An LTE Uplink Receiver PHY Benchmark and Subframe-Based Power Management

Magnus Själander, Sally A. McKee
Chalmers University of Technology
Gothenburg, Sweden
{hms | mckee}@chalmers.se

Peter Brauer, David Engdal, and Andràs Vajda
Ericsson AB
Gothenburg, Sweden
{peter.brauer | david.engdal | andras.vajda}@ericsson.se

***Abstract*—With the proliferation of mobile phones and other mobile internet appliances, the application area of baseband processing continues to grow in importance. Much academic research addresses the underlying mathematics, but little has been published on the design of systems to execute baseband workloads. Most systems research is conducted within companies who go to great lengths to protect their intellectual property. We present an open-source LTE Uplink Receiver PHY benchmark with a realistic representation of the baseband processing of an LTE base station, and we demonstrate its usefulness in investigating resource management strategies to conserve power on a TILEPro64. By estimating the workload of each subframe and using these estimates to control power-gating, we reduce power consumption by more than 24% (11% on average) compared to executing the benchmark with no estimation-guided resource management. By making available a benchmark containing no proprietary algorithms, we enable a broader community to conduct research both in baseband processing and on the systems that are used to execute such workloads.**

I. Introduction

The cellular and mobile broadband market has increased tremendously over the last decade and the number of base stations is predicted to increase with 14% in 2011 [1]. The number of mobile broadband subscribers has exploded in recent years with a 93% year-on-year growth in March 2009 [2]. The worldwide mobile data bandwidth usage increased by about 30% in the second quarter of 2009 alone [3]. The Climate Group projects that by 2020 wireless networks will be responsible for 13% of the global CO_2 footprint of the information and communication technology (ICT) sector; likewise, they project that datacenters will be responsible for 18% of the ICT emissions [4]. These data underscore the need to develop power-efficient base stations whose processing capacity scales to meet ever-growing demands.

The research community has heretofore lacked a shared, freely available benchmark model of baseband processing. This prevents comparison of system designs and strategies. Given the potential market share that can be controlled by the base-station providers, companies have been strictly guarding their intellectual property. Nonetheless, in light of the rapid expansion of the number, needs, and distribution of consumers of mobile connectivity, an argument can be made for working together as a research community to meet future baseband processing requirements.

To that end, we have developed an open-source benchmark [5] that provides a realistic model of LTE (Long Term Evolution) [6] uplink receiver workloads. LTE is the next generation in radio access technology, following 3G/WCDMA and HSPA. It is optimized to deliver data rates of up to 100 Mbit/s and is currently being globally deployed. LTE delivers these high bit rates through complex baseband processing that increases the computational demand and thus the power requirements of a base station.

Our LTE benchmark is organized as a software pipeline in which modules can easily be replaced to model different algorithms. Although our benchmark's performance does not rival state-of-the-art proprietary software and platforms, it provides a reasonable baseline for comparing hardware designs or resource management strategies — precisely because it contains no proprietary algorithms. In short, it realistically captures the dynamic behavior of an LTE baseband uplink as viewed by the base station.

The demand for wireless communication varies over time with periods of peak loads (rush hours) and periods of low loads (late nights). Adapting the available computational resources to variations in load can save power. This can reduce the operational cost, which is a large portion of the base station total cost-of-ownership.

We use our LTE Uplink Receiver PHY benchmark to develop a subframe-based workload estimator, the results of which we use to adapt the number of active cores on a Tilera TILEPro64 processor. Our measurements show that dynamic power can be reduced by 4% compared to no estimation-based deactivation of cores. Further, we show that the potential to power gate cores can reduce power by more than 24% (11% on average). These results are encouraging in that they are "unoptimized" — we do not yet take full advantage of low-load scenarios where most of the computational resources of a base stations could potentially be turned off.

II. LTE Overview

We first present a short overview of the resource allocation and signal processing in an LTE baseband uplink receiver. For more detailed information on the LTE baseband standard, refer to the 3GPP specification series 36 [6], [7], [8].

A. Resource Allocation

To explain the operation of an uplink of an LTE base station, we first address how the frequency spectrum and time are allocated to the LTE transmitters/users (called UEs) that communicate via that base station.

Each base station is allocated a certain frequency band (1.25 MHz to 20 MHz). This frequency band is divided into 15 kHz sub-carriers. Time is divided into frames lasting 10 milliseconds, where each frame is further subdivided into 10 subframes lasting one millisecond. A subframe is divided into two slots, each with seven SC-FDMA symbols. The symbols are arranged such that three data symbols are transmitted, followed by one reference symbol used for channel estimation, followed by three more data symbols [7]. The smallest schedulable unit is a physical resource block (PRB), which is a grouping of twelve sub-carriers that lasts for the duration of one slot. An illustration of the frequency and time domains is shown in Fig. 1.

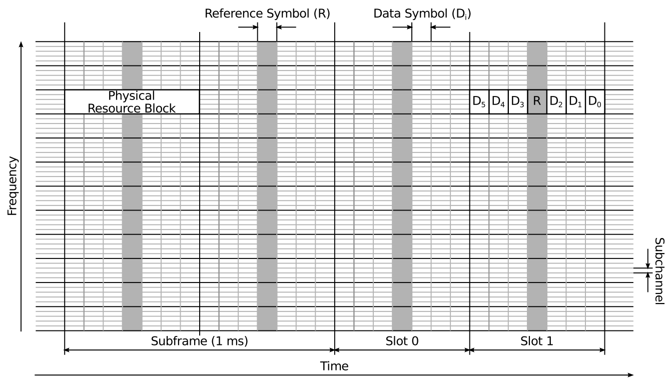


Fig. 1. Frequency and time organization for an LTE base station.

A base station might service several hundreds of users. For a single subframe, the base station can typically schedule a maximum of ten users. Each scheduled user is allocated its own set of PRBs to be transmitted during that subframe.

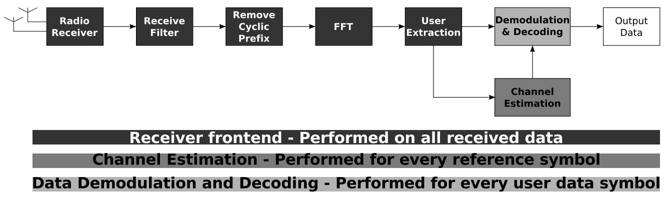
B. Channel Modulation and MIMO Systems

Various coding and modulation schemes can be used, depending on the signal quality between the transmitter and receiver. When noise and interference are low, a higher-order modulation scheme can be employed (e.g., 16-QAM or 64-QAM [9]) to achieve higher data rates compared to when noise and interference are high.

