Journal of the Indian Institute of Science
A Multidisciplinary Reviews Journal
ISSN: 0970-4140 Coden-JIISAD

REVIEWS

Semiconductor Solutions for the Internet of Things: The Role of Event Detection, Asynchronous Design, Energy Harvesting and Flexible Electronics

G. Venkatesh

Abstract | Within the next decade, it is expected that the number of items of common use that would get connected to the Internet would be an order of magnitude higher than the population of humans. This is exciting mainly because it means that every useful object would become "smarter"—i.e. that they would be able to sense their context, analyse information locally, co-ordinate with each other, communicate with remote analytics services and take intelligent actions. Such "smartness" would need to be embedded within objects in a reliable and secure way, without significantly adding to its cost or energy use. Recent System on Chip (SoC) solutions released from a number of semiconductor companies claim to meet precisely these requirements. This paper highlights four research areas that could make a significant impact on the evolution of such solutions in the future: (a) development of event detection mechanisms to closely match the activity within the SoC to that of its context, so that its energy use is minimized (b) use of asynchronous design techniques to better manage demand variations within available (energy) resources (c) techniques to harvest energy from the object's immediate environment and (d) ways to seamlessly embed these solutions into the end object by leveraging advancements in printed and flexible electronics.

1 Introduction

There is considerable excitement around the concept of Internet of Things (IoT) which essentially would enable almost every useful object to be connected to a network.[1–3] The belief is that this would unleash a revolution in the way these objects would be employed similar to the way the Internet (of humans) revolutionized everyone's lives. While the numbers vary from a few to many 10s of billions, most market analysts agree that within a decade or so, the no of connected objects would be about an order of magnitude larger than the number of connected humans.[2]

1.1 *IoT and M2M*

In most of the recent literature, the term Machine to Machine communication (or M2M) has been used interchangeably with the term IoT. However, historically, the main focus in M2M has been on connecting a "machine" (or object) to a remote service enabling the remote monitoring and control of a number of machines in the field. One would assume that the new term IoT would extend this concept by making each object "smarter" in terms of enhanced abilities to do sensing and actuation, information processing, co-ordination with other objects and communication with analytics services on the cloud.[1,2] The emphasis is on processing the collective information from several units in a region in order to create effective actions for each unit.[3] There is also an attempt to build the IoT around open standards (in keeping with the way in which the Internet has evolved),[4,5] whereas M2M could

Department of Electrical Engineering, Indian Institute of Technology, Chennai 600036, India.

have been deployed in the form of proprietary in-house solutions.

Thus while electronic control of machines has already led to the proliferation of electronic logic and microprocessors in many objects of daily use, M2M has led to incorporation of a communication scheme to transfer local data to remote servers, and IoT goes beyond this by adding significantly new capabilities: the leveraging of cloud services,[3] the integration of more complex sensing or actuation functions,[3] the recording and analysis of information to become more context aware,[3] managing energy use in a clever way,[6] harnessing energy from its operating environment,[15] or the integration of all these functions into low cost devices that can fit into all kinds of shapes and forms.[16] However, if one feature has to be picked that makes an IoT solution stand out when compared to other systems, this would surely be its *low duty cycle mode of operation,* i.e. the IoT device is turned off (in sleep mode) most of the time waking up once in a while to perform some task.

1.2 *IoT semiconductor solution*

In this article, an IoT device is taken to be one that sits on the interface between the physical environments on one side and the network environment on the other side (see Figure 1a), and implements the following functions:

- ***Sense***: Pick up some relevant data from the environment
- ***Analyse***: Store and partially analyse the sensed data to create useful information
- ***Communicate***: Securely communicate with remote servers or cloud services to transfer information about the sensed environment
- ***Co-ordinate***: With other IoT elements to either:
  - improve the quality of information collected, or
  - to communicate the collected information to a cloud service
- ***Deploy Resources***: Take valid instructions from other elements, remote servers or from the cloud service and schedule the deployment of resources
- ***Actuate***: Perform actions in the environment (according to received instructions)
- ***Conserve***: Manage its energy use wisely to conserve its scarce energy source; mostly by operating in a low duty cycle mode.

1.3 *Maintenance challenges and their impact on IoT solutions*

The challenge of maintaining the IoT system post deployment is often underestimated when an IoT solution is proposed. A quick back of the envelope calculation would reveal that the cost of producing and deploying an IoT solution may be dominated by the cost of maintaining it over an extended period of time. Thus it is important to ensure that the deployed IoT solution operates in a maintenance free fashion, or in the rare event that maintenance is required, it is possible

