ENTRA

318337

Whole-Systems ENergy TRAnsparency

Metrics and Case Studies

Deliverable number:	D6.1
Work package:	Evaluation (WP6)
Delivery date:	1 October 2013 (12 months)
Actual date:	13 November 2013
Nature:	Report
Dissemination level:	PU
Lead beneficiary:	XMOS Limited
Partners contributed:	Roskilde University, University of Bristol, IMDEA Software Institute, XMOS Limited

Project funded by the European Union under the Seventh Framework Programme, FP7-ICT-2011-8 FET Proactive call.

Short description:
This deliverable provides a definition of the parameters within which improvements delivered by other workpackages will be measured, a set of applications that can be used for the evaluation, and a platform for measuring power and energy.

The first part of the deliverable contains a description of the metrics. These are the expected metrics, but are placed in the context of system design.

The second part of the deliverable presents three application domains. Real-time audio processing, robot and motor control, and real-time networking. The first and third application all real-time applications, and as such have a constraint as to when the computation must finish. The second application domain is part real-time, part off-line.

The third part of the deliverable presents the first evaluation platform that we have created, including tools to obtain performance measurements.

Contents

1 Introduction

This deliverable is the first deliverable of workpackage 6. Workpackage 6 concerns the evaluation of the project, and this deliverable reports on the outcomes of the first two tasks: T6.1 (Metrics) and T6.2 (Case studies). These are relatively small workpackages that have identified the metrics that we will use to assess our work, and the *application domains* that we will use to find suitable case studies, and work on creating an evaluation platform that can be used to perform measurements.

The remainder of workpackage 6 is concerned with selecting the actual case studies and the actual evaluation. For this purpose, we have developed a platform on which we can effectively measure physical metrics, and the software associated with the evaluation platform.

2 Metrics

The metrics that we will employ to measure progress are three-fold. First, there are a set of physical metrics that govern power, time, and energy. Second, there is a set of economic metrics that govern the effectiveness of a solution, and then there are the predictability metrics, which govern whether we have gained in what we set out to do: that is to provide transparency.

2.1 Energy metrics and background

Energy metrics are well understood, and in this section we explain the metrics we use, and the extend to which we use them.

The basic unit of energy consumption is the Joule - it expresses the amount of energy required to complete a task. This could be the energy required to lift a person to the top of Mount Everest (approximately 7 mega Joules), or the amount of energy to render an HTML web page (approximately 5 Joules [TAN$^+$12]). This metric is relevant for tasks with a defined beginning and end, but not so for tasks that execute continuously; for this we need to measure *power*.

The basic unit of power is the Watt, which is one Joule per second. It expresses the consistent effort supporting an activity. For example, this can be the power used by a USB mouse (the USB standard limits all devices to 2.5 W, a typical mouse will use a fraction of that), or the power required to run a car at 60 mph (in the order of 10 kW). Power multiplied by time is the energy required, so consuming little power for a long time, or a lot of power briefly both require the same energy.

In DC electronics, power is the product of the voltage and the current. Typical voltages for devices are between 1 and 3.3 volts. Current is used both to charge and discharge *parasitic capacitances* in the design, and current *leaks* from the power to the ground rail.

2.1.1 Dynamic current

All electronic components have some capacitance, and unless the component is designed to have a capacitance, this is a parasitic capacitance. When, for example, a transistor is switched from ON to OFF or vice versa, the gate must be (dis)charged, which takes a small bit of power. Although the gate capacitance of a single transistor on a chip is only minuscule, switching 100,000 transistors at 500 MHz will require a measurable current. This current is related to the work that the chip puts in and has the following characteristics:

- Dynamic current is linear in the frequency; if a processor is clocked twice as fast, its dynamic current consumption will double. Note that the amount of work performed may also have doubled, meaning that the energy consumed due to this is invariant under changing the frequency. Often, numbers such as "μA per MHz" are used to indicate this invariant. Assuming the voltage is known, then this form of invariant is analogous to an inverse representation of the performance per watt metric described in Section 2.1.5.

