Compiler back end design for translating multi-radio descriptions to operating system-less asynchronous processor datapaths

Dipnarayan Guha
Centre for High Performance Embedded Systems, Nanyang Technological University, Singapore
Email: guha@ntu.edu.sg

Thambipillai Srikanthan
Centre for High Performance Embedded Systems, Nanyang Technological University, Singapore
Email: astsrikan@ntu.edu.sg

***Abstract*—Most asynchronous processor Instruction Set Architectures (ISA) are based on a single type of underlying asynchronous circuit design style. Asynchronous processor ISAs are entirely dependent on the type of asynchronous design style chosen and can support a limited set of simple applications only. Design reuse is typically difficult to realize in such cases. In this paper, we show a behavioral model of a predictor circuit system that configures an application profile-driven asynchronous processor ISA comprising two asynchronous design styles. The predictor circuit system is used to translate application profile and multi-radio code to the processor datapath through a compiler back-end. The target is an asynchronous processor that does not run an operating system and is used both as a complement and alternate to software-defined radios with high degrees of design reuse.**

***Index Terms*—asynchronous processor, ISA, predictor circuit, multi-radio, compiler back-end, design reuse**

I. Introduction

Common asynchronous processors' underlying design styles include QDI (Quasi-Delay Insensitive) [1], STAPL (Single Track Handshake Asynchronous Pulse Logic) [2], and STFB (Single Track Full Buffer) [3] self-timed circuit families. The asynchronous processor Instruction Set Architecture (ISA) is entirely dependent on the asynchronous design style type and performance metrics like datapath performance and branch prediction for multi-level instructions remain difficult to be streamlined for multiple sets of application profile tasks.

This paper targets asynchronous processors based on a hybrid of asynchronous design styles (QDI and STAPL), where reconfiguration in the circuit components of underlying asynchronous design styles can take place based on the application profile.

This paper further describes the behavioral model of a predictor circuit system that can translate application profile and radio protocol code to the asynchronous processor datapath directly through a compiler back-end. This design methodology serves two purposes: 1. helps manage complex code mapping onto processor datapaths in the absence of an operating system, 2. improves upon asynchronous design compiler back-ends in generating hybrid asynchronous design style handshaking circuits by incorporating application profile and radio protocol module (finite state machine executions) codes in a single framework.

This paper is arranged as follows. Section II discusses the design approach of a hybrid asynchronous processor that supports multi-radios in the absence of an operating system; Section III discusses the arithmetic of event sequences in asynchronous datapath models, Section IV shows the system behavioral modeling math of the predictor circuit, Section V shows the evaluation of the predictor circuit system, Section VI shows the evaluation of the compiler back-end design and Section VII brings about the conclusion.

II. Design Approach of a Hybrid Asynchronous Processor supporting multi-radios without an OS

Handsets and portable consumer electronic devices typically need to support diverse applications and usually provide multiple radio choices for communications. The multi-radio choices are usually managed by the operating system through driver switching among the individual protocol Network Interface Cards (NIC). In the absence of an operating system and software management of protocol modules, the processor must be able to support these radio protocol modules when invoked by the application. Till now, the only radios that are supported on operating system-less asynchronous processor cores are small-sized radios like TR1000 [4,5]. It is very difficult to map large complex radio stacks optimally onto the asynchronous core, radio protocol description language translation being a major bottleneck. The asynchronous core and datapath is derived out of a single type of asynchronous design style and realizing programmability in reconfigurable style select based on radio-based application profiling is very difficult.

Based on "Reconfigurable Frame Parser Design for Multi-Radio Support on Asynchronous Microprocessor Cores", by Dipnarayan Guha and Thambipillai Srikanthan which appeared in the Proceedings of the IEEE International Conference on Computing: Theory and Applications, ICCTA 2007, Platinum Jubilee of the Indian Statistical Institute, Calcutta, India, March 2007. © 2007 IEEE.

In the absence of an operating system, the major challenge becomes to design efficient micro-architectures that respond to external interrupts generated by high-level description languages. Application codes and radio descriptions are first profiled to identify common functional modules in a micro-architecture aware manner, so that the processor datapath can be configured in exactly the manner as it would be if a software interrupt was invoked by an operating system. A predictor circuit system forms a key component of this micro-architecture aware translation of protocol and high level description languages to the asynchronous processor datapath.