Another technique for increasing the data rate is multiple-input and multiple-output (MIMO) [10]. This technique employs multiple input and output radio channels to improve the spectral efficiency and link reliability through higher tolerance for fading. Spatial multiplexing, a transmission technique that can be used in conjunction with MIMO [11], allows transmission of several independently encoded data signals via multiple transmit antennas. The independently encoded data signals are called *streams* or *layers*. The latest standards for LTE Advance include support for up to four transmission layers in the uplink [12].

C. LTE Uplink Receiver Baseband Processing

The front-end of the LTE base station receiver includes several components: the radio receiver, receive filter, cyclic prefix removal, and fast Fourier transform (FFT), as shown in Fig. 2. Once the FFT is computed, the PRBs of one subframe for one user can be processed as a chain of signal processing kernels that can be modeled as shown in Fig. 3.



Radio Receiver
Receive Filter
Remove Cyclic Prefix
FFT
User Extraction
Demodulation & Decoding
Output Data
Channel Estimation
Receiver frontend - Performed on all received data
Channel Estimation - Performed for every reference symbol
Data Demodulation and Decoding - Performed for every user data symbol

Fig. 2. Illustration of a typical model of an LTE receiver.

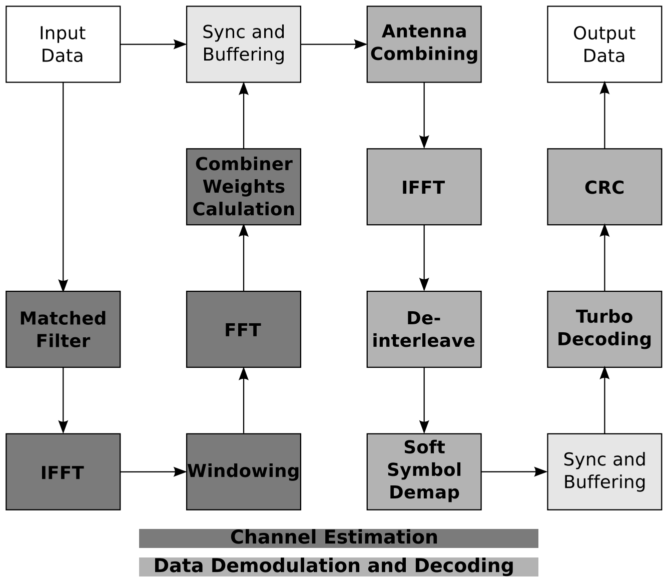


Fig. 3. Illustration of a typical model for processing a user.

Channel estimation is typically performed for a slot before the data are demodulated and decoded for the user. This requires that the first three symbols be buffered until the reference symbol is received and processed. Channel estimation first applies a matched filter that multiplies the received reference symbol (which has been distorted by the channel) with the defined reference symbol. An inverse FFT (IFFT) follows the matched filter, transforming the data back to the time domain, where a window is used to extract part of the time domain samples. An FFT then transforms these samples back to the frequency domain. For a MIMO system, channel estimation must be performed for each receive antenna and layer, the results of which are used to calculate combiner weights to be used in demodulation.

The combiner weights are used to merge the data from multiple antennas and adjust for channel conditions. An IFFT then transforms the data back to the time domain. In that domain, the data are deinterleaved and soft symbol demapping is performed. A turbo decoder consumes the soft demapped symbols, and the results undergo a cyclic redundancy check.

III. Parallelism in the LTE Uplink Receiver

Meeting the processing requirements of one new subframe per millisecond requires parallelization. Fortunately, baseband processing can be parallelized at multiple stages. The most obvious approach processes each user separately (see Fig. 4). A base station can typically handle ten users in a single subframe. However, the amount of processing required by different users can vary significantly (e.g., one user might use voice over IP at kbits/sec, while another might upload files at Mbits/sec). If parallelization is only performed across users, the workload for different cores becomes uneven.

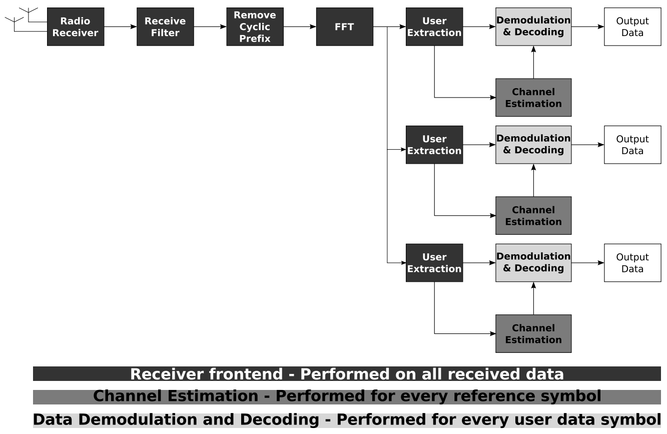

Radio Receiver
Receive Filter
Remove Cyclic Prefix
FFT
User Extraction
Demodulation & Decoding
Output Data
Channel Estimation
Receiver frontend - Performed on all received data
Channel Estimation - Performed for every reference symbol
Data Demodulation and Decoding - Performed for every user data symbol

Fig. 4. Illustration of parallelization across three different users.

User processing can also be parallelized: *i)* For channel estimation, the matched filter, IFFT, windowing, and FFT kernels are processed separately on data from each of the receiver antennas and layers. For a four-antenna receiver and the maximum four layers [12], the reference symbols can be processed by up to 16 (four antennas $\times$ four layers) tasks. The combiner-weight computation considers all the receiver channels and layers, and is therefore not easily parallelized. *ii)* For data demodulation and decoding, antenna combining and FFT can be performed on each separate symbol and layer. With six symbols in each slot and a maximum of four layers, the data can be processed by up to 24 individual tasks (six symbols $\times$ four layers). The remaining processing is performed across all data, and is thus not easily parallelized. Fig. 5 shows a schematic illustration of the parallelization of channel estimation and data demodulation and decoding.

Further algorithmic-level parallelization could, for instance, spread the computation of an FFT across multiple processing units. This type of parallelization is orthogonal to what we consider here, and employing it in addition to our schemes would enable further scaling for higher performance.