- Dynamic current is linear in the voltage that is applied. Halving the voltage will halve the current, reducing the power consumed to a quarter (as power is the product of voltage and current). Reducing the voltage reduces the frequency that they device can run at (in a non-linear manner) and a trade-off can be made between speed, voltage, and energy consumed.
- Dynamic current is inherent in the chip-design; chips can and are designed to reduce dynamic current consumption. In this project we take the design of a chip as given.

2.1.2 Static current

Unlike dynamic current, leakage current serves no useful purpose. Dependent on the technology node used, the voltage applied, and the temperature of the chip, leakage current varies from almost nothing (for technology nodes with large features), to a large part of the design. As leakage is independent on whether the processor is actually performing a task or not, there is an advantage to performing a task in a shorter period of time and then switching the power supplies to the device off. This leads to a series of strategies for devices:

Wide and slow is a strategy where the clock frequency is reduced, voltage is reduced, and using many devices in parallel. This minimises dynamic power consumption at the expense of leakage.

Run to halt is a strategy where the clock frequency is set high, the voltage may be increased, and the computation is performed as fast as possible whereupon the device is switched off. This minimises leakage.

Dynamic Voltage and Frequency Scaling (DVFS) is a strategy where at any time in the computation the frequency is adapted to be just fast enough, and the voltage is set to be just high enough.

An appropriate strategy depends on the system under design. A task that has a defined beginning and end often benefits from run-to-halt, as the battery can be disconnected once the task is completed, optimising the energy consumption. A task which is a continuous background task cannot run-to-halt (as there is no end stage), but it may be executed in little sprints using DVFS to optimise the power consumption.

Leakage current is one component of *static current* the counterpart of *dynamic current*. Other parts include parts of the system that are “always on” or parts of the system where the power consumption is otherwise unrelated to the frequency

2.1.3 Losses and power budgets

All electronic components have some resistance - that means that they will dissipate power when a current flows through them; power that has served no useful purpose. Two components in particular cause losses: voltage converters and power-switches. A voltage converter converts, say, 3.3 to 1 V. A good converter can do so with 90% efficiency, meaning that 11% extra power is used to run the system from 3.3V rather than from 1V. Power-switches that are used for *run-to-halt* systems dissipate power themselves too. In this project we are not concerned about these losses as they add to the power bill in a predictable manner.

Input and output operations take power. When designing a system, a power or energy budget is created that allocates power to all parts of the system. Analogue amplifiers, front-ends, LCD screens, and memories all take significant amounts of power. These are outside the scope of our work. The cost of communicating with devices could be looked at.

Ideally an energy budget can be seen to be the simple sum of components. However, that simplicity is often impossible to achieve, for example because strategies to reduce energy consumption may have a multiplicative effect, which make it difficult to attribute savings to a specific strategy. For example, we may be optimising a system that uses 100 mW. A voltage reduction saves 19 mW, and a subsequent optimisation of a sequence of instructions may save an additional 10 mW, resulting in a 71 mW system. However, starting from the 100 mW, the latter would have saved 12 mW, and voltage reduction will gain only 17 mW. The final answer is the same; the optimised system requires 71 mW. But the 29 mW savings cannot be split in a unique way over the various strategies: voltage scaling saved 17-19 mW and optimising an instruction sequence saved between 10-12 mW. There is no simple compositional model, and care has to be taken to present something meaningful to the end user. Even something as simple as an ordered list of savings that can be made is difficult, as savings may depend on each other (for example, voltage scaling is only possible once the frequency has been reduced), or the true effect of a saving only come into effect when other savings have been applied.

2.1.4 Peak power

In addition to the energy and average power requirements, *peak power* is an important metric. There are two sorts of peak-power:

Microscopic peak power Microprocessors made with state-of-the art technology nodes almost always rely on an internal clock signal that synchronises all internal components of the chip. As a consequence, all chip components will start to operate simultaneously on a clock edge; that is, in the first 100 ps of a 2000 ps (500 MHz) clock cycle, many transistors will open and close, causing a current requirement that may be an order of magnitude higher than the average current. For this purpose *decoupling capacitors* are placed closely to the die (typically within a few millimetres).