Most of the current embodiments of asynchronous processors using reshuffled communication processes in pipelined asynchronous circuits are based on QDI circuit models [1]. A difficulty with QDI design in supporting arbitrary frequent changes in the application profile is the difficulty in satisfying the requirement of correct working of the design in presence of unbounded operator delay variances. Ensuring the property that the asynchronous processing entity (the processor instruction set) does not get locked to a particular radio protocol, it is important to ensure that there is a mechanism to switch over the radio mode after a self-inferred time interval. For QDI, if this is to be achieved, checker circuits to detect occurrence of transitions (as in radio protocol stacks that are handled if an operating system is present) are needed even though they would happen anyway. This can be overcome by STAPL circuits that eliminate the acknowledgement and data reset phases of the four-phase handshaking protocol used in QDI circuits.

In this paper, a hybrid QDI-STAPL design style processor datapath has been considered. The predictor circuit system is also comprised of this hybrid design style, and behavioral modeling has been carried out to conform to the arithmetic of event sequences in asynchronous datapath models.

III. Arithmetic of Event Sequences in Asynchronous Datapath Models

LAD (Length Adaptive Data) [4, 5] is commonly-used arithmetic in modeling event sequences in asynchronous processor datapaths and instruction set architectures. This is a particularly useful arithmetic logic where bit-serial instruction set architectures is tuned in line with the length adaptive data words. This proposition is extremely useful for implementing design reuse in asynchronous processor based platforms.

Currently, LAD arithmetic axioms support only one task at a time, and do not allow data interleaving or delimiter re-assignment. A few extensions of the LAD arithmetic axiomatic foundations to include simultaneous concurrent multi-tasks by collaboratively re-assigning delimiter LAD digits (A delimiter bit is a bit upon whose occurrence, all the leftmost bit positions are taken to be of the value of the delimiter bit) has been shown in [6, 7].

Earlier finds [6] have suggested enhancing LAD arithmetic in the context of nondeterministic data-flow computation for collaboratively assigning delimiter LAD digits. This is a system design challenge because it involves non-deterministic dataflow computation (based on the fact that data arriving should be processed immediately) and computations with arbitration (for quality-of-service priorities), which makes the usual LAD architecture embodiments in related asynchronous processor designs difficult for parallel re-conversion and optimized LAD arithmetic management. Delimiters can occur anywhere within a data word. An example of this situation is that an adder may encounter 0111 + $\underline{1}$111, where $\underline{1}$ is a delimiter bit. The result should be a single 1, and not 1111, which is wastage of energy for processor datapath compression. This problem is even more pronounced for nondeterministic data-flow computation maps that are an inevitability of multi-radio support in asynchronous processors, primarily due to arbitrary radio invocations and data interleaving. This work is currently being further studied and we present some of our initial finds on addressing this issue through a system behavior model of a predictor circuit that maps application tasks to the processor datapath through multiple radio class linkage associations.

IV. System Behavioral Modeling Math of the Predictor Circuit

The system behavioral model of the predictor circuit is based on the following assumptions:

1. Application tasks are randomly distributed and can occur at any time (any type of application can request a service at any time).
2. A radio is invoked by an application (the application that wants service will need to be communicated).
3. A radio can be invoked by multiple applications at the same time (the same radio may be used in communicating multiple applications with a requested quality of service simultaneously).
4. Multiple radios are necessarily invoked by multiple applications at the same time (multiple radio interfaces are present in the communicating device).
5. The association of application task job processes to radio class linkage sets is instantaneous. (Radio invocation is done instantaneously in the device as soon as an application arrives for communications. This is done locally based on guarantee of quality of service and to support the processing and communication capabilities of that application).
6. Application task job processes can shift across radio class linkage sets. (This means that the tasks can be partitioned and locally assigned to a computing group ad-hoc based on the nature of computation similarity).
7. Priority of the task job processes generated by multiple applications may change during the processing stage. (This is to accommodate applications that may need more time-critical resource-based processing. Following point 6, this also means that there is provision to locally

change the task job process priority associated with a particular radio class linkage set. To guarantee quality of service, if an application is advertised using a radio type for which the channel is degraded, the application may start to be advertised using a different radio on the same physical channel but different performance characteristics)

8. Tasks separated into sets invoked under different radio linkage sets may be shifted internally within the set groups. (This is to ensure that even if the control channel condition changes, data are still communicated with the quality of service requested across different radio media available. This is a local policy and internal to the predictor circuit for helping choose the radio select for communicating)
9. An application has to be immediately processed and cannot be buffered (Without an operating system and without significant interrupt handling support, the tasks would need to be executed as soon as they arrive to prevent radio lock-in and degradation of quality of service).

A. Mathematical assumptions

1. The n radio linkage class sets that services application task job processes is considered to form a n dimensional vector space.