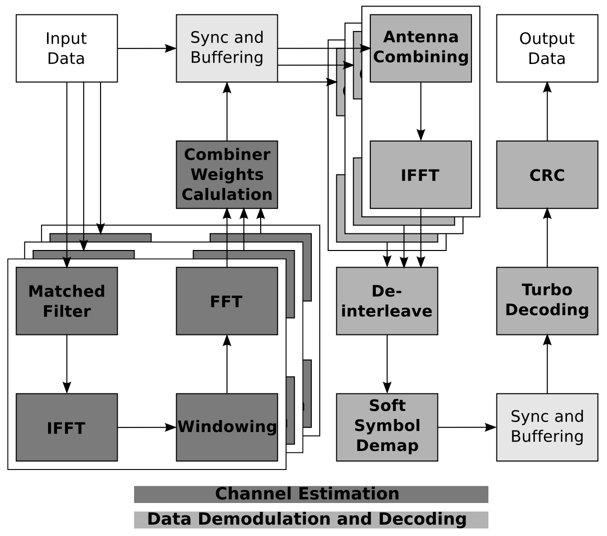


Fig. 5. Illustration showing parallelization of the channel estimation and data demodulation and decoding.

IV. LTE Benchmark Implementation

We created the LTE Uplink Receiver PHY benchmark to capture the dynamic behavior of an LTE baseband uplink of a base station. We exclude the computations of the frontend (see Fig. 2) from our benchmark, since the frontend is statically defined and performed on all data received.

The dynamic behavior comes from the allocation of physical resource blocks (PRBs) to users, the number of users allowed to transmit data, and the quality of each user's radio channel. The channel quality determines the modulations and the number of allocated layers. The base station allocates PRBs to users for each subframe. The duration of a subframe is one millisecond and is therefore the rate at which the workload changes. The following input parameters define the workload for a subframe:

- number of users;
- number of PRBs allocated to each user;
- number of layers used for each user; and
- modulation technique used for each user.

A. Parallelization Framework

We have created two versions of our LTE benchmark: a serial version that processes a subframe sequentially and a parallel version based on POSIX threads (Pthreads) [13]. We implemented the serial version as a reference to verify parallelized versions of the benchmark.

We implemented the LTE benchmark using Pthreads, a well defined standard [13] supported by many operating systems and architectures. The Pthreads version of the LTE benchmark implements task creation, scheduling, and resource management, allowing for complete control of the benchmark's behavior and interactions with the underlying architecture. This is the default version of the LTE benchmark, and it is the version on which this article focuses.

B. Subframe Generation and Dispatch

The LTE benchmark consists of one maintenance thread and a configurable number of worker threads. The maintenance thread produces the inputs for each subframe and dispatches a subframe every millisecond. In practice, the rate at which subframes are dispatched is configurable; this allows the benchmark to run on hardware that cannot sustain a rate of one subframe per millisecond. During initialization, the maintenance thread creates a defined number of worker threads and initializes all required data structures. After initialization, the maintenance thread enters a loop in which input data and parameters for a subframe are created and dispatched every DELTA milliseconds (where DELTA is configurable).

1) Subframe Input Data: At benchmark initialization, input data sets are created for multiple subframes and then reused across all dispatched subframes. This avoids the overhead of creating new data for each subframe while assuring that all subframes being processed in parallel have unique data. The number of unique input data subframes to generate is configurable (with ten as the default).

2) Subframe Input Parameters: The input parameters for a subframe depend on the usage scenario modeled, which, in turn, depend on two functions that create the input parameters. The first function, *init_parameter_model(parameter_model *pmodel)*, initializes the necessary variables for a model, and the second function, *user_parameters *uplink_parameters(parameter_model *pmodel)*, returns the number of users along with their parameters and input data for one subframe.

3) Subframe Dispatch: Given the input data and parameters generated for a subframe, the maintenance thread waits for a signal alarm to be triggered every DELTA milliseconds. Immediately after a signal alarm is triggered, the subframe's users are written to a global queue for processing, after which the maintenance thread starts creating the input parameters for the next subframes and waits for the next signal.

C. Subframe Processing

Subframes are processed by a parameterized number of worker threads. Task scheduling employs work stealing: each worker thread has a local task queue, and if no work exists in its own queue, it tries to steal work from another worker thread. Work stealing has been shown to achieve good load balance across multiple cores, to scale well with the number of cores, and to achieve efficient performance where the number of cores can change dynamically [14], [15]. Before a worker thread tries to steal work from another thread, it first checks the global user queue to ensure that a new subframe has not been dispatched. If a user exists in the queue, an idle worker thread will dequeue one user and start processing its PRBs (this thread is now considered a user thread).

1) Channel Estimation: Subframe processing starts with the user thread creating a number of tasks equal to the number of receive antennas multiplied by the number of layers used for the current subframe. The tasks are placed on the local task queue, and the user thread then processes them until the queue is empty. Other idle worker threads, can steal tasks to help perform the channel estimation. Once the local task queue is empty, the user thread waits until the results from all tasks become available. This is necessary because other worker threads might still be processing some of the channel-estimation tasks. When all data are processed, the user thread performs the combiner-weight calculations used for antenna combining and data processing.

2) Data Demodulation and Decoding: In the next processing stage, antenna combining and inverse FFT can be performed independently for each data symbol and layer. To do this, the user thread creates a number of tasks equivalent to the number of symbols multiplied by the number of layers. These tasks are placed on the local queue, and the user thread processes them until the queue is empty. Idle threads can steal work to help with data processing. Data from both slots are required for processing to proceed. When all processing is completed for the two slots, the user thread performs the remainder of the subframe processing, i.e., interleaving, soft demapping, turbo decoding, and cyclic redundancy checking (CRC). The computational intensive turbo decoding is commonly executed on dedicated hardware, and thus we omit it from our benchmark. The call to perform turbo decoding simply passes the data through.

D. Verifying the LTE Benchmark

We validate the parallelized uplink benchmark, by comparing the results to those of the serial implementation. The serial version processes a predetermined sequence of subframes, recording and storing the results from each subframe. By processing the same sequence of subframes in the parallel versions of the benchmark, results from each subframe can be compared against the serial version's data. This can be used to verify that the computation is consistent across different architectures, as well.

V. Evaluation Setup

We first describe the input parameter model and then the hardware platform used to generate our results.

A. Subframe Input Parameter Model

A base station might service hundreds of users, each with their own requirements and channel conditions, while only being able to schedule approximately ten users per subframe. The selection of ten users out of a larger set makes the parameters of a subframe look rather random. It is therefore realistic to approximate the subframe parameters with randomized values. For lower load scenarios we also use randomized values for simplicity of the parameter model.