Macroscopic peak power During the execution of the program, there may be program fragments that require more power. For example, a program may have to occasionally compute an FFT, or send or receive a packet. During these periods the average power consumption as measured over tens or hundreds of clock cycles increases, requiring the DCDC converter to supply a higher current.

The latter is an important measurement for system design: sizing the power supply to be appropriate for the task reduces the system cost and gives the system designer the opportunity to select a power supply that is most efficient. As efficiency typically depends on load, having a system where peak performance and average performance are similar can also lead to a substantially more efficient power supply on average. When integrating the power profile to compute the energy consumption, the efficiency of the power supplies varies over time, and this should be taken into consideration.

Peak power is also important in battery powered designs. Batteries have an internal resistance that limits the current that they can supply. Hence, the maximum power consumption governs the type of battery that can be used in a battery powered system. Determining the peak power consumption at any time, and reducing the peak power consumption will hence have an important effect on system design.

2.1.5 Performance per unit of power

Performance per watt is a measure of the energy efficiency of a specific computer system. It measures the rate of computation that can be delivered by a system for every watt of power consumed. Performance (in operations/second) per watt can also be written as operations/joule, since 1 watt = 1 joule/second. The performance metric used depends on the definition and the purpose. The most common measures of performance used are FLOPS, MIPS, or the score for any performance benchmark. Measures of power usage can vary to fit purposes, for example, a metric might only consider the electrical power delivered to a machine directly, while another might include all power necessary to run a computer, such as cooling and monitoring systems. The power measurement is often the average power used while running the benchmark, but other measures of power usage may be taken into consideration, like peak power or idle power.

2.2 Economics

In low power design there is a constant trade off between functionality and energy consumption. Typically maximum power budgets and minimum functional requirements are defined, and then a best effort solution is picked that fits both budget and requirements.

Other dimensions that may be important are costs and time-to-market.

Using, for example, four rather than a one core to design a "Wide and slow" system is going to be more expensive, but we are largely going to ignore the costs aspect for two reasons:

- First, the costs of the actual processing component is only a small part of the total cost of the system, and doubling the number of processing cores does not double the system cost: power supplies, I/O systems, casing, and many other parts are shared and costs of these parts will stay the same.

- Second, and most importantly, further miniaturisation of underlying technology nodes provides extra processing cores economically. With feature sizes decreasing from 65 to 40 to 28 (to 20 to 14) each step roughly doubles the number of transistors that fit on the same size die, whereas wafer manufacturing costs increase more slowly. Shrinking feature sizes can either be used to reduce chip size and costs, or they can be used to add functionality in order to improve efficiency. As the costs of the core transistors is only a small part of the chip cost (packaging, the pad ring, and testing contribute significantly to the final cost), the cost of an few cores in a smaller process node is surprisingly low.

Time-to-market is something we will address indirectly. Time-to-market is an intangible cost factor in system design. Shortening time to market improves competitiveness, and increases innovation. Providing a reliable mechanism that predicts energy usage early in the design cycle will shorten the time-to-market of low-energy products.

Another intangible factor (or at least one that is hard to measure) is the complexity of using tools which perform analysis or optimisation. For example, is the tool fully automated, or does it need specialist expertise to operate. Ideally, it is easier to operate the tool to perform the analysis and optimisation, then it is to perform said analysis and optimisation by hand.

2.3 Accuracy of result

A further metric that must be considered is the accuracy of the result. One way to save power is to reduce the precision of computations; for example, compute on 16 bit values rather than on 32 bit values. The effect on the end result may be quite small. For example, in an audio system good quality audio is 24 bits; 32 bits is considered better by some expert listeners, even though the difference is imperceptibly small; 16 bits is audibly different, but it may be acceptable for a system to switch to 16 bits if that extends battery life.