2. The mathematical representative values of the predictor circuit system resources are constrained to the range of (0, 1). This means that the predictor circuit system handling a particular type of application invoking only one radio class linkage set is represented by (0,0,0…) (no load constraint) and the condition that it maps multiple applications invoking all the radio class linkage sets is represented by (1,1,1,….), signifying a fully loaded constraint.

3. The optimal cost function for task admittance across different radio class linkage sets spans over this n dimensional vector space.

4. The n dimensional vector can be used to describe the current state of the predictor system, and the norm of this vector forms the basis of the ranking of the predictor circuit load in term of radio class linkage set associations.

5. For trying to make the problem of multi-application task job process allocation across all radio class linkage sets and sub-partitions of the vector space NP hard, jobs are described and ranked using the same computation resource requirement dimensions. Jobs can be added and subtracted from the predictor system state vector.

6. The system behavior of the predictor circuit is modeled in terms of the unit function that is a dual variable of radio class linkage sets and resource demand from an application task job process.

B. Techniques for application's service request task job fit across the radio class linkage sets

First Fit: Partition the vector of radio class linkage sets equally among all the application tasks that requests service

Best Fit: Partition the vector of radio class linkage sets optimally and shift task job processes to that radio class linkage set which most closely fulfils the application's service request with the least number of jobs involved.

Worst Fit: Partition the vector of radio class linkage sets and shift task job processes to that radio class linkage set which most closely fulfils the application's service request with the maximum number of jobs involved.

C. Predictor circuit system load estimation (in terms of application jobs processed through radio class linkage set associations) and process job shift

The load distribution mechanism in the predictor circuit system receives global state updates (in terms of applications processed and radios invoked) arbitrarily in time. This presents a closure problem of the prediction of the degree of associating a task to radio class linkage sets as no immediate feedback to a job placement is provided. Therefore the prediction subsequent to the first in each time frame are made with data that is known to be out of date, and problems similar to those encountered by the least loaded initial fit model may arise.

One advantage of using predicted resource requirements in terms of processed application job tasks for different radio linkage class sets is that the same trial fitting technique that is used to select the datapath map can be used to provide an estimate of the system load after job shift and placement. Thus the result of the fitting computation is used to directly update the load value stored for the datapath map.

Characteristics of the partitioned vector class model in the system behavior description of the predictor circuit can be modeled under this assumption.

A key issue of the predictor circuit system load estimation is the actual job shift of application task processes in order of the radio linkage class set association priority involving change in ongoing task job execution priority status. The other issue is the resolution of priority conflicts for time-critical and non-time critical tasks where the level of priority is subject to change. Mathematically, this means that the vector space partitioning will change and dimensional representation might not remain the same.

D. Mathematical modeling of the predictor system (target machine compilation unit)

Let

p_i = The application task corresponding to radio linkage class set i; process corresponding to class i.

E = Set of eligible processes, i.e. the number of tasks that can be shifted among radio linkage class sets

τ = Expected process lifetime, i.e. the time till a process finishes executing

C = Cost of shifting the process among radio linkage class sets

k = Normalization constant.

$e = |E|$

Then

$$\forall p_i, p_i \in E, iff, p_i.\tau > k * p_i.\tau \text{ ….. (1)}$$

Let

c_i = Condition of predictor system at time $(t = t_i)$, in terms of the radio class linkage occupancy vectors.

j_i = Job in the predictor system at time $(t = t_i)$, for shifting the task process among radio linkage classes.

Fundamentally, the optimization problem for computing resource allocation in the predictor system for mapping onto the processor datapath would be:

$$\min(\|c_i + j_i\|)\forall i \text{ ….. (2)}$$

, i.e. minimize the norm across all the transition process vectors.

The general selection policy for job shifting across the radio linkage classes is thus given by:

$$\max(\|p_i\| * \frac{\tau_i}{C_i})\forall i \in e \text{ ……. (3)}$$

There exists a lesser than the least feasible solution in terms of general discrete (GD) cost functions to a class partitioning problem if the optimal cost threshold value is at least $\sum_i C_{i_{MAX}} / n$ .During the shift of application tasks instantaneously over the n radio linkage classes, there remains to be shown and proven that there exist a feasible cost that is always attained which is lesser than a least threshold value. This means that the cost of transferring application tasks of a particular radio linkage class internally during radio protocol language mapping to the processor datapath does not overall reduce the penalty for buffering it without processing immediately.