For this evaluation we created an input parameter model that varies the workload over time. A random number of users is created, with each user having a random number of PRBs. Fig. 6 presents pseudocode for the algorithm and Fig. 7 shows the distribution of users for 68,000 subframes created with the algorithm. To make the graph clearer, we only plot every 25th subframe. The figure shows that the number of users varies constantly and rapidly. For each of these subframes

Fig. 8 shows the number of PRBs that are allocated. The figure shows for each subframe the total number of PRBs allocated to a subframe and the maximum and minimum number of PRBs allocated to a single user. The maximum number of PRBs allocated to a user varies between 20 and 190, while the minimum number of PRBs varies between two (a user has to have at least two PRBs to be scheduled for a subframe) and 100. Fig. 7 and Fig. 8 show that the distributions in both number of users and number of PRBs are large.

```
 1: const MAX_PRB = 200
 2: const MAX_USERS = 10
 3: nmbPRB = 200
 4: nmbUsers = 0
 5: while nmbUsers<MAX_USERS and nmbPRB>0 do
 6:   userPRB = MAX_PRB $\times$ random()
 7:   # Create a larger spread in number of PRBs
 8:   distribution = random()
 9:   if distribution < 0.4 then
10:     userPRB = userPRB/8
11:   else if distribution < 0.6 then
12:     userPRB = userPRB/4
13:   else if distribution < 0.9 then
14:     userPRB = userPRB/2
15:   end if
16:   newUser(userPRB)
17:   nmbUsers++
18: end while
```

Fig. 6. Pseudocode for user input parameter generation, where `random()` generates a random value between 0 and 1.

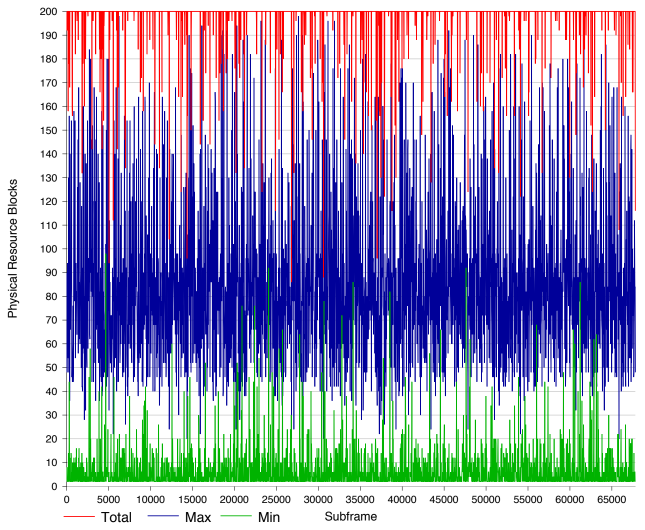


Fig. 8. The total number of physical resource blocks allocated to all users and the maximum and minimum number of physical resource blocks allocated to a single user for every 25th subframe.

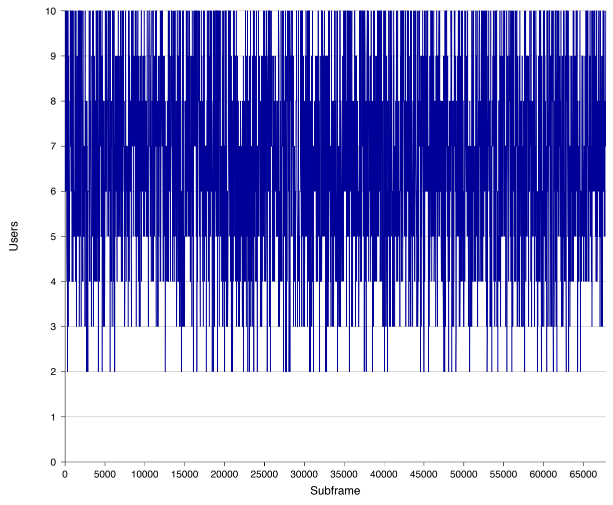


Fig. 7. Number of users for every 25th subframe.

The input parameter model varies the workload by increasing the probability that a user has more layers and a more complex modulation according to the algorithm in Fig. 10. The probability (`prob` in Fig. 10) is increased/decreased every 200th subframe from a probability of 0.6% to a probability of 100%. The probability is linearly increased over the first 34,000 subframes, upon which maximum workload is reached, with every user having four layers and 64-QAM as modulation. After reaching maximum workload, the probability is linearly decreased to 0.6% over the next 34,000 subframes. The maximum and minimum number of layers that users have across the 68,000 subframes are shown in Fig. 9.

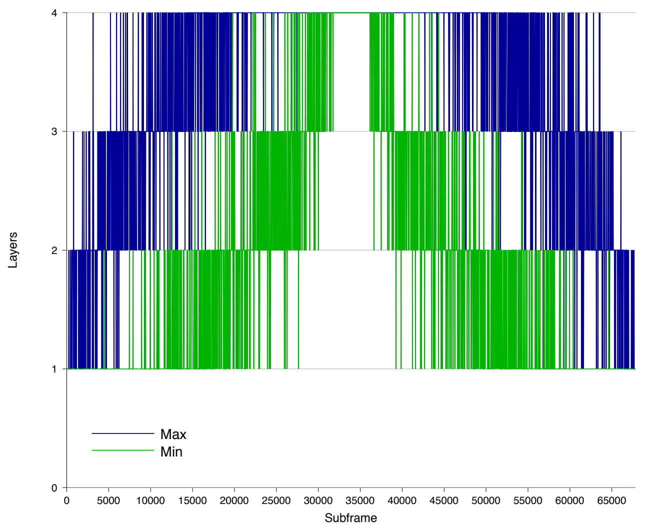


Fig. 9. The maximum and minimum number of layers that users have for every 25th subframe.

The complete pseudocode algorithm for the input parameter model is given by replacing line 16 in Fig. 6 with the pseudocode in Fig. 10. The input parameter model does not try to model a realistic use case, but rather tries to effect a high variation with rapid changes in the number of users, PRBs, and layers and rapid change of modulation while still achieving a continuous trend in increasing/decreasing the total workload over a number of subframes.

```
prob = current_probability(subframe)
userLayer = 1
if prob > rand() then
  userLayer++
end if
if prob > rand() then
  userLayer++
end if
if prob > rand() then
  userLayer++
end if
userMod = MOD_QPSK
if prob > rand() then
  userMod = MOD_16QAM
  if prob > rand() then
    userMod = MOD_64QAM
  end if
end if
newUser(userPRB, userLayer, userMod)
```

Fig. 10. Pseudocode for layer and modulation input parameter generation.