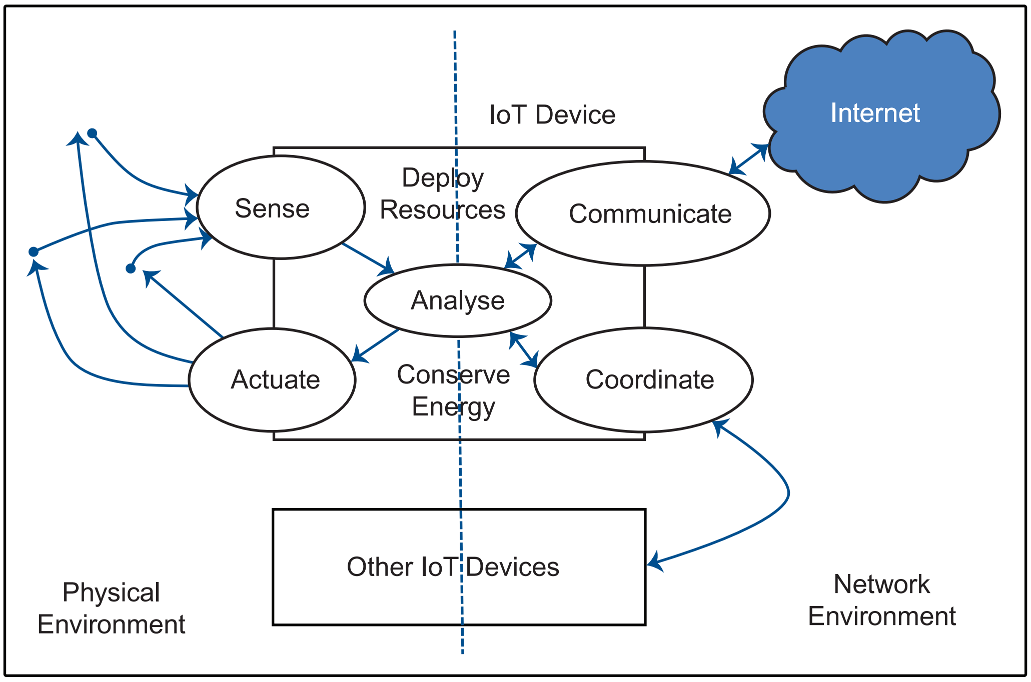


Figure 1a: The IoT device sits between the physical and network environments.

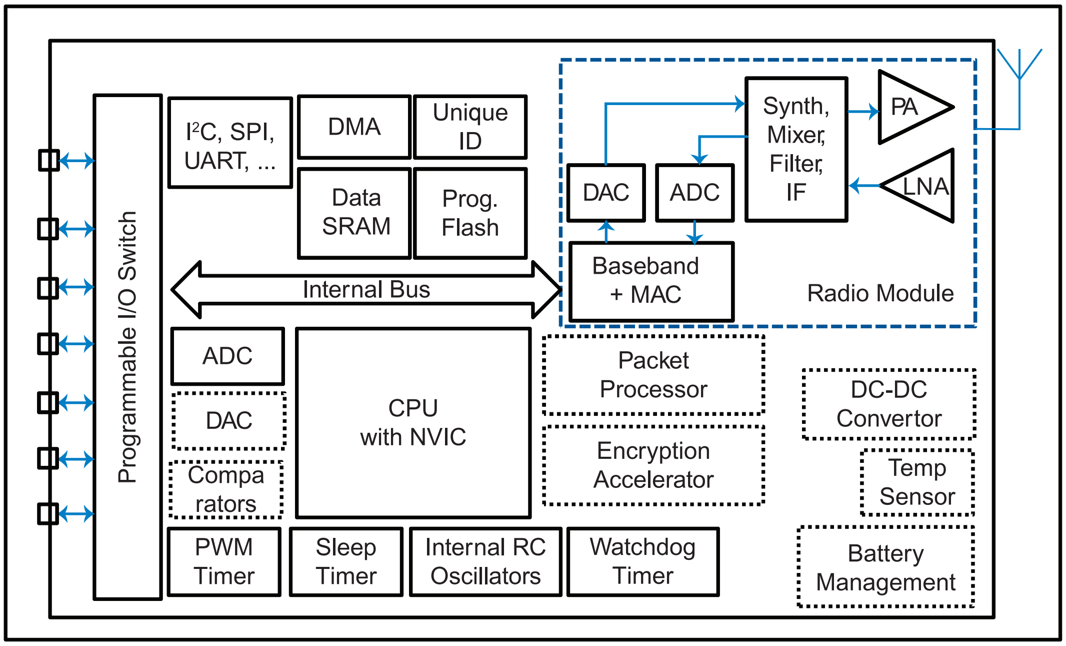


Figure 1b: Architecture of a typical IoT semiconductor device.

to do a substantial part of that maintenance remotely.[5]

Maintenance may be required because of wear and tear of the hardware, drift in the parameter values of the components in the system (especially sensors which require calibration), to replace the battery, to do software updates to fix defects or software upgrades to introduce new features. Hardware wear and tear would require physical replacement of the part, unless the system is designed with redundancy, or if the system can fall back to a simpler mode of operation in which the failed part is not required. The recalibration could sometimes be done remotely if the IoT hardware is properly designed. Software updates or upgrades are easily done remotely without any manual intervention. It is the battery replacement part that is tricky—the safest way of dealing with this issue is to design the system so that the battery does not need to be replaced for the expected lifetime of operation (which could be 3 to 10 years). For this, the system's energy use needs to be managed extremely wisely. In the event that the system is working off the limited energy harvested from another source, this becomes even more critical.

1.4 *Key research trends that could impact IoT semiconductor realisation*

Among several areas that contribute to advancing the realization of IoT semiconductor solutions, this paper picks four research areas that is attracting attention due to their potential high impact. Each area is introduced briefly below.

1.4.1 Event detection and activity management: As explained in section 1.3, it may be necessary to manage the energy use within the IoT device so that the system's battery does not need to be replaced over its lifetime. In case the system is running off harvested energy, the energy use needs to match the available energy from harvesting. This means that the IoT device needs to be designed so that at any given time there is practically no activity running within it that is not needed for its functional operation at that time. The simplest way of matching the IoT internal activity to its external environment is to put unneeded internal components of the IoT device in a sleep state when there is no functional activity that depends on them (depending on the acceptable time for wake up from sleep, there could be multiple sleep levels). Surprisingly this simply stated requirement turns out to be somewhat difficult to implement. If done through a scheduling scheme which wakes up the component at the appropriate time,[6,7] then this scheme needs to make assumptions about the environment,[7,8] which would then be optimal only in some conditions. An alternative may be to get the environment to take on the role of waking the component or the entire IoT device. If so it could be the Internet side which would wake it (radio driven wake[9]) or the sensing side (sensor driven wake).[8,10] In either case, the IoT semiconductor needs to have some of its modules running just to detect the wake signal, which wastes considerable amount of energy. The trick then seems to lie in the ability to do external event detection by expending as little energy as possible.