In order that the predictor system modeling for individual radio linkage classes on a one-to-one basis holds, we shall show that the cost of transferring application tasks across different radio linkage classes subjects the behavioral modeling to be the same as when the tasks are processed individually by the predictor on a per-radio linkage basis.

Let $x \in p_i$ be an optimal partition across E for some vector space $\|c_i + j_i\|$. The physical interpretation of this is that the radio linkage classes form a convex function set and has an optimal partition across the set of vector spaces spanned by the radio linkage classes. The difference is that these classes are set statically. The issue of NP-hardness comes into consideration if the space $\|c_i + j_i\|$ spanned by c_i and j_i behaves quasi-statically at every time when there is a transition among the p_i 's.

The questions to be answered in this problem formulation are thus:

1. Does there exist an "optimal" partition of a quasi-static vector space V_1 for an element transition p_i defined from a convex function set F_1 to another convex function set F_2 defined in another quasi-static vector space, V_2 ?

2. Is this problem NP-hard?

Let us first focus on the definition of the convex functions across quasi-static vector spaces.

A function c is called strongly convex if for three points x_1 , x_2 and x_3, the following result holds:

$$c(x_2) < \frac{(x_2 - x_1) * c(x_1) + (x_3 - x_2) * c(x_3)}{x_3 - x_1} \forall x_1 < x_2 < x_3$$

….. (4)

From (1), Let us construct a set $P = \{p_i\}\forall p_i$ over some τ , the expected process lifetime.

For a bounded $\tau = \tau_{MAX}$, P will be a bounded set with some least upper bound P_{upp} …. (5)

For the sets P_i and P_j , thus we can choose the triplets $(x_1^{P_i}, x_2^{P_i}, x_3^{P_i})$ and $(x_1^{P_j}, x_2^{P_j}, x_3^{P_j})$ such that equation (4) holds for each of these triads on their respective domain sets….. (6)

A function $F(P_i)$ can thus be defined to be convex for the triplets $(x_1^{P_i}, x_2^{P_i}, x_3^{P_i})$ across P_i .Similarly, a function $F(P_j)$ can be defined to be convex for the triplets $(x_1^{P_j}, x_2^{P_j}, x_3^{P_j})$ across P_j … (7)

From the definition of c_i in equation (2), the vector space $V(c_i)$ spanning across the set c_i can be defined as the basis span of $(x_1^{P_i}, x_2^{P_i}, x_3^{P_i})$, essentially noting the fact that the condition of the predictor system c_i at time $(t = t_i)$ where $(t_i < \tau_{MAX})$ is a permutation of the elements of P_i . From (5), because of the boundedness of P_i , there exists a function $F(P_i)$ which can be permuted exactly to a convex function F^{C_i} such that $F^{C_i} \equiv \{x_1^{P_i}, x_2^{P_i}, x_3^{P_i}\}$ and $\|V(c_i)\| \equiv \|V(F^{C_i})\| \equiv \|c_i\|$ …. (8)

Similarly from (8), $\|V(c_j)\| \equiv \|V(F^{C_j})\| \equiv \|c_j\|$ (9)

From the definition of vector spaces,

$\left\|V(c_i + c_j)\right\| \le \left\|V(c_i)\right\| + \left\|V(c_j)\right\| \Rightarrow \left\|c_i + c_j\right\| \le \left\|c_i\right\| + \left\|c_j\right\|$ …… (10)

From the definition of j_i in equation (2), a space $V(j_i)$ spanning across the set j_i can be thus defined on the basis span of $(x_1^{P_j}, x_2^{P_j}, x_3^{P_j})$, essentially noting that job of shifting a process in the set E across the space $V(c_i)$ for defined c_i is a function $F(P_j)$ that maps the $\{j_i\}$'s to some elements $\{c_i^{'}\}$'s for $t = [t_i, t_i^{'}]$ defined in τ and bounded by τ_{MAX}. …. (11)

As $\{c_i^{'}\}$ are permutation elements of c_i's, the result from (11) and (8) is that $F(P_j)$ maps to F^{C_i} …..(12)

$V(j_i)$ is then a subspace of $V(c_i)$ …..(13)

From equation (10) it thus follows that

$\left\|j_i + j_j\right\| \le \left\|c_i + c_j\right\|$ …. (14)

and from (10) again,

$\min(\left\|c_i + j_i\right\|, \left\|c_j + j_j\right\|) \le \left\|c_i\right\| + \left\|c_j\right\|$ ….. (14)

This shows that an "optimal" partition of the quasi-static vector spaces spanned by the radio linkage classes does exist…. (15) Q.E.D.

We now show that indeed this is a NP-hard problem.