B. Hardware Platform

We evaluate the LTE benchmark on the Tilera TILEPro64 processor [16]. We choose the TILEPro64 because it is highly parallel and is optimized for low power, thus it is suitably representative of the type of (proprietary) processors found in LTE base stations.

The TILEPro64 processor consists of 64 identical 32-bit VLIW cores that are connected through an on-chip mesh network. The VLIW cores consist of: *i)* a three-stage pipeline with an issue width of three instructions, *ii)* a private 16 kB L1 instruction cache, and *iii)* a private 8kB L1 data cache. Each core also has a 64 kB L2 cache that is shared with all other cores, thus creating a virtual L3 cache of 4 MB.

Each worker thread of the LTE benchmark is mapped to a single core of the TILEPro64. One of the TILEPro64's cores is dedicated to drivers, and another is used for the maintenance thread; the 62 remaining cores are available for running worker threads. When executing the LTE benchmark with maximum workload and 62 worker threads, a new subframe can be received every fifth millisecond. This rate results from the general-purpose nature of the TILEPro architecture: it contains no application-specific optimizations for baseband processing, as would dedicated base-station hardware. We run the same, generic C code on the TILEPro64 as on x86 architectures with Linux as the operating system. In conventional base stations a tailored real-time operating system is used.

To study power dissipation, we measure the voltage drop across two resistors that are used to balance the load between the two phases of the buck converter supplying power to the TILEPro64 chip. Our National Instruments USB-6210 [17] data acquisitioning unit can sample the two voltages at a minimum period of eight microseconds. We calculate the current by dividing the two voltages by the value of the precision resistors over which the voltage drops are measured and then adding them together. The current varies rapidly, so we compute the root mean square (RMS) value of the current for every 100 milliseconds. The supply voltage for the TILEPro64 chip is 1.0 V. The measured current is therefore equal to the power dissipated, and we use power (W) for the measurements presented in the following sections.

The TILEPro64 has an assembly "nap" instruction that clock-gates the core on which it is executed. This instruction can therefore be used to affect the dynamic power. There is no easy way to reactivate a "napping" core; a core therefore periodically wakes up to see if its status has changed, and if not, it goes back to sleep.

The base power when the TILEPro64 chip performs no work is 14W. We measure this by generating 63 threads and mapping each to its own core of the TILEPro64 before the "nap" instruction is executed. This effectively puts 63 of the cores to sleep and gives us a base-power of 14W. This is also the power measured when no applications are executed.

VI. Power-Aware Resource Management

LTE uplink processing is predictable in that a new subframe is received every millisecond. Responsiveness requirements limit the time permitted to process a subframe. A base station therefore processes no more than two to three subframes concurrently. If the workload of a subframe can be estimated accurately, it becomes possible to predict the amount of computational resources required. Such prediction must be done for each subframe, and the amount of required resources potentially changes with every received subframe.

A. Subframe Workload Estimation

The work of a subframe is characterized by its input parameters (see Sec. IV), and it is the responsibility of the base station to allocate resources to users. The input parameters of a subframe are therefore known before the subframe is received. Knowing the correlation between workload and subframe input parameters makes it possible to predict a subframe's workload. To determine if such a correlation exists, we created an input parameter model that produces a single user for which the number of physical resource blocks (PRBs), layers, and the modulation are all varied. The subframe lasts for mere milliseconds, and the fact that multiple subframes (two to three) are processed in parallel makes it difficult to measure the workload of a single subframe. To circumvent this problem, the parameter model creates a steady state with the same user parameter configuration (fixed number of PRBs, layers, and modulation) for a duration of ten seconds. The workload is measured by inserting *get_cycle_count()* function calls before and after every part of the benchmark that performs useful processing. The number of useful cycles for each part of the benchmark is calculated by computing the delta of the two *get_cycle_count()* function calls (Eq. 1). Summing all the compute cycles and dividing by the total number of cycles over a period of time gives the activity of the system (Eq. 2). In our case, we chose to compute the activity during a period of one second. This results in ten activity measurements for every input parameter configuration.

$$computecycles = cycle_{end} - cycle_{start} \quad (1)$$

$$activity = \frac{\sum computecycles_i}{totalcycles} \quad (2)$$

Fig. 11 shows the results when the number of PRBs is increased from two to 200 in steps of two for different numbers of layers and different modulations. The figure shows a clear correlation between subframe input parameters and the activity of the system: activity has a linear relation (k) to the number of PRBs for a given set of layers (L) and modulation (M) (Eq. 3). The design of our subframe workload predictor is based on these results (Fig. 11). Given a set of equations for estimating the activity for all layer and modulation configurations, the total workload of a subframe can be modeled as the sum of the workload for each user of the subframe (Eq. 4).

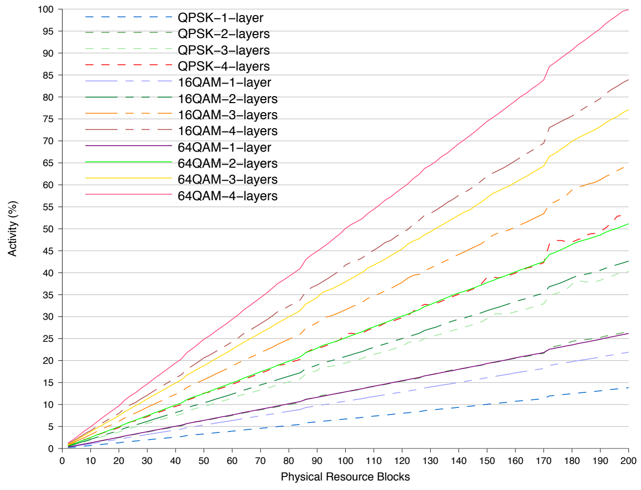


Fig. 11. Correlation between subframe input parameters for a single user and the workload (Activity) of 62 worker threads on a TILEPro64.

$$estimated\_user\_activity = PRBs \times k_{LM} \quad (3)$$

$$estimated\_activity = \sum estimated\_user\_activity_i \quad (4)$$

The estimated workload given by Eq. 4 is based on simplifications made to measure the activity as a result of subframe input parameters and the assumption that a user's workload is not affected by the workloads of other users. To verify that our workload estimates are valid, we use the input parameter model described in Sec. V-A to calculate the average estimated activity during one second and then compare it with measured activity for the same period. Fig. 12 shows the result.