1.4.2 Asynchronous design methodologies: The benefits of using asynchronous design methodologies to realize digital systems have been orchestrated a number of times over the last five decades. The key advantage of asynchronous designs is that they are event based and so provide a natural way to match internal to external activity (as highlighted above for energy management).[11] Asynchronous designs also show much better tolerance to power supply variations, since the slowing of the circuit at a lower power supply voltage has no impact on the design. This is an essential requirement when dealing with harvested energy or run down batteries, where it may be desirable to deliberately slow down the circuit to match the available energy. Another benefit of asynchronous circuits is that larger circuits can be compositionally assembled from smaller ones without the need to verify the timing correctness of all the parts—however this may not be a significant factor in IoT SoC realisation. Finally, asynchronous circuits seem to be much more energy efficient compared to synchronous circuits when there is substantial variation in the activity levels in their operating environment,[12] as the synchronous circuit would have to be overdesigned for the worst case. This means that asynchronous designs may be more suitable in IoT devices that are designed for unpredictable environments.

1.4.3 Energy harvesting: A key aspect of IoT devices that was discussed in section 1.2 is that they would be operating in a physical environment from where they would draw data using sensors. If so, then it would be reasonable to assume that this physical environment would have enough energy sources which the IoT device could rely on to run its operations. A simple example of this line of thinking is evident in the way Radio Frequency Identification (RFID) systems are designed.[13] The tag which stores the unique ID of the object to be identified can be passive (i.e. has no energy source), but is able to harvest energy from the RFID reader to operate. Its operation involves producing a specific backscatter RF pattern using the stored ID within the tag, which is sensed by the reader. Clearly the process of reading an ID would require energy, so the IoT device (in this case a simple RFID tag) can harvest this energy to do its task. The same logic could apply to any physical measurement—for instance pressure, light, heat, strain, vibration etc. The physical phenomenon that is being measured may be at a sufficient energy level that the IoT device may be able to harness some of it to measure and communicate without impairing the measured value. Even if the measured phenomenon does not have sufficient energy, there could be other phenomena happening in the same environment which could act as sources of energy. For example, an IoT device measuring the strain within an element of a structure may be able to harvest energy from the vibrations of the structure. Since there are very many RF energy sources operating within urban environments today, harvesting this RF energy for IoT device operation has been a subject of a number of recent studies.[14,15]

1.4.4 Printed and flexible electronics: Since an IoT semiconductor solution is expected to be embedded within a physical object, one of the key considerations would be the physical attributes of the IoT solution (size, weight, form factor, flexibility, ruggedness, etc.) that would allow it to be seamlessly placed within the physical object without altering any of the physical attributes of that object.[16] When the packaged objects go through approvals from respective agencies, the embedding of the IoT device must comply with the requirements from those agencies. The use of printing techniques to create small electronic circuits opens up new possibilities to insert IoT functionality during the manufacturing or packaging process which already use some printing methods at least to label the parts or packages. Printed electronics has evolved to a state where sensors, antennas, batteries, touch pads, displays, logic and memories can be printed to realize simple circuits that can sense/record physical parameters and communicate/display the values.[17] What makes printed electronics interesting is that these circuits can be printed directly onto plastics, stainless steel, ceramics or other surfaces of the manufactured product or on flexible substrates such as plastic, paper or textiles used in packaging it.

1.5 *Organisation of the paper*

The paper is organized as follows: Section 2 identifies the basic building blocks required within an IoT semiconductor solution. This is further detailed with the specific example of IoT solutions for smart energy meters, which is an excellent use case for IoT. Section 3 explores issues in managing the activity within the IoT semiconductor device to minimize energy use by employing low energy event detection schemes. Section 4 discusses the advantages of asynchronous design techniques for creating IoT solutions. Section 5 looks at the possibility of harvesting energy from different energy sources that may be available in the environment. Finally, section 6 examines the potential of printed electronics to embed IoT circuits naturally into the manufactured or packaged objects through printing processes.

2 IoT Semiconductor Requirements

A rough high level architecture of the IoT device and its environment is depicted in Figure 1a. According to this, the IoT device is the one that realises the interface between the physical environment on one side and the network environment on the other side. Note that an implementation of just the sense, analyse, actuate steps is what is found in microcontroller based systems found in daily use appliances like washing machines. Augmenting it with resource deployment and communication would bring it in line with conventional M2M implementations. Addition of co-ordination and energy conservation is what makes these systems complete IoT solutions.

The architecture of a typical IoT SoC that implements this is depicted in Figure 1b. This section explains how an architecture and modular composition such as that given in 1b can be used by an IoT device to implement all the requirements stated in section 1.2.

2.1 *Sensing*

A possible plug and play sensor architecture is shown in Figure 2a (conforming to IEEE 1451.4 standard).[18] Since the IoT semiconductor solution is expected to be somewhat general purpose and not tailored to fit a specific vertical need, the physical transducer for the sensor needs to be external to the semiconductor chip. To lower the cost, most of the post-transducer electronics should be integrated within the semiconductor chip (including if possible the operational amplifier). This means that the chip must be able to input (poor quality) analog signals (maybe from multiple sources), for which it needs to allow configuration of some of its ports as appropriate high impedance analog input ports.