This metric can be applied in a more continuous manner, in that the number of bits can be gradually reduced, or the number of bits in intermediate results can be reduced.

The choice of accuracy is not a simple compiler optimisation. Standards often dictate specific number of bits that need to be supported, for example in the case of audio 8, 16, 24, or 32. However, the implementation can choose to reduce the number of bits during computations (at the cost of a loss of fidelity of the results).

2.4 Accuracy of timing

Many programs that are of interest have timing requirements that can either be a hard real-time requirement where the system fails if the timing is not met, or a softer requirement where the system will gradually lose fidelity or usability if timings are not met. An example of the former is the deployment of an airbag; failing to inflate it in time is more dangerous than not inflating it. An example of the latter is decoding an MPEG stream, where it is OK to occasionally miss a deadline, it just means that the video does not look as good as it could.

The percentage of missed deadlines is a metric that can be traded-off against the power savings made. For hard real-time problems, this percentage cannot be more than 0. For softer problem, the percentage depends on the user expectation.

2.5 Accuracy of prediction

Part of the project is to be able to analyse and predict the relationship between the program and the energy profile. The accuracy of this prediction is another metric. One aspect is whether the predicted complexity is right. A second aspect is whether the predicted value is right. If the complexity is predicted correctly, then the programmer can meaningfully make trade-offs early in the design, and they will need to make a few calibration runs to get the correct absolute values.

Getting an absolute prediction right will very much depend on the underlying technology. No two chips are the same; even though they may have been cut from the same wafer they can exhibit different power and timing characteristics. Modern chip manufacturing processes produce chips with wide variations in the process, meaning that chips can easily be 10% slower/faster, or consume 10% more or less power. During chip design a “worst case” speed, voltage drop, and power consumption are computed, and verified to be within the desired envelope. This “worst case” is the worst that is acceptable, and is picked so that the process will yield an acceptable number of die per wafer that are better than the worst case.

A relative prediction is likely to be far more useful to the end user.

3 Application domains for case studies

We have identified three domains that we will explore for case studies: Audio, Motor control, and real-time networking. We will give a description of these three fields.

3.1 Real-time audio processing

Real-time audio processing comprises, for example:

- Digital mixing desks
- USB head-phone amplifiers
- DJ equipment.

It is characterised by the requirement to input, process, and output digital audio samples in real-time.

- Digital audio comprises one or more audio channels. Typically there are two channels (Left and Right), but some DVDs carry 8 channels (7.1 surround), and a music stage may have 48 or 96 channels.
- Each channel comprises a stream of data values approximating the analogue value of the channel. The *sample rate* defines the number of samples that are recorded or played each second. Sample rates are normally between 8 and 192 kHz, for low quality voice recording to high fidelity recordings. Common values are 44.1 kHz (Compact Disc recordings) and 48 kHz.
- Each sample is represented in a fixed number of bits, normally between 8 and 32, common values are 16 and 24 bits. The encoding is normally a fixed point signed integer representation. For example, if the samples are represented using 16 bits, then the value `0111 1111 1111 1111` represents maximum positive amplitude, and the value `1000 0000 0000 0000` represents the maximum negative amplitude.

Audio applications typically comprise a mixture of three sorts of tasks:

- Audio samples are input from and output to external hardware. This may be a very direct interface such as I2S where the bit values are clocked into a Digital to Analogue converter.

- Digital signal processing (DSP) may take place on the audio signal. This may be as simple as volume control (scaling sample values), mixing two streams (adding sample values), or a more complex filter.

- Audio samples may be transmitted over some sort of real-time network, such as AVB Ethernet or Isochronous USB. This will require some form of buffering to ensure that sufficient samples are kept to prevent underflows in the audio stream.

Energy and power prediction and optimisation can be applied at many levels in this domain:

- Power consumption is crucially important for audio devices that are self powered (such as headphone amplifiers for a mobile phone), or for device that are powered over USB (such as laptop speakers).