Let us define Q as the optimal partition of the quasi-static vector spaces spanned by the radio linkage classes, based on equations (1) through (15). From (11) and (13), it is easy to see that $Q = \min(\left\|c_i^{'}\right\|)$ …. (16)

Let us define a Vector Space V now spanning over $(c_i + j_i)$, drawn from the elements c_i in $V(c_i)$ and j_i in $V(j_i)$ … (17)

From (13), we can say that

$c_i = \beta j_i$, where β is a scalar…. (18)

We define $Q^{'}$ as $\{q_i\}$, where $q_i = 1$, if c_i maps exactly to j_i, which means, there's perfect in-order processing of individual application tasks with all the different radio linkage classes. $q_i = 0$, otherwise…. (19)

We can thus solve a subset problem of set $Q^{'}$, derived from V, over the elements $\{q_i\}$ and a bound $Q_b^{'}$, such that $Q_b^{'} = \sum_i q_i - Q$ …..(20)

From the cost function definition (1) and the policy of job shifting across different radio linkage classes (equation (3)), we can say from Equations (16), (17), (18) and (19) as the fact that $Q_b^{'}$ exist *iff* $\exists i = n$ for which $\sum_i q_i = Q$ …. (21)

The physical interpretation of this statement is that the selection policy chooses the application tasks in that particular radio linkage class to shift for which there's a maximum in the cost function for shifting, i.e. the match between the number of tasks to shift matches precisely the number of tasks associated in that radio linkage class which are being mapped onto the datapath by the predictor system at that instant of time.

This proves that the problem for quasi-static vector spaces spanned by the multi-radio linkage classes is NP-complete. (Q.E.D)

Subsequent to this, we shall now prove a more powerful relation in respect to determining the optimal radio linkage association cost function when mapping multiple application tasks to the asynchronous processor datapath through individual radio linkage associations.

Suppose the set $Q^{'}$ has n elements in all. Let us also assume that the total set of all possible radio linkage classes is K. This is to generalize the problem more, the practical consequence of which is that more radios can be supported in the same platform through firmware upgrades, patches and entirely new software suites for newer radio standards that may come in the future. Let the cost function, as defined from Equation (3), on $Q^{'}$ be $C_Q^{''}$ …. (22)

Then, the cost per each radio linkage class $C_Q^{'} = C_Q^{''} / K$ …..(23)

In the usual case, the cost function for application task admittance to a particular radio linkage class (determines which radio is invoked for which application) is denoted by C_c.

At any point of time $t < \tau_{MAX}$, there exists some $\{q_i\}$ for which equation (21) is satisfied. This banks on the assumption that the processor is not idle as the tasks are mapped onto the datapath through individual radio linkage classes. Mathematically, this is equivalent to the fact that there is an optimal kernel map among quasi-static vector spaces.

For K, thus there exists a minimal threshold for each partitioned radio linkage sub-class, the value of which is given by Equation (14).

We will prove now that $C_Q^{'} \ge (1 - \varepsilon) C_c$, which signifies that task admittance cost function for a set of multiple applications invoked across different types of radio linkage class associations converges towards the task admittance cost function where a particular radio linkage class is associated with a particular application.

There is thus the provision of handling application task processes even if the precedence level changes in the order of radio class linkage set elements currently being executed, as well as any new application tasks that are generated which invokes a different radio, creating a new association to a radio linkage class.

From the properties of any set, we have $C_Q^{''} - nQ_b^{'} \le nK$ …..(23)

Thus,

$$C_Q^{''} \ge n(Q_b^{'} - K) \Rightarrow C_Q^{''} \ge KQ_b^{'} - nK \text{ …. (24)}$$

From equation (20), obviously, $KQ_b^{'} \ge KC_c$, as for any vector subspace,

$$\left\| \sum_i^n \sum_j^n e_i + e_j \right\| \le \left\| \sum_i^n e_i \right\| + \left\| \sum_j^n e_j \right\|$$

Partitioning into subspaces always increases the overall norm of the subspace [8].