Sensing requires an analog to digital convertor (ADC). The simplest way is to compare the input signal with different set levels. However, this loads the CPU subsystem and takes too much time to input a sample. A successive approximation register (SAR) and DAC can be used to accelerate this process in hardware. The Σ-Δ convertor converts the input signal into a series of pulses whose average signal value is equal to the input signal (See Fig. 2b).[19] This can be done by using only digital logic, a comparator and a 1 bit DAC but the circuit needs to operate at a much higher

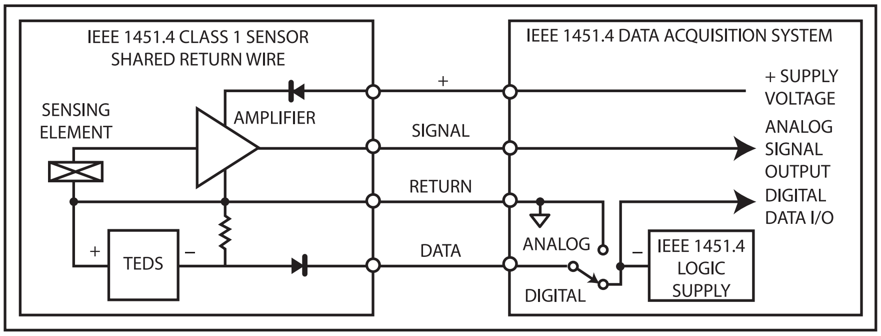


Figure 2a: Transducer—Sensor architecture as per IEEE 1451.4 standard.

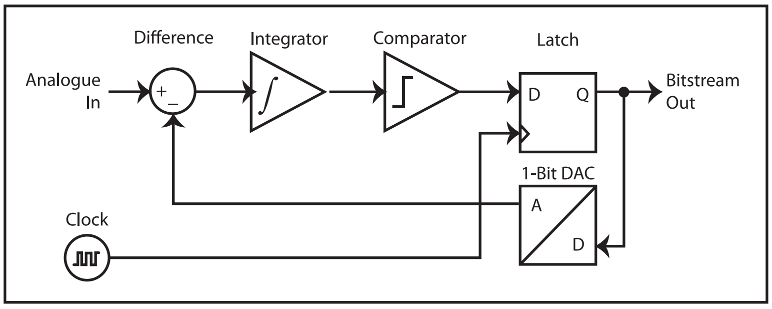


Figure 2b: ADC built using a Σ-Δ modulator.

rate than the rate at which the signal is being sampled.

Choice of the right ADC depends on the application—the survey in[20] compares a very large number of ADC implementations for different uses; however, the IoT semiconductor can only implement one of these. Designing an ADC for the IoT that would be suitable for a variety of applications is a non-trivial task[21] which has not received the attention it deserves.

The sensing can be triggered: (a) periodically (using a timer) (b) through a fixed schedule programmed on the CPU (c) by an event detected by the sensor or (d) through a remote request. This triggering is typically implemented using a (nested) vector interrupt controller (NVIC), which passes control to an interrupt service routine that configures the input ports, sets up the ADC and records the sample into memory. However, for energy efficiency it may be desirable for the ADC to be directly activated by a timer or input event which then writes its output directly into memory without disturbing the CPU.

2.2 *Analysis*

Analysis could be carried out on a single sensory sample, or on a sequence of samples. Data may also be collected from a single or multiple sources. Before the analysis is carried out, some cleansing of the data may be required to remove noise and to ensure that the data is not erroneous. As a simple example, consider an ultrasound based diesel level monitoring sensor mounted within the fuel tank of a diesel generator. An ultrasound pulse is sent from the top of the tank, which is reflected by the top surface of the fuel and received back by the sensor—the time delay indicates the fuel level. The mechanical vibrations of the diesel generator cause rippling of the fuel in the tank, and this shows up as noise in the reading. A good reading of fuel level will thus require a filtering mechanism to remove this noise.[22]

Figure 3a illustrates use cases showing different combinations of number of samples and sources. Figure 3b shows one of these use cases (the polyphase energy meter) in more detail.

The genetic IoT semiconductor would need to deal with a wide variety of sensor types. Due to this a programmable microcontroller with memory comes out as the natural choice for implementing the analysis step. The sensory inputs would need to be recorded first in memory. Depending on the environmental context, the memory may need to be non-volatile so that a temporary disruption in power does not lead to loss of data. The CPU subsystem needs to have the appropriate compute power and be supported with adequate program and data memory to do the analysis.

2.3 *Actuation*

IoT devices are expected not only to sense and report physical parameters from the environment, but also to take actions where needed to control those physical parameter values. For instance, in a smart energy management situation, the energy consumption from several devices in one home would be aggregated by an intelligent control module and if this is found to exceed a threshold, then (based on a defined policy) some of the energy consuming devices will be turned off to keep the aggregate within the desired threshold.[23] In this example, the actuation would simply consist of switching off a relay. Actuation can also cause an increase or decrease in the rotational speed of an appliance by modulating the duty ratio of the Pulse Width Modulation (PWM) signal that controls it. In the event that the external device needs analog inputs for control, the IoT

