFG are.

Figure 3: WLAN 4 × 4 mode (combined platform)
The results have been normalized to single processor based implementation. The proposed methodology results in P&A schemes which are more energy efficient and at significantly lower area, as compared to the StoA approach. The proposed methodology achieves up to 30% energy gains as compared to single processor implementation and up to 20% energy gains as compared to the StoA approach, without incurring any area overheads (P&A schemes labelled as B, D, F, I, L and O). The area overheads are minimized in the proposed methodology because the tasks assigned to a partition are very well matched in complexity, parallelism, duty cycle and hardware requirements, and do not have conflicting requirements. P&A schemes E, M, P and Q also achieve energy reductions similar to the above mentioned P&A schemes but with at least 10-20% area overhead because in these schemes the lower complexity tasks were also parallelized. The methodology explores up to 3 pipeline stages beyond which the latency requirements of WLAN 11n cannot be met.

Figure 4 shows similar results for LTE where the proposed methodology also achieves up to 30% energy gains as compared to single processor implementation with area overhead of only 5-10%. LTE is a memory intensive application, unlike WLAN, and the memory dominates the total area. Our results are better up to 2.5x and 8x in energy and area, respectively as compared to the StoA approach [17] because the memory sizes and accesses are also taken into account while carrying out the P&A exploration.

Better than Worst Case Energy Efficiency: The area-energy trade-offs for the 2 x 2 mode, when mapped with 4 x 4 mode on the same platform, shows similar trends as the 4 x 4 mode. Hence, we do not show results for it separately. However, the trend changes when 2 x 2 mode is mapped independently on a platform. This is because the clustering of task groups and the partitioning of task clusters changes depending on other modes being mapped together. Besides, 2 x 2 mode is a lower complexity mode with lesser parallelization possibilities.

We compare the most efficient P&A scheme for 2 x 2 mode on independent platform with the pareto-optimal schemes obtained for 2 x 2 mode in the combined exploration with 4 x 4 mode. Although, the area overhead in each case is at least 2x, the energy overheads lies between 0-7%.

Thus, the proposed methodology guarantees mapping of low complexity modes without or negligible energy overheads on platform which also supports higher complexity modes.

6. CONCLUSIONS

We have proposed a systematic methodology for exploring application partitioning and assignment together with platform architecture instantiation. It has been shown for real-life streaming multi-mode wireless applications, WLAN 11n and LTE, that the proposed methodology obtains energy gains up to 30%, with negligible area overheads and while meeting performance constraints, as compared to single processor mapping solution. The methodology also enables mapping of lower complexity modes together with higher complexity modes on the same platform without energy overheads.

7. REFERENCES

APPENDIX

A. WLAN 11N APPLICATION DESCRIPTION
An 802.11n packet, as shown in Fig. 5, consists of two parts, i.e. (a) preamble with predetermined data, and (b) payload consisting of the actual data symbols. Both the preamble and the payload consists of multiple 4μs symbols. The preamble contains the header information for the packet, such as number of transmit and receive antennas, channel bandwidth, coding and modulation schemes, number of payload symbols, etc. The payload contains the transmitted data.

802.11n has several modes based on the number of transmitting and receiving antennas, modulation schemes, data-coding rate, etc. We have considered two modes of 802.11n in this work, i.e. 4 × 4 and 2 × 2 MIMO 40MHz with 64-QAM modulation. The number of preamble symbols is 7 and 5 for the 4 × 4 and 2 × 2 modes, respectively, whereas the number of payload symbols can vary from 1 to 2048 [12].
In a N × N MIMO mode, each symbol is a set of N × Q complex values where N is the number of transmit and receive antennas and Q is the number of sub-carriers per symbol which is dependent on the channel bandwidth. In our case, N is 2 for 2 × 2 and 4 for the 2 × 4 and 4 × 4 modes, respectively and Q is 128 for the channel bandwidth of 40 MHz.

The block diagram for the inner-modem processing is shown in Fig. 6. All the shaded blocks carry out preamble processing whereas the rest carry out payload processing. The complexity of a 802.11n packet processing depends on the selected mode and the number of payload symbols in the packet.

B. LTE APPLICATION DESCRIPTION
The input data of LTE has a very different format as compared to WLAN. In case of LTE, the transmission is based on frames of 10ms duration. Each frame consists of 10 sub-frames, each of which further consists of 2 slots as shown in Fig. 7. A sub-frame is the smallest unit of time which is allocated to a user. Each slot consists of 7 symbols. Thus, a LTE frame consists of multiple 7μs symbols.

LTE also has several modes similar to WLAN 11n. We have considered only a single mode, i.e. 2 × 2 MIMO 20 MHz with 256-QAM modulation.

A LTE symbol can also be considered as a set of N × Q complex values, where N is the number of antennas and Q is the number of sub-carriers. In our case, N is 2 for the 2 × 2 mode and Q is 1200 for the channel bandwidth of 20 MHz. Unlike WLAN, LTE does not have dedicated symbols carrying only the header information. All the 7 symbols in a slot carry the transmitted data and the symbols 0 and 4 also carry the header information.