Thus from equation (24), we have $C_Q^{'} \ge Q_b^{'} - n$, which reduces to $C_Q^{'} \ge C_c - \varepsilon C_c$, where ε is a scalar expressing the cost function of task admission to all the radio class linkage sets. Thus, we have,

$$C_Q^{'} \ge (1 - \varepsilon)C_c \text{ …….. (25)}$$

Now, again, we can write from Equation (23)

$$C_Q^{''} \le nQ_b^{'} + nK ,$$

which reduces to $C_Q^{'} \le (n/K)Q_b^{'} + n$ …..(26)

Since $KQ_b^{'} \ge KC_c$,

$$1/KQ_b^{'} \le 1/KC_c \Rightarrow n/KQ_b^{'} \le n/KC_c \text{ …. (27)}$$

Hence from (26), we have

$$C_Q^{'} \le n(1 + 1/KC_c) \Rightarrow C_Q^{'} \le (1 + \varepsilon)C_c \text{ …… (28)}$$

From Equations (25) and (28) it is now easy to see that $C_Q^{'}$ **has a positive convergence towards** C_c**.**

From Equations (25), (26) and (28), the convergence is of the order of $\log \varepsilon$, by expanding binomially equations (25) and (28). This shows an important result. While the dynamic shifting of application task processes across the partitioned radio linkage classes helps in reducing the task admittance cost function, this is independent of the overall predictor system behavior, and can be processed instantly without buffering, saving on costly hardware and pattern matching circuits. The mathematical analysis also shows that the predictor system circuit can work well under conditions where multiple applications invoke multiple radios, the physical interpretation of which is - multiple applications can be supported on the asynchronous processor platforms increasing the utility of such devices while extending battery life (reducing dynamic power consumption by doing away with the OS)

The mathematical analysis forms the basis of designing the predictor circuit system and is the key strategy behind the handshaking circuit design in terms of time and concurrency. The system is behaviorally modeled considering independent radio behavior and linkage class generation based on per-application task set type, whereas the operation of the circuit involves internal task process transitions across radio linkage classes. This behavior is then modeled to eliminate computations with arbitration and change in precedence levels of the executed jobs. The problem of detecting "when" a particular event occurs (an application task is partitioned and linked to a particular radio linkage class) is reduced to a trivial case of accommodating it in any particular class set as and when it comes along. There is no need for special detector circuits for doing this costly operation.

V. Evaluation of the Predictor Circuit System

Based on the extended LAD mathematics [6, 7], an intermediate protocol description and asynchronous behavioral model language is being investigated currently in conjunction with the predictor circuit system design. We decided to use the Balsa [14] asynchronous synthesis language and compiler interface because it is open-source and easily customizable. Balsa generates QDI handshake circuit models for synthesis, and we first looked at how to extend the compiler back-end to generate handshaking models other than QDI based on our extended LAD arithmetic.

A. Modeling using the Balsa toolset

Based on the behavioral modeling of the predictor system, it becomes possible to extend a concept of programmable state level encoding in delay-insensitive pipelines where priority of the jobs being executed can change at random. This can help in keeping data and delimiter information independent of one another, and help in stricter optimal maps of radio-linked application profiles to the processor datapath.

The predictor system also needs to be modeled with the trace results obtained during the code generation and optimization phase associated with application profiling and compiler back-end construction. Investigations related to this phase of work are currently being carried out. This design methodology can help to overcome a difficulty in asynchronous processor based platform design that relates to knowing when a specific computation is done for non-deterministic event generation. The profile-derived code has to be integrated with the modified Balsa compiler framework for this purpose. The behavioral model of the predictor system is described in the Balsa language in the modified compiler framework. The behavioral model can be synthesized by usual Balsa API integration with standard EDA synthesis tools like Icarus Verilog [10]. Balsa has dedicated APIs for integration with this tool.

The simulation was run to completion for our designed predictor system model and the time of simulation noted

till the entire handshaking circuit graph was traversed. The left side of the window pane shows the values held in the buffer ports, buffer lines and communicating rails. The red line corresponds to the handshaking circuit generated in the first target machine generation. We present the buffer simulation traces for a 2-QDI circuit model for handshaking times of 50 μs and 500 μs.

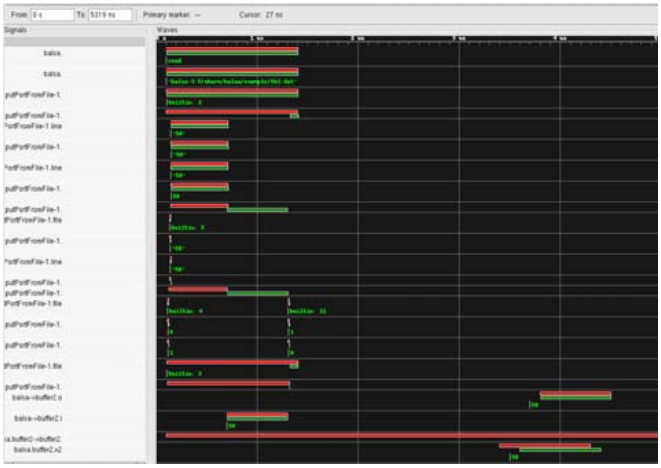

Fig. 1. Handshaking simulation traces of the predictor system in terms of QDI and STAPL circuit models – 2 QDI buffer, handshaking time 50 microseconds. Red surge in buffer 2 shows the radio task allocation set almost exclusively occupied for one type of application task.