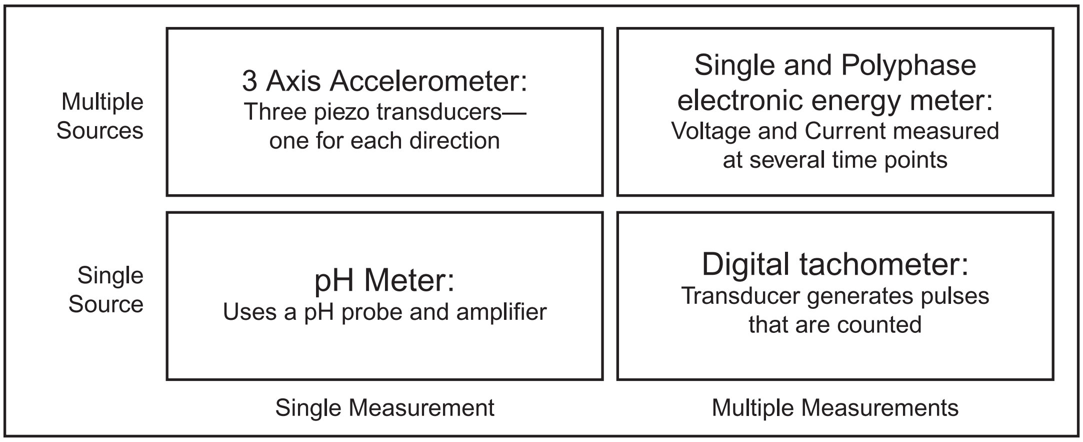

Multiple Sources
3 Axis Accelerometer:
Three piezo transducers— one for each direction
Single and Polyphase electronic energy meter:
Voltage and Current measured at several time points
Single Source
pH Meter:
Uses a pH probe and amplifier
Digital tachometer:
Transducer generates pulses that are counted
Single Measurement
Multiple Measurements

Figure 3a: Different use cases of sources and measurements.

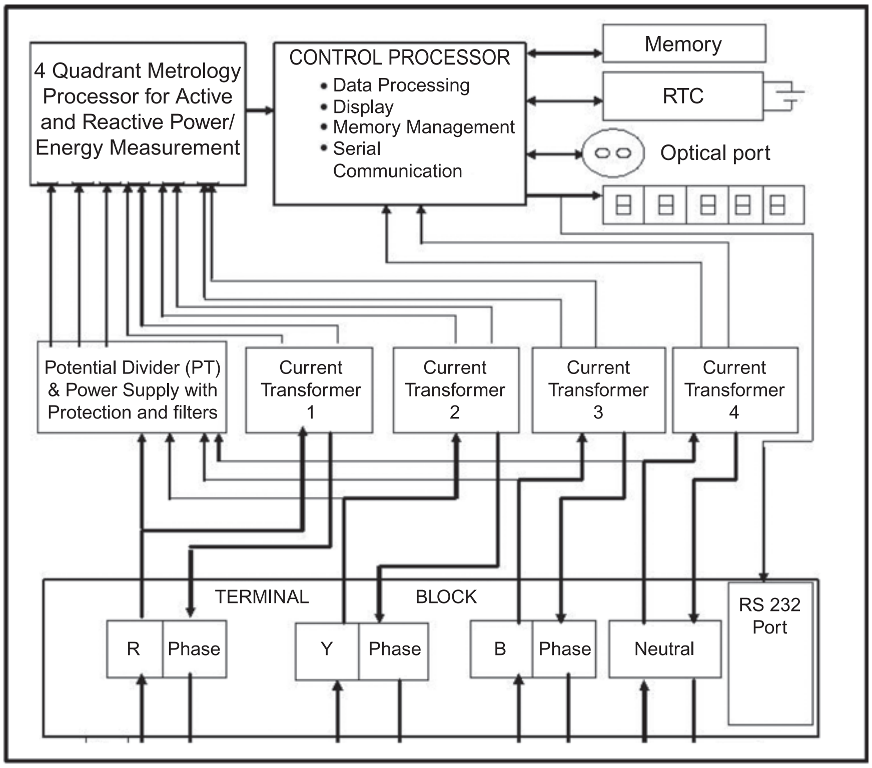


Figure 3b: Electronic polyphase energy meter.

would need to carry a Digital to Analog convertor (DAC) unit within it.

Actuation will require that some of the ports of the IoT chip be configurable as outputs, with the ability to source or sink a certain amount of current (depending on how the external circuitry is configured). In the event that analog signals need to be generated for control, the output ports must be configurable as analog ports with the ability to source sufficient current at each analog voltage level of the output. The IoT semiconductor would also need to support a variety of timers for producing different kinds of waveforms, especially PWM signals. Again for energy efficiency, it would be desirable to produce these actuation outputs at the output ports of the IoT chip without loading the CPU, memory or internal buses of the IoT chip. In other words, when required, the CPU subsystem could be switched off without disturbing the external actuation sequence.

2.4 *Deploying resources*

An IoT device is typically designed to be a reactive system waiting for certain events to occur to which it would respond. These events could be from the physical environment or from the network environment (see Figure 1a). The typical way to implement this is through interrupts—each event is mapped to an interrupt service routine. To deal with events that occur concurrently, interrupts will need to be prioritized and handled by a nested vector interrupt controller. Since the system starts preparing resources to work on an event only after it has occurred, interrupt based implementations may produce unacceptable response times. In these cases, a timer based implementation may be preferred, where resources could be prepared in advance according to a schedule. This also allows better resource sharing between different events. As an example consider an IoT semiconductor device with a single ADC unit that needs to sense inputs from multiple sources according to a fixed schedule. This can be realized within the IoT by writing a custom scheduler in software running on the CPU or relying on the scheduler of a real time operating system that is ported onto the CPU.[24]

2.5 *Communication*

The mechanism through which the IoT device would communicate to a distant server is still an area that is under considerable churn. The consensus view seems to be that two-layer network architecture is best suited for IoT, where the IoT device would first communicate to a hub or gateway which then routes the information to the remote service.[25] Taking cost and energy considerations into account, the short range wireless link from the IoT device to the hub is best implemented using an unlicensed frequency (there are several unlicensed frequencies available in the so called ISM or Industrial, Scientific and Medical bands). The long range communication from the hub to the remote site can use a conventional long distance data connection from an ISP provider or from a cellular operator.