The block diagram for the LTE inner-modem processing is shown in Fig. 8. The figure shows the processing for the 7 symbols of each slot, the complexity of which mostly depends on the selected mode.

C. COMPARISON OF WLAN 11N AND LTE
WLAN has very tight latency constraints whereas LTE is very data dominated. As a result of this, WLAN requires higher computational resources whereas LTE requires higher memory resources. Both WLAN and LTE have packet based input data but WLAN packets have large number of smaller symbols whereas LTE packets have lower number of larger symbols. Thus, processing on the input data in case of WLAN is largely uniform resulting in a very regular control and data flow (CDF) whereas in case of LTE it is non-uniform resulting in an irregular CDF. WLAN can thus be more effectively pipelined than LTE, which results in higher utilization of the hardware resources. Besides these, parallel input data streams can be processed in parallel for large part in the WLAN application chain. In case of LTE, the parallel input data streams are merged after conversion from time to frequency domain and the granularity of data-level parallelism changes, thus impacting the possibilities of partitioning and parallelization of the LTE tasks. Thus, WLAN and LTE provide very different constraints and opportunities for P&A during the mapping.

D. INPUT APPLICATION MODELING
The 2 × 2 and the 4 × 4 modes of WLAN are considered as two different scenarios, whereas the 2 × 2 mode is an average complexity scenario whereas the 4 × 4 mode is an worst-case complexity scenario. It is also assumed that 2 × 2 mode is the more frequently occurring scenario for WLAN. Fig. 9 shows the CDFG (G_{4×4}) for the 4 × 4 mode scenario. In this graph, the control and data flow are denoted by the dotted green and the solid blue edges, respectively. Fig. 10 shows the CPGs created for G_{4×4} and the input data (symbols) which each control path graph processes.

E. EXTEND OF PIPELINE CALCULATION
For streaming applications, the processing of input data can be pipelined where the tasks in the processing chain are partitioned across different pipeline stages. However, pipelining increases the overall latency of the entire processing. The total latency for processing N input data on a pipelined architecture with S stages can be given by Eqn. 1.

\[ L_{pipe} = N + (S - 1) \cdot L_{stage} \]  

where, \( L_{pipe} \) and \( L_{stage} \) are the total and per stage latency, respectively. The application constraint on the allowable latency will impose an upper bound on the extent of pipelining, i.e. the number of pipeline stages, as shown in Eqn. 3.

\[ L_{pipe} \leq N \cdot L_{data} + L_{app} \]  

\[ S \leq \frac{N \cdot L_{data} + L_{app}}{L_{stage}} + N + 1 \]  

where, \( L_{app} \) is the total allowable latency and \( L_{data} \) is the allowable latency for processing each input data, at the application level.

For example, in case of WLAN 11n, a new symbol arrives at every 4μs. We consider that each symbol must be processed within 4μs (=L_{data}) and \( L_{stage} = L_{data} \). The WLAN 11n standard specifies a latency constraint, referred to as SIFS timing budget, which has been considered to be 8μs (\( L_{app} = 2 \cdot L_{data} \)) for the inner-modem processing, in this work. Thus from Eqn. 3, the maximum number of pipeline stages is constraint to 3. Thus, for the P&A exploration of WLAN, we consider three cases of pipelining, i.e. with one-, two- and three-stages. We derive P&A schemes for each of these cases.

F. COMPARISON WITH EXISTING APPROACHES
In Table 1, we compare the features of the proposed approaches to other existing approaches. These approaches do not explore partitioning. Each of them considers one or more application as a set of given tasks and explore allocation and binding of these tasks on the underlying architecture. In the proposed approach, instead of starting with a given set of tasks, we systematically split the application into a set of tasks and then systematically derive different partitions, considering multiple parameters such as available parallelism, hardware resource requirements, complexity, etc. These partitions are then allocated on the underlying architecture. The existing approaches consider architectures which are either fixed or have limited configurability. In contrast, in the proposed methodology we consider template based domain specific architecture where we explore the instantiation of the platform architecture along with partitioning and assignment. The methodology instantiates a heterogeneous MIPSoC platform for each P&A scheme explored where the number of processors and configuration of each processor is also explored.
Table 1: Comparison of proposed methodology with existing approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Multi-mode</td>
<td>Multi-mode</td>
<td>Single-mode</td>
<td>Multi-mode</td>
<td>Multi-mode</td>
<td>Multi-mode</td>
<td>Multi-mode</td>
</tr>
<tr>
<td>Partitioning</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Platform</td>
<td>Heterogeneous</td>
<td>Homogeneous</td>
<td>Homogeneous</td>
<td>Heterogeneous</td>
<td>Heterogeneous</td>
<td>Heterogeneous</td>
<td>Heterogeneous</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
<td>High</td>
</tr>
</tbody>
</table>

![Figure 8: LTE inner modem processing](image)

![Figure 7: LTE frame structure](image)

![Figure 9: Task graph for 4 x 4 mode (G_{4x4})](image)

![Figure 10: Control path graphs for G_{4x4}](image)