Since the workload of a single subframe is practically impossible to measure, we measure the average workload over one second, instead. One second is also the period at which the input parameter model changes the probability for increased/decreased workloads. The figure shows that the estimated workload tracks the measured workload well. The maximum error is an underestimation of 5.4%, and the average error is only 1.2%.

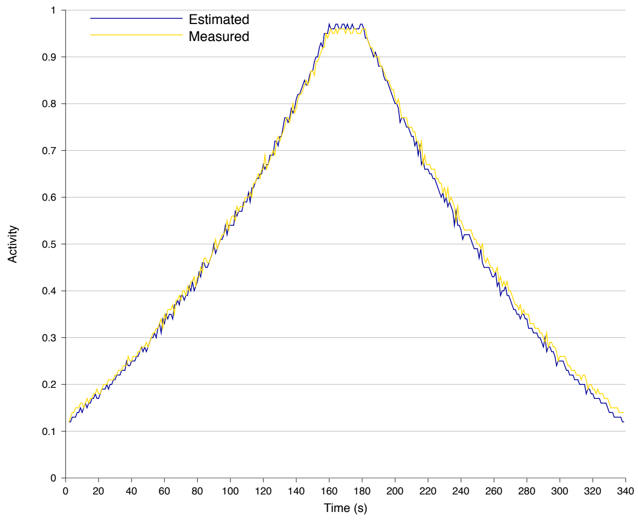


Fig. 12. Measured and estimated workload averaged over 200 subframes (1 second) for a total of 68,000 subframes (340 seconds) executed by 62 worker threads on a TILEPro64.

B. Dynamic Power Reduction

Armed with the workload estimation model described in Sec. VI-A, it is now possible to adjust the number of active cores in the system to the actual workload of the subframes being computed. To provide some margin of error in the estimation, the system is over-provisioned with two cores. The number of active cores is calculated for each subframe according to Eq. 5.

$$active\_cores = estimated\_activity \times max\_cores + 2 \quad (5)$$

Applying Eq. 5 to the subframes generated by the input parameter model described in Sec. V-A yields an estimate of the number of active cores. Fig. 13 shows the estimated number of active cores (to improve readability, we again show results for every 25th subframe). The number of active cores changes rapidly throughout the duration of the 68,000 generated subframes.

Worker threads deemed not to be required (according to Eq. 5) are deactivated with "nap" instructions. Fig. 14 shows the resulting power savings. The figure shows the power measured on the TILEPro64 as the LTE benchmark is executed with (NAP) and without (NONAP) deactivating cores. It shows that dynamically adapting the number of active cores reduces power. The difference is largest when the workload is low (6-7W), resulting in a reduction of more than 25%. For the maximum workload NAP also achieves a lower maximum power of almost 1W, or 3%. This is because the average power for NONAP is 18‘% higher (25W) than for NAP (20.5W). The higher average power raises the TILEPro64's temperature, which increases power. The effects of increased temperature are also shown by the higher power values (on the right side of the graph) after the maximum load has been reached.

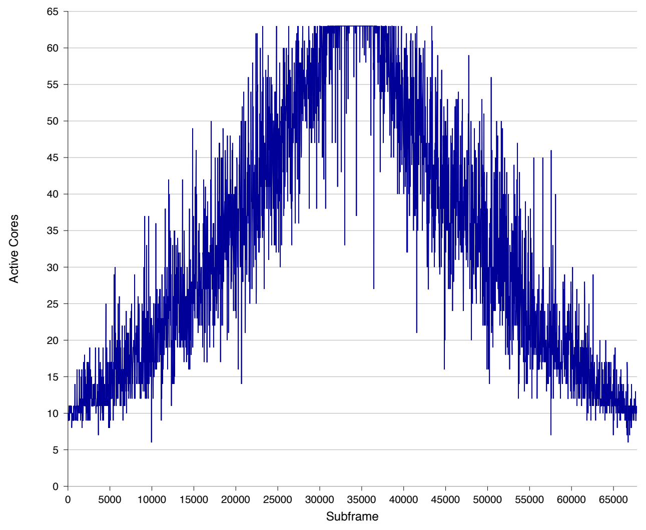


Fig. 13. Estimated number of active cores for every 25th subframe.

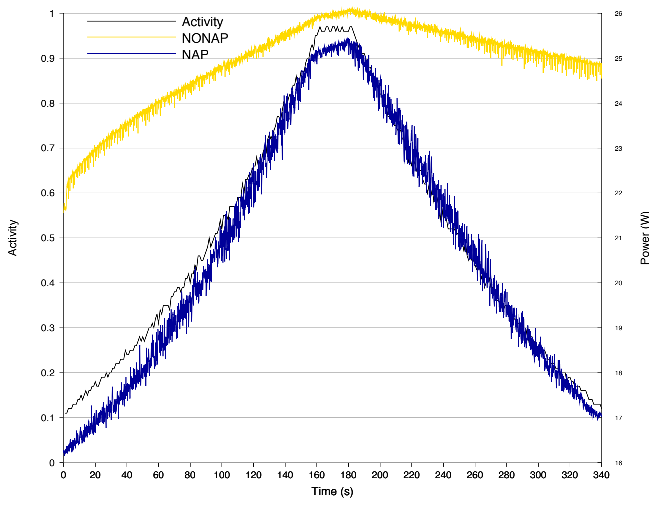


Fig. 14. Measured power as a result of executing the LTE benchmark with (NAP) and without (NONAP) deactivation of cores based on workload estimation.

A logical alternative to deactivate cores based on estimated workload (proactive) is to deactivate cores when there is no work to be performed (reactive). By deactivating threads that cannot find any tasks to steal (IDLE) we reduce power (20.7W) by close to what the estimated deactivation achieves (NAP), as shown in Fig. 15. For low workloads, deactivating cores based on estimated workload achieves lower power. This is a result of a majority of the cores being deactivated, and thus, these cores do not periodically look for work to perform. This periodical check, which is required in a reactive system, causes overheads that result in a higher power. On average, the power is increased by 1%.