The IoT device typically integrates a transceiver that includes all of the essential RF circuitry—filters, amplifiers, mixers, modulator/demodulator etc. This is usually designed to operate at any ISM band frequency below 1 GHz (called the sub-GHz ISM bands) or else at the 2.4 GHz ISM band. To increase the reliability and capacity of the wireless channel, the device typically supports a range of modulation options including FSK, OOK, BPSK, QPSK, O-QPSK and GMSK together with some direct sequence spread spectrum (DSSS) spreading to overcome narrowband interference and signal cancellation effects of multipath fading. The difference between the transmit power and the receive sensitivity provides a first order approximation for the total RF link budget, from which the node to node range of the device can be determined. For the 2.4 GHz bands for example, this is usually about 50 to 100 m.

In order to ensure that the communication cannot be intercepted and misused, security schemes incorporating encryption have to be provided. A good choice is the Advanced Encryption Standard (AES) used in RFID due to its high security and amenability to hardware realization at low cost and power utilization.[26] There are efficient block and stream ciphers that could be used[27] but these are yet to be standardized. Secure hash algorithms (SHA) that are used in secure IP communications could be employed, but need to be made lightweight. Research on lightweight dedicated hash functions is at an early stage.[27]

Use of higher layer protocols significantly increases the interoperability of the IoT solution and reduces its development effort. Typical higher layer protocols that are supported by some of the IoT semiconductor solutions are Bluetooth, Wifi and Zigbee.

2.6 *Co-ordination*

During the last few years, there has been a lot of interest in arranging wireless sensors into a self-organizing network (called a wireless sensor network) so that they could co-ordinate among themselves to realize tasks.[28] Such co-ordination may be required for two reasons (a) to improve the quality of the sensing of the same physical environment by correlating inputs drawn from different IoT devices and (b) to create a fault tolerant mesh network that can be used for communication between each of the IoT devices and the hub (described in section 2.5 above), and from there onwards to the remote service.

The IoT device typically has a unique ID that allows it to identify its presence periodically to its neighbours using a (wireless) communication link (could be the same ones described in section 2.5). Each device then keeps a record of its neighbour nodes so they now form what is called a mesh network. Some nodes may identify themselves as masters and others may connect to them forming what is called a star network. Master nodes may link with other master nodes to form a cluster-tree, extended or dynamic star network.

Protocols for creating and managing such networks are Zigbee, Wireless HART—an extension of the established (wired) Highway Addressable Remote Transducer (HART) protocol used in industrial automation (standardized as IEC 62591), the related ISA100.11a (standardized as IEC 62734) and the 6 LoWPAN protocol which is promoted by the IP for Small Objects (IPSO) Alliance. The Zigbee alliance has also come together with IPSO to offer Zigbee IP.

These protocols can be implemented in software running on the CPU; however, this would unnecessarily load the CPU, and would also be power inefficient in the event that the network has to be kept running while the chip is mostly idling. The IoT semiconductor thus typically provides some hardware support for packet processing for such a co-ordination protocol.

2.7 *Energy conservation*

Several energy conservation steps have been discussed in each of the sections 2.1 to 2.6—in which different modules are switched off when not in use without affecting the interface functionality of the system and its response time. Most IoT devices allow each module to be turned off when its functionality is not likely to be needed for some time, with its critical data being retained in non-volatile memory. The module is said to be in a sleep mode till an event occurs that needs the use of that module at which time it is woken up (i.e. its state is restored). Due to dependency between the operations of these modules, they may be put to sleep and woken up in a defined sequence. Typically, the more complex and state rich modules (e.g. the CPU) would have the most dependencies and highest wake up time; on the other hand, it is precisely these that consume the most power. Hence the IoT device provides a number of different sleep modes which allows the right balance to be struck between power consumption and wake up sequence. This is discussed in more detail in section 3.

2.8 *Case study: Smart home energy management*

To give more concrete shape to the architecture and modular composition of the IoT semiconductor, this section takes the example of an IoT device used in the implementation of smart energy management for a home.[30,31] Smart energy is a good use case for IoT, since it sits between the physical environment (electric supply lines inside the house) on one side and the network environment on the other. At the very minimum, such a system would have a smart energy meter compliant with the requirements of Advanced Metering Infrastructure (AMI).[29] Such a meter would record energy use at different times in the day and communicate this to the remote billing server, which may use demand based billing practices (such as time of day pricing) to calculate the billed amount. Providing feedback to the user about the billing pattern is expected to lead to energy conservation behaviour.[32] A more sophisticated system would implement the full demand response and load control algorithm (see)[31] where the load control units could turn off (or turn down) different appliances in the home based on demand signals coming in from the network.