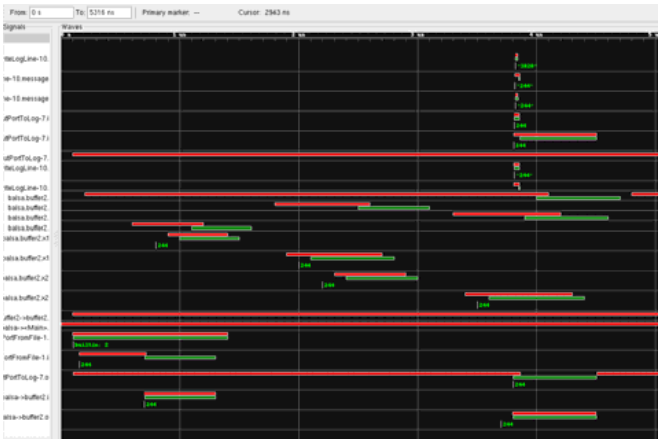

Fig. 2. Handshaking simulation traces of the predictor system in terms of QDI and STAPL circuit models – 2 QDI buffer, handshaking time 500 microseconds. Dual red surges in buffer 2 and the main module shows the radio task allocation sets being swapped for a given application profile, meaning that application task jobs are shifted internally within the sets.

VI. Evaluation of the Compiler Back-End Design

Balsa also provides a set of wrappers that can be used to map the target machines generated directly onto conventional EDA synthesis tools. This is leveraged upon to generate the target machines after the radio protocol description optimization and modified LAD arithmetic implementation in the Balsa compiler framework is integrated and built into one.

A number of simulations were carried out using the modified LAD arithmetic based compiler back-end using Balsa. The experiment was carried out by setting different values of handshaking times for two dual-rail configurations with 8 QDI buffers connected in a parallel topology with STAPL rails. The tests were designed to keep in mind non-deterministic interrupt generations. As radios and application profiles can change pretty arbitrarily in the real-world for portable devices, a particular circuit that is already processing information will need to be interrupted and configured again for processing a different set of information. As there is no operating system to channel these frequent sets of interrupts, the asynchronous datapath must wholly channel this to the processor core.

With respect to that new interrupt, a handshaking circuit configuration must now be generated as the target of computation. The handshaking circuit configuration that is currently present needs to be acted upon based on the interrupt priority. Potential deadlocks may arise owing to arbitration contention. With respect to this time of new target generation, handshaking times are negative for the targets already generated in the process. This also helps in verification of the predictor circuit system to channel the information profiles to the datapath removing arbitration issues. Fig.3 shows the FSM transition of the handshaking task graph for 8 dual-rail QDI buffers with parallel decompositions for a handshaking time of 100 ns.

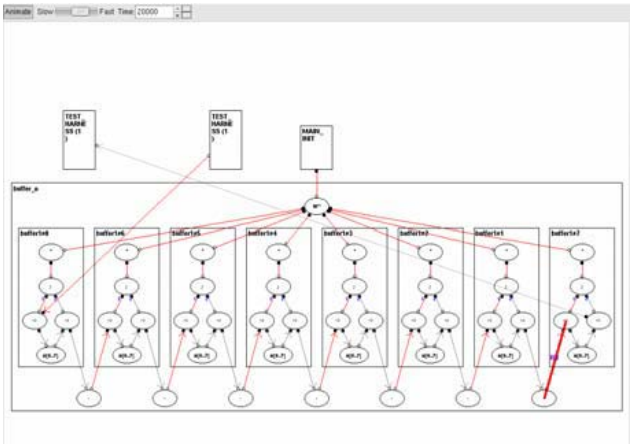

Fig.3. FSM transition of the handshaking task graph for 8 QDI buffer with STAPL rails with handshaking time = 100 ns (This means that the excitation to the second target machine generation configuration was done 100 ns after the first target machine was generated)

VII. Conclusion

High-level description language compilation on target machines involving hybrid asynchronous processor ISA is one of the recent hotbeds of focus in handset vendors owing to superior battery life prospects. This paper shows a design methodology whereby high-level description languages encompassing circuit and system-level behavior as well as radio finite state machine (FSM) executions can be fed in to generate hybrid asynchronous design style handshaking circuits. This paper discusses preliminary finds obtained during the first prototyping of hybrid asynchronous processor-based platforms targeted at handset IPs.