Combining the two techniques (proactive and reactive) achieves even better results. By deactivating cores based on estimated workload, the overhead of looking for non-existent work can be avoided. The remaining active cores will not be utilized to 100%, as there are fluctuations in how much the subframe processing can be parallelized. Deactivating cores that have no work to perform reduces power further, as shown by NAP+IDLE in Fig. 15. The combined techniques yield an average power of 19.9W, which is an additional reduction of 3% compared to only using NAP or a 20% reduction compared to no core deactivation.

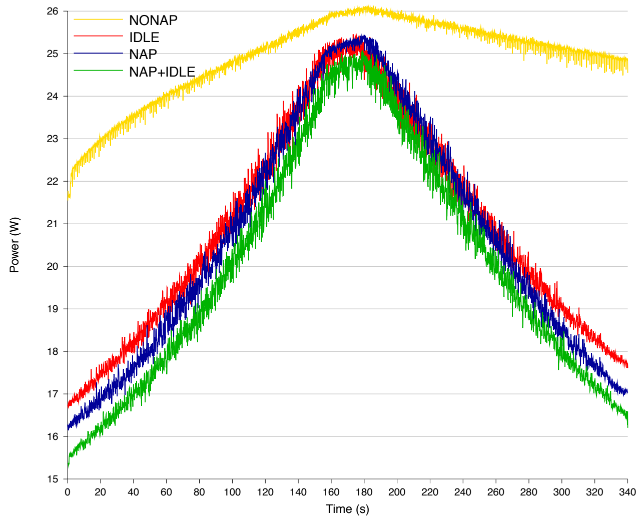


Fig. 15. Measured power as a result of executing the LTE benchmark when cores are not deactivated (NONAP), deactivated based on estimated workload (NAP), deactivated when there is no work to perform (IDLE), or when combining the two techniques of deactivating cores based on estimated workload and when there are no work to perform (NAP+IDLE).

TABLE I
AVERAGE POWER DISSIPATION WHEN NOT INCLUDING BASE POWER

Technique	Power (W)	Reduction
NONAP	11	0%
IDLE	6.7	39%
NAP	6.5	41%
NAP+IDLE	5.9	46%

Table I shows the average power for the four different use cases when the base power (see Sec. V-B) is subtracted from the total power. The base power cannot be affected by purely using clock gating, and as such is a constant factor of 14W. As the table shows, clock gating in any form is important for reducing the dynamic power of the cores. It also shows that by estimating the workload the dynamic power is reduced by an additional 7%, on average.

C. Static Power Reduction

Clock gating is an efficient method for reducing dynamic power. One property of clock gating is its responsiveness: a circuit can be clock gated on a cycle-by-cycle basis. This property makes it possible to use reactive techniques in which the number of active cores can quickly be adjusted to the system workload.

Techniques for reducing static power do not share this property, and it generally takes time to power on or off a core. The act of turning a core on or off also incurs an overhead, due to large power cut-off transistors being switched. These properties put limits on when a decision for turning cores on and off is made and how often a core is power gated. If the workload can be estimated ahead of time, that provides time for turning on cores to meet the computational demand of increasing workloads and for turning off cores as soon as the computational demand decreases. A power-aware resource manager can exploit that the LTE base station knows ahead of time what the workload will be for a particular subframe. The schedule for a subframe is known at least two milliseconds before processing begins, providing ample time to adjust the number of powered-on cores.

The hardware platform we use for executing the LTE benchmark does not provide power gating for cores. We therefore rely on an analytical model to estimate reductions in static power. Separate power domains (power grids) are required for the circuits to be power gated. Power gating of a single core is possible, but for this work we assume that cores are managed in groups of eight. For a 64-core chip this requires eight power domains, which is a reasonable number for a chip of this complexity.

The number of powered-on cores for a subframe is estimated by discretizing the estimated number of active cores:

$$active = \lceil active\_cores/8 \rceil \times 8 \tag{6}$$

To assure that enough cores are powered on and to reduce the number of cores being turned on and off, we take the maximum number of active cores across five consecutive subframes. Input parameters are known two subframes in advance, and a maximum of three subframes are concurrently computed in the system. The number of powered-on cores for each duration of a subframe can be calculated according to Eq. 7, where $active_i$ represents the subframe that has been received and is about to be processed.

$$\begin{aligned} powered = max(&active_{i+2}, active_{i+1}, \\ &active_i, active_{i-1}, active_{i-2}) \end{aligned} \tag{7}$$

If we assume that 25% of the TILEPro64's 14W base power (3.5W) is due to the 64 idle cores, the static power of each core is 55 mW. We also assume that turning on or off a core dissipates 15 mW of additional power (OH) for the duration of a subframe (Eq. 8). The total power savings for each subframe can then be calculated according to Eq. 9.

$$OH = |powered_i - powered_{i-1}| \times 0.015 \tag{8}$$

$$power\_saving = (64 - powered_i) \times 0.055 - OH \tag{9}$$

Fig. 16 shows the result if power gating were used to power off unneeded cores. These data are calculated by subtracting the savings given by Eq. 9 from the NAP+IDLE case. The average power is 18.5W, which is a reduction of 1.4W (7%) compared to when applying no power gating. For low-workload scenarios the reduction in power is 3W or 19% compared to the best achieved dynamic power management. Compared to the case where workload estimates are not used (IDLE), the power reduction is more than 4W (>24%) for low-load cases.

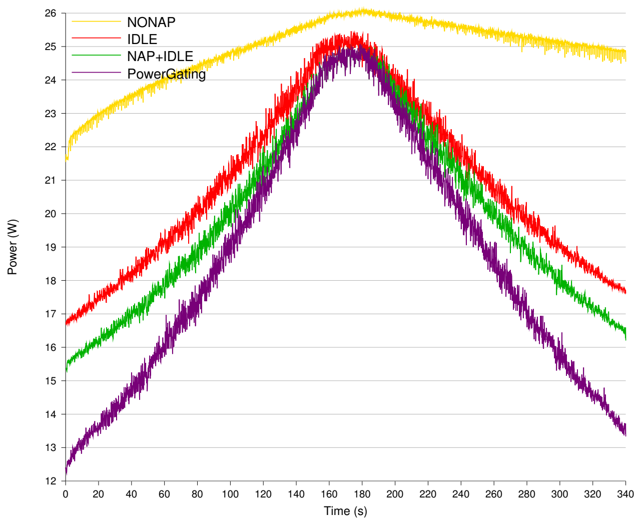


Fig. 16. Estimated power reduction when applying power gating based on estimated number of active cores.