2.8.1 Semiconductor solutions for smart home energy management: Many of the big semiconductor players have recently released solutions for such smart energy management (examples are CC2538 from Texas Instruments, STM32 W108 from ST Microelectronics, KW20 from Freescale, EM357 from Silicon Labs, JN5148 from NXP, 88 MZ100 from Marvell). A quick examination of these solutions shows that their architecture is similar and contains the same set of modules—which would indicate the emergence of a de-facto standard implementation for this application. The main modules present in these microcontrollers are listed below:

a. 32 bit MCU (ARM Cortex M3 or equivalent), clocked at 8 MHz to 48 MHz with NVIC
b. 64-512 KB of Flash for programs, 8-32 KB of RAM for data (part of it non-volatile SRAM for retention in sleep modes)
c. 2.4 GHz 802.15.4 compliant transceiver, with 90+dB link budget
d. Packet processing for 6LoWPAN and different Zigbee profiles (e.g. smart energy 2.0)
e. AES 128/256 bit encryption support for security
f. Internal RC oscillators—slow (typically 32 KHz) and fast (16+MHz).
g. DMA, Timers (supporting PWM), Watchdog, Programmable I/O ports, SPI, UART, I^2C.
h. ADC, 12 bit or higher resolution, Multi-channel support, 250 Ksps to 1 Msps.
i. Programmable I/O ports: both analog and digital; 25 mA current sources/sinks.
j. Low power sleep modes with schemes for fast wake up.
k. Other features selectively present: DAC (10 bit or higher), Comparators, Temperature sensor, Battery management, DC-DC convertors.

2.8.2 Power/current consumption: Since energy conservation is a critical part of any IoT device, it is also useful to look at the energy consumption patterns of these modules. As an example, the CC2538 has the following current consumption characteristics (others are similar):

- CPU running at 32 MHz (Radio off, peripherals idle): 13 mA
- RF: Receive mode active (CPU idle): 20 mA
- RF: Transmit mode active (CPU idle): 24 mA
- ADC when converting: 1.2 mA
- Timers: 120 μA
- Sleep timer (32 KHz oscillator): 0.9 μA
- Fast wake up (4 μS wake up) mode: 6 mA
- Deep sleep, wake up with internal timer: 1.3 μA
- Deep sleep, wake up with external interrupt events: 0.4 μA

As can be readily seen, the main consumer of power consumption is the RF transmit and receive blocks, followed by the CPU. While the CPU power can be controlled by running it at slower speed (and/or at lower voltage levels), the only way to reduce the power dissipation of the radio is to use it sparingly. We discuss this further in section 2.8.4.

2.8.3 Power conservation by putting the IoT device to sleep: To get a sense of what these current consumption measures mean, a battery of typical 1100 mAh capacity can cumulatively support about 40 h of analysis tasks, 10 h of communication/co-ordination tasks leaving sufficient room for sensing, actuation and for idling. This means that if the battery has to last for 5 years, then the different modules of the IoT device has to be turned on for only about 0.1% of the year and kept idle in different sleep modes during more than 99.9% of the year, in other words it is active for only 1/1000 of the time—also called a “low duty cycle” mode of operation.

2.8.4 Power saving scheme in Zigbee: This low duty cycle mode described in section 2.8.3 is what is used to reduce average power requirement of the RF circuit in standards such as Zigbee (802.15.4) and Bluetooth Low Energy (BT 4.0). A typical Zigbee device is expected to operate with 0.1% to 1% of the time, enabling it to operate with a single battery for 5 years.[33]

Zigbee networks are formed from two kinds of devices—full function devices (FFD) and reduced function devices (RFD), both carrying 64 bit addresses. RFDs can connect only to FFDs which act as coordinators to manage the activity within the network or as routers to forward packets from other nodes. Zigbee networks can operate in one of two modes—non-beacon or beacon mode. In the beacon mode, the coordinators periodically wake up and send beacons to the other nodes in the network. When nodes receive a beacon, they wake up to check if there are any incoming messages which require processing. If not, the nodes and coordinators go back to sleep again. This ensures a low duty cycle mode of operation. In the non-beacon mode, some devices are always active (typically the ones that are powered from the mains), while the others sleep. The coordinators and routers cannot sleep in the non-beacon mode, since any node may wake up suddenly and decide to talk to one of them. In the sleep mode, the RF module is turned off and only the internal (32 KHz) oscillator is kept running to keep the power consumption really low.[33] Zigbee specification demands that a node needs to wake up within 15 ms from its sleep state in order to provide an acceptable response time.

3 Activity Management Using Low Energy Event Detection Schemes

As seen at the end of the above section, the IoT device is mostly in an inactive (sleep) state and wakes up only when required to do an activity. This leads to a significant design problem—because when an event occurs that requires a response, the IoT system must first spend time in waking up from its sleep state, and then process the event to produce a response. A large wakeup time slows the response time of the system, and it is quite possible that the IoT device may miss the event altogether. Thus, an IoT device typically accommodates a number of different power (sleep) modes that trades off wakeup time with power consumption.

As an example, the STM32 W108 offers four sleep modes:

- Idle Sleep: CPU is in idle state waiting for an interrupt (CPU, interrupt controller and peripherals not powered down). Consumes 2 mA if CPU is clocked at 6 MHz with memory off and up to 7.5 mA when CPU is clocked at 24 MHz with memory kept powered on. This allows the system to respond within a few clock cycles to external events.
- Deep sleep 1: CPU is fully powered down with only a 32 bit timer running at 1 KHz as a sleep timer, which wakes up the CPU after the preset duration. CPU will also wake up by an activity on a programmable I/O pin. Current consumption drops to 1 μA. Time to go to sleep is 5 μs and wakeup time is 110 μs.
- Deep sleep 2: Sleep timer also turned off. Can be woken up only by an activity on a programmable I/O pin, but current consumption drops further to 0.4 μA, without affecting wakeup time.
- Deep sleep 0 (also known as emulated deep sleep): CPU powered up, but peripherals except the system debug components powered down to permit debugging. Current consumption is 300 μA.

What is immediately evident is that the current consumption in idle mode could be 10,000 times the current consumption in the deep sleep mode, but the response is 100 times faster.