There is an open issue about the target machines changing during compilation (the configuration of QDI

and STAPL circuits changing based on the type and number of applications handled by the platform). Translation of such events like hardware interrupts in such cases would need to be translated to the asynchronous processor ISA in a pre-predicted manner. Dynamic target changing in the context of a single run-time program partitioning for asynchronous targets is one aspect that needs to be enhanced in the functional extensibility of the compiler back-end framework.

The other open issue is the contention in datapath timing closures due to handshaking when a large number of application tasks are involved. This may be difficult to implement using the current QDI circuit model theory that Balsa uses. Further investigations are being currently carried out in this area.

REFERENCES

[1] M. Renaudin, P. Vivet and F. Robin, "ASPRO-16: A standard cell QDI 16-bit RISC asynchronous microprocessor", in proc. of 4th. International Symposium on Asynchronous Circuits and Systems, *ASYNC 1998*, April 1998, pp. 22-32

[2] Mika Nystrom and Alain J. Martin, "Method and apparatus for an asynchronous pulse logic circuit", *United States Patent Application* # 20050007151, Ser. # 10/693543, Jan 13, 2005

[3] USC Asynchronous VLSI Design, http://jungfrau.usc.edu/new/research/current/async1

[4] Rajit Manohar, "Width-adaptive data word architectures", in proc. of Advanced Research in VLSI 2001, *ARVLSI 2001*, 14-16 March 2001, pp. 112-129

[5] V. Ekanayake, Clinton Kelly IV and R. Manohar, "An ultra-low power processor for sensor networks", in proc. of 11th. International Conference on Architectural Support for Programming Languages and Operating Systems, *ACM ASPLOS*, April 2004, pp.27-36

[6] Dipnarayan Guha and Thambipillai Srikanthan, "Reconfigurable Frame Parser Design for Multi-Radio Support on Asynchronous Microprocessor Cores", in proc. of IEEE International Conference on Computing: Theory and Applications, *ICCTA 2007*, Platinum Jubilee of the Indian Statistical Institute, March 2007, pp. 122-127

[7] Dipnarayan Guha and Thambipillai Srikanthan, "Multi-Radio support on Asynchronous Processor Cores: A Design Methodology approach for Cognitive Radios", in proc. of IEEE International Conference on Portable Information Devices, *PORTABLE 2007*, May 2007, pp. 1-4

[8] Daniel Simson, "Linear representations of partially ordered sets and vector space categories", *Gordon and Breach Science Publishers*, 1992

[9] The Balsa Asynchronous Synthesis System, University of Manchester Advanced Processor Technologies Group, http://intranet.cs.man.ac.uk/apt/projects/tools/balsa

[10] Icarus Verilog, http://www.icarus.com/eda/verilog

Dipnarayan Guha received his B.Eng. (First Class Honors) degree in Electronics and Telecommunications Engineering from Jadavpur University, Calcutta, India, in 2000 and his M.Eng degree in Electrical Engineering from Cornell University, Ithaca, New York, in 2004.

He currently holds a tenured position of Research Associate with the Center for High Performance Embedded Systems, Nanyang Technological University, Singapore. Prior to this appointment, he had worked with Sasken, Agilent Technologies and Samsung. He has published more than 15 papers in the areas of embedded systems design methodologies and data communications. He is also an active contributor to internet engineering standards and has co-authored five internet drafts on path computation that were sourced to become RFCs in the IETF PCE WG.

Mr. Guha won the British Chevening Scholarship in 2003 and an NSF student paper award in 2004.

Dr. Thambipillai Srikanthan received his B.Sc. degree (Honors) in Computer and Control Systems and his Ph.D. in System Modeling and Information Systems Engineering from Coventry University, United Kingdom, in 1980 and 1986 respectively.

He is the Director of Center for High Performance Embedded Systems and tenured Professor of Computer Engineering, Nanyang Technological University, Singapore. He has published more than 220 papers and holds three invention disclosures. His research interests include application-specific architectures for embedded systems and design methodologies for high performance embedded systems.

Dr. Srikanthan is a Corporate Member of the Institution of Electrical Engineers (MIEE) and Chartered Engineer (CEng) since 1990. He is also a Senior Member of the Institute of Electrical and Electronics Engineers (SMIEEE) since 1994. Dr. Srikanthan was awarded the Public Administration Medal (Bronze) on 2006 Singapore National Day for outstanding contributions to education in Singapore.