TABLE II
AVERAGE TOTAL POWER DISSIPATION

Technique	Power (W)	Relative NONAP	Relative IDLE
NONAP	25	0%	+21%
IDLE	20.7	-17%	0%
NAP	20.5	-18%	-1%
NAP+IDLE	19.9	-22%	-4%
PowerGating	18.5	-26%	-11%

Table II shows the average power and the improvements that can be achieved by actively managing resources (Relative NONAP). The table also shows the improvements that our proactive workload estimation technique provides over reactive resource management (Relative IDLE). These data is for a use case where the average workload is 50% (see Fig. 12). A typical workload for base stations is 25%, for which our workload estimation method performs even better.

VII. Related Work

We know of no other open-source benchmark that models an LTE uplink receiver, but several existing benchmarks model portions of the LTE processing.

The BDTi OFDM receiver benchmark^{TM} [18] is a multi-channel design for evaluating multi-core and other high-performance processing engines. Public information about this benchmark is limited, but the benchmark seems to target a single user system. This is in contrast to a mobile base station where multiple users are serviced in parallel. This benchmark requires licenses for use.

MiBench [19] is a freely available embedded benchmark suite that includes GSM related processing. The benchmark implements voice encoding and decoding and uses a combination of time and frequency division multiple access (TDMA/FDMA). The inputs for this benchmark are speech samples, and thus, the computational effort required for this benchmark is low.

The LTE Uplink Receiver PHY benchmark relies on work stealing to distribute the workload across a multi-core processor. Work stealing has been shown to achieve good load balance across multiple cores, to scale well with the number of cores, and to achieve efficient performance in an environment where the number of cores changes at runtime [14], [15].

One of the most important steps in dynamic power management is to predict future load and adapt available resources accordingly to achieve acceptable performance at low power dissipation. Choi *et al.* present a technique for setting the voltage and frequency of a dynamic voltage and frequency scaling (DVFS) system based on frame-based prediction for an MPEG decoder [20]. Like us, they take advantage of application-specific knowledge to estimate the workload of an application based on its inputs. In their case, they use frames in MPEG decoding, while we use subframes in LTE baseband processing. We use our workload estimation for clock gating and show the potential when power gating cores, but we could also use it in combination with DVFS to create further power management opportunities.

VIII. Conclusions

We present an open-source LTE Uplink Receiver PHY benchmark that realistically represents the baseband processing of an LTE uplink. Using this benchmark, we show that it is possible to estimate the workload of a subframe and that this estimate can be used for power-aware resource management. We show that if power gating were available, the power of a TILEPro64 processor executing the LTE benchmark could be reduce by more than 24% (11% on average) compared to the best achieved power without using workload estimation for resource management.

The presented results are for an input parameter model with an average workload of 50% and with a minimum activity above 10% (see Fig. 12). This is an overly pessimistic use case, since most base stations have an average load of about 25% and have long periods where the load is much lower (e.g., nights). The input parameter model is created to stress the workload estimation and resource management and not favor our proposed technique. Our technique would show even greater benefits for a more realistic use case.

References

[1] ABI Research, "Wireless Infrastructure Market Data," Oct. 2011.

[2] Informa Telecoms & Media, "World Cellular Data Metrics," 2009.

[3] Allot Communications, "Global Mobile Broadband Traffic Report," July 2009.

[4] "SMART 2020 : Enabling the Low Carbon Economy in the Information Age," *GeSI's Activity Report, The Climate Group on behalf of the Global eSustainability Initiative (GeSI)*, June 2008. [Online]. Available: http://www.mendeley.com/research/smart-2020-enabling-the-low-carbon-economy-in-the-information-age/

[5] "LTE Uplink Receiver PHY Benchmark," 2011, http://sourceforge.net/projects/lte-benchmark/.

[6] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) Radio Transmission and Reception," 3rd Generation Partnership Project (3GPP), TS 36.104, Sept. 2008, http://www.3gpp.org/ftp/Specs/html-info/36104.htm.

[7] ——, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Channels and Modulation," 3rd Generation Partnership Project (3GPP), TS 36.211, Mar. 2010, http://www.3gpp.org/ftp/Specs/html-info/36211.htm.

[8] ——, "Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding," 3rd Generation Partnership Project (3GPP), TS 36.212, Mar. 2010, http://www.3gpp.org/ftp/Specs/html-info/36212.htm.

[9] L. Hanzo, S. X. Ng, T. Keller, and W. T. Webb, *Quadrature Amplitude Modulation: From Basics to Adaptive Trellis-Coded, Turbo-Equalised and Space-Time Coded OFDM, CDMA and MC-CDMA Systems*. Wiley-IEEE Press, Sept. 2004, iSBN: 978-0-470-09468-6.

[10] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge University Press, 2003, ISBN: 0-5218-2615-2.

[11] G. J. Foschini, "Layered Space-Time Architecture for Wireless Communication in a Fading Environment When Using Multi-Element Antennas," *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41–59, 1996.

[12] C. S. Park, Y.-P. E. Wang, G. Jöngren, and D. Hammarwall, "Evolution of Uplink MIMO for LTE-Advanced," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 112–121, Feb. 2011.

[13] POSIX.1c, *Threads Extensions*, ser. Information technology—Portable Operating System Interface (POSIX). IEEE Computer Society, 1995, IEEE Std 1003.1c-1995.

[14] R. D. Blumofe and C. Leiserson, "Scheduling Multithreaded Computations by Work Stealing," in *In Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 356–368.

[15] R. D. Blumofe and D. Papadopoulos, "The Performance of Work Stealing in Multiprogrammed Environments (Extended Abstract)," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 266–267, June 1998.

[16] *Tilera TILEPro64 Processor Product Brief*, Tilera Corporation, http://www.tilera.com/sites/default/files/productbriefs/PB019_TILEPro64_Processor_A_v3.pdf.

[17] *NI USB-6210, 16-Bit, 250 kS/s M Series Multifunction DAQ, Bus-Powered*, National Instruments Corporation, apr 2009, http://sine.ni.com/nips/cds/view/p/lang/en/nid/203223#.

[18] "BDTI OFDM Receiver Benchmark^{TM}," http://www.bdti.com/Services/Benchmarks/OFDM.

[19] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," in *Proceedings of the IEEE International Workshop on Workload Characterization*, Dec. 2001, pp. 3–14.

[20] K. Choi, W.-C. Cheng, and M. Pedram, "Frame-Based Dynamic Voltage and Frequency Scaling for an MPEG Player," *Journal of Low Power Electronics*, vol. 1, no. 1, pp. 27–43, Apr. 2005.