- The input and output parts of any audio application may be susceptible to static scheduling, parallelisation, or serialisation; enabling the device to run at a low frequency, and a low voltage

- The DSP part of the audio application can be parallelised, and can have an adaptive accuracy in order to save power.

Two example case studies in this domain may be a sample rate converter, or a USB-audio device. The former is a system that takes an audio stream sampled at one frequency, and produces a stream sampled at a different frequency. The data streams are input and output over I2S, and the frequencies concerned are downsampling from 192 to 48 KHz. This case study is closely related to the FIR and I2S benchmarks: it comprises an I2S input process, a process that splits the audio into a left and right data stream, two processes that execute a Finite Impulse Response filter, a process that joins the left and the right stream, and a process that outputs on I2S. The latter is detailed in Deliverable 5.1.

3.2 Robot and motor control

A second field that may be of interest is that of motor control. In its simplest form a motor control programs controls the drive to just a single motor; but multiple motors may have to be driven in synchrony, for example when a robotic arm is to be controlled. Motor control is found in many places such as:

- White goods such as vacuum cleaners and washing machines

- Products on factory lines such as robotic arms and automated belts
- Small products, such as an electric toothbrush or a robotic spider.

Motor control is usually a closed-loop process where real-time feedback is used to precisely control a motor:

- At the lowest level, a pulse width modulated (PWM) signal drives the power FETs to drive the motor. The density of this signal is varied to create the right pull and push on the magnets. This PWM loop may run at speeds of 10 kHz - 1 MHz.
- A control algorithm ensures that the PWM signal produces a stable and desired speed of the motor. This control algorithm may be a basic Proportional-Integral-Differential (PID) controller, or it may perform a more complex computation. These computations are typically fixed point computations with a precision of 16-32 bits, and either with scalars or with matrices of up to 4x4 elements, and execute at a speed of between 1 and 50 kHz.
- At the highest level, a complete trajectory is computed involving multiple motors, using some form of Inverse Kinematics (IK). These are typically written using floating point computations. These typically run slowly as they just compute a trajectory once.

The software structure typically comprises the three tasks above, taking their input from a variety of sources:

- The highest level is typically only used in a system where there are multiple motors that are physically interacting with each other; for example the joints in robotic arm, where there are choices to be made how to extend the arm (which joint is extended first, or whether both joints extend simultaneously) and how this affects the dynamics of the system (for example vibration when the arm comes to a stop).
- There is a control loop for each motor. The control loop has a *setting* (the desired speed, position, acceleration), and this may be set by the previous task, or over some real-time-network (in the case of multiple motors on a factory floor), or directly by the user (in the case of white goods).
- For each motor there are multiple PWM processes, that all execute synchronously, and also synchronously with any measurements operations (such as measuring the current flow through the motor). The number of PWM components depends on the type of motor, for example three.

The gains of making the software for motor control power efficient may be relatively small in applications that drive large motors. When driving a one kWatt motor, more can be gained by increasing the efficiency of the motor by 0.1%, than by increasing the efficiency of the software by 50%. Having said that, for small motors that consume less than a Watt, the cost of software is significant. In addition, there is no excuse to waste power simply because it appears to be a small fraction.

In robot control there is by nature a multitude of control tasks that happen simultaneously - often three or four tasks per joint. This lends itself to a strategy of wide and slow. Adapting precision may also be a strategy: the eventual aim is for control to be smooth; for this smoothness to be achieved an appropriate precision of computation is required. Measuring smoothness is, however, out of scope of this project.

Two example case studies in this domain may be the driver for a single brushless motor, or the inverse kinematics algorithm for a joint. The former comprises at least five tightly synchronised tasks: driving three PWM tasks in synchrony with measuring the current, and computing the required modulation. The inverse kinematics task can be formulated as a single monolithic task that can be optimised.

3.3 Real-time networking

The final field that could be of interest is that of real-time networking: delivering data from A to B, typically at just the right time. Example real-time networking includes:

Real-time networking comprises, for example:

- Distributing sensor data over Ethernet in an automotive environment.
- Transmitting audio data from a laptop over a USB cable.
- Controlling all motors of a conveyor belt from a single host using a PowerLink network.

It is characterised by the requirement to accept data on the network and deliver it for processing without adding delay, and to place data on the network in at a pre-allocated time slots.

- Real-time networking typically comprises a series of point-to-point links, where a fraction of each of the links has been allocated for each real-time stream.
- Real time streams have a limited bandwidth, and are routed over each of the links. Since all bandwidths are known (either statically, or dynamically) all packets can always be transmitted on time. This limits the buffering required, and limits the latency incurred by traffic.

- At the end-point of a real-time network data may enter or exit the network. Any transmitted data has to depart in the window set aside for this data. In an efficient and low-latency system, software is organised so that there is little delay between data collection and data being placed on the network cable. Similarly, received data should be processed at the correct time.

Real-time networking applications typically comprise at least four tasks:

- Transmitting data
- Receiving data
- A MAC (Medium Access Control) layer
- Real-time Clock recovery

Energy and power prediction and optimisation can be applied at many levels in this domain:

- In its most extreme case, sensor networks often use a process where the node is asleep for a long time and then wakes up just prior to processing a network packet. This is a specialised form of power saving that is hard-coded, ideally more general analysis tools can provide power saving.
- Controlling the network may take 25-50% of the processing resources of the endpoint (with the remaining 50-75% being used for the application task). As such, using the often predictable nature of real-time networks can be used to reason about software and reduce power consumption.
- Often real-time networks are specified to run at a higher speed than strictly necessary: for example, Gigabit rather than 100 Mbit Ethernet. The reason for this is that this reduces latency (!) as the time for a packet to be in transit reduces by a factor of 10; in the case of a 125 byte packet this reduces latency from 10 to 1 μs. This reduction in latency may be achievable by writing or analysing the software correctly, enabling much more power efficient PHYs to be used.

Two example case studies in this domain may be the lowest layers of an Ethernet AVB system or a PowerLink system. AVB (Audio Video Bridging) is a real-time protocol that executes on top of Ethernet. It uses an 8 kHz beat to transmit packets, and requires a *traffic shaper* to ensure that AVB packets are transmitted on the 8 kHz beat in a time window that is reserved for AVB, and that the remaining traffic is transmitted outside this window. The size of the window is computed dynamically dependent on the number of streams reserved. It also enables us to examine speeds of links. PowerLink is one of the many standards for real-time industrial Ethernet. It is open, and as such an easier target for research.

4 Evaluation platform

We have created a hardware and software platform that enables us to measure the energy, time, and power used during execution on real hardware. We have achieved the following:

- We have created a variant of the XTAG2 debug adapter that enables power to be measured (informally named the XTAG3, this is available to the consortium only at present).
- We have defined a protocol that enables power measurements and application probing to be performed simultaneously, and data to be transported simultaneously over the USB connection to the host computer.
- We have designed and built two development boards that have power measuring structures on them. One of the boards uses a 5 V supply and has a single tile, the other uses a 3V3 supply and provides two tiles. Both boards support voltage scaling.
- We have designed and implemented an extension to the XMOS toolchain that allows power measurements to be recorded and/or displayed in real-time.

4.1 The basics

In order to measure power on a supply, we have chosen to place a small, known, resistor in series with the power supply, and measure the voltage drop across that resistor. Following Ohm's law, this enables us to calculate the current through the resistor, subsequently the power consumed across the load, and the energy used by the load:

- A small resistor of R_{shunt} ohm (say 0.1 ohm) is placed in the supply line (this is known as the shunt resistor).
- Given an unknown load I on the power supply of (say 0.2 A) this resistor will cause a voltage drop of $I \times R_{shunt}$ (say 0.2 A $\times$ 0.1 ohm $=$ 0.02V).
- This voltage drop is too small to measure directly and is fed into an OpAmp with a gain of G, creating a signal V_{shunt} (say with a gain of 100 a small signal of 20 mV would be amplified to $0.02V \times 100 = 2V$).
- The debug adapter measures V_{shunt} using an Analogue to Digital converter, and simultaneously measures V_{sup} on the low-end of the resistor (say, 0.95 V)
- The instantaneous power consumed is therefore $V_{shunt}/100/0.1 \times V_{sup}$ (say $2/100/0.1 \times 0.95 = 0.19W$).

This is a fairly standard process to measure power, but requires the values for the gain (G) and for the resistor (R_{shunt}) to be chosen carefully.

4.2 The XTAG3

We have designed a new version of the XTAG2. It has an extra connector that carries four analogue signals. The first two wires carry the signals on either side of the shunt resistor in the primary supply and the second pair of wires carries the voltages on either side of the shunt resistor in the secondary supply. The AD-converters can measure over a 0-3.3V range, and we have chosen to use OpAmps with a gain of 100 for amplifying V_{shunt}. An integrated AD-converter samples the four signals at a 250 kHz rate, giving us a power measurement every 4 μs. The four measurements are transported to the host computer over USB using the XScope interface that can also carry software measurements supplied by the program [xmo]. This enables both software information such as “Function started” and “Function ended” and power information to be collected simultaneously.

It is up to the board developer to decide which signals should be measured. Typically we measure the supply going into the main DC to DC converter that feeds the processing core as the primary supply (this is a 3V3 or 5V signal), and we measure the IO voltage of the board as the secondary supply (3V3).

It is also up to the board developer to decide on suitable resistor values. Too high a resistor value may cause an unacceptable voltage drop, but too low a resistor value may cause the resulting voltage drop to be unmeasurable. As the voltage drop is amplified by a factor 100 and the ADC is limited to measure 3.3 volt, the resistor should be chosen so that the drop over the resistor will not exceed 33 mV.

4.3 Development board

We have developed two boards for performing power experiments. A single-tile board that contains an XS1-U8A-64-FB96 processor and a dual-tile board that contains an XS1-A16-128-FB217 processor.

The first board runs off a 5 V power supply; it can run up to eight simultaneous tasks on a single processor, and supports voltage scaling and frequency scaling. The second board runs off a 3.3 V power supply; it can run up to 16 simultaneous tasks on two processors. It supports voltage scaling (but both processors must run at the same supply voltage) and frequency scaling (on a per-tile basis).

4.4 Tool support

The tool that collects data from the XTAG is `xgdb`, the debugger that is part of the XMOS toolchain. xgdb connects to the XTAG over a USB interface (using libusb), and reads both ordinary XScope traffic and voltage/current measurements.

The data is normally stored in an XML file, but sampling four channels at 250 kHz produces approximately 80 MBytes of XML every second. Hence, instead, `xgdb` can pipe the data directly into an analysis program that can only record data that is relevant (between start and end) and only compute the relevant metrics (maximum current, total energy, etc) rather than store all 250,000 measurements each second. Code to read from the socket is available publicly `https://github.com/xcore/sw_ethernet_tap` and we will develop a bespoke application to measure relevant metrics as part of T6.3.

5 Summary

In this deliverable we have defined the metrics and the application domains that we wish to use for evaluating the ENTRA results. This is a set of metrics and domains that will enable us to perform an evaluation. If the industrial partner finds new areas that are better suitable for evaluation, it will be happy to supply them for consideration by the consortium.

The evaluation platform and software are made available to all partners, and enable us to look at real data.

References

[TAN$^+$12] Narendran Thiagarajan, Gaurav Aggarwal, Angela Nicoara, Dan Boneh, and Jatinder Pal Singh. Who killed my battery?: analyzing mobile browser energy consumption. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 41–50, New York, NY, USA, 2012. ACM.

[xmo] Use xTIMEcomposer and XScope to trace data in real-time, at https://www.xmos.com/download/public/Trace-data-with-XScope(X9923H).pdf.