3.1 *Wake up schemes and the event detection problem*

Since the IoT device lies between the physical environment and the network environment, there could be three sources for the signal that wakes up the IoT device from sleep state:

- ***From the internal sleep timer***: This clearly is the choice to use if the IoT device needs to process something periodically (say once every hour for example). The beacon mode of operation in Zigbee may be simply implemented through the sleep timer. Its use in other situations becomes limited, as there is no way to determine what should be the preset sleep time. If it is set too long, some important activity in the physical or network environment may be missed. If it is set too short, then the device wakes up often to find no activity to perform and goes back to sleep.
- ***An event from the physical environment that needs to be processed***: For instance a physical

parameter moves beyond its safe range of operation generating an alarm event which is connected to one of the programmable I/O pads to wake up the device from its sleep state. The issue with this approach is that it needs an external (electronic) circuit to generate the event, and this circuit is always powered on, negating all the benefits of being in the sleep mode. If the internal A/D and comparators are used to detect the event, then these modules need to be always powered on—which means that the device cannot be put fully in the sleep mode.

- ***An event from the network environment that needs to be processed***: For example, the network requests the device to take a physical reading, which would require the device to first wake up. The problem with this is that the network is not connected via a programmable I/O port of the device, so it does not create any activity there. The communication network occurs through the radio interface module which needs to be kept powered on if the device has to recognize a request from the network side. However, the radio module is the most power hungry component in the device and so cannot be powered on all the time.

In other words, all three methods of waking up the IoT device have problems—the timer method cannot predict external events well, waking from the physical environment requires the sensor circuit to be always on, while waking from the network environment requires the radio circuit to be kept always on. Keeping the entire sensor circuit or the entire radio always on introduces unnecessary activity within the IoT device. However, if the sensor or radio is not on, then it may not be possible to recognize the external event. What is needed is a circuit that lies in between these OFF-ON extremes which can recognize the external event, but which consumes only a fraction of the power of the sensory or RF circuits.

The problem described above can be summarized as follows:

To ensure that the activity within the IoT system is matched closely to the activity to be monitored in the environment, is it possible to design a minimal always on low power circuit whose function is restricted to recognizing external events?

3.2 *Wakeup through a bootstrap process*

One way to implement wake up is through a bootstrap process depicted in Figure 4a. The small always on circuit ActivityDetect recognizes some external activity and triggers the wakeup of next level circuit EventDetect. This circuit further processes the incoming signal to determine if it is a legitimate event that requires handling, which in turn powers up the actual module within the IoT—ADC, Comparator, Radio transceiver, CPU, etc. as needed.

ActivityDetect would need to operate at extremely low current since it is always on. EventDetect would need to operate at current levels comparable to the deep sleep state (1 μA). If possible, EventDetect should be powered up by the incoming signal to conserve energy. This can be realized if ActivityDetect is implemented as a circuit that accumulates energy from the incoming signal to switch on and provide power to EventDetect. This works because the IoT device has a low duty cycle mode of operation where legitimate events happen only intermittently.[8] A simple circuit to accumulate power is a charge pump circuit depicted in Figure 4b. This consists of a sequence of zero bias schottky diodes each charging the storage capacitor, which act to step up the low incoming voltage signal till it reaches the required threshold needed to trigger the switching on of the next stage.

3.3 *Sensor driven wakeup through bootstrap*

A method that uses such cascaded detection for monitoring the presence of an endangered bird, the Golden-cheeked Warbler, is presented in.[10] The method leverages the knowledge that the event is rare to get 12 to 20x energy reduction by first doing a rough energy detection of the incoming audio signal and deciding whether to run a more detailed signal processing algorithm for detecting if it is the rare bird. Other adaptive signal processing methods have been proposed earlier that employ a recursive scheme to incrementally refine the analysis of the input signal till the energy budget runs out.[34]

However, these are software algorithms that would need the I/O, ADC and the CPU running (maybe at lower clock speeds), and would not yield the three or more orders of magnitude energy savings being visualized in this paper. Hardware schemes for designing low power wakeup from sensors are described in[8]—these had only limited success. Quoting the authors of,[8] "Low-power wakeup sensors … represent promising areas of innovation since they extend the event-driven hierarchy into hardware". Further research would be needed to find more innovative hardware schemes for such bootstrapped sensing.

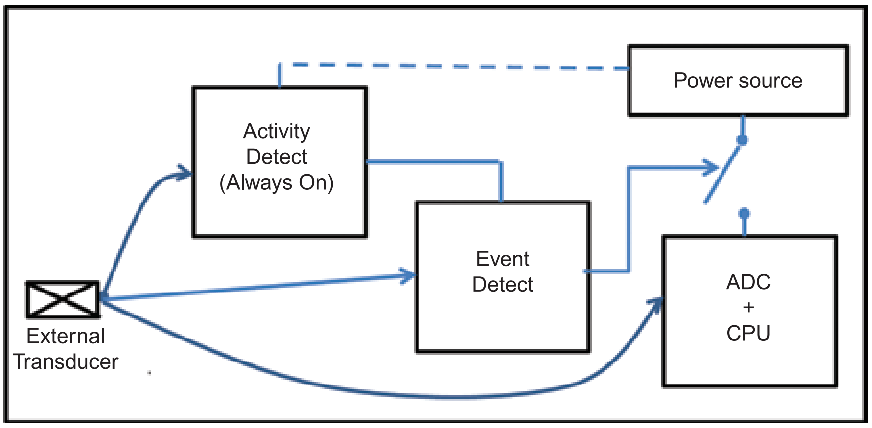


Figure 4a: Bootstrap event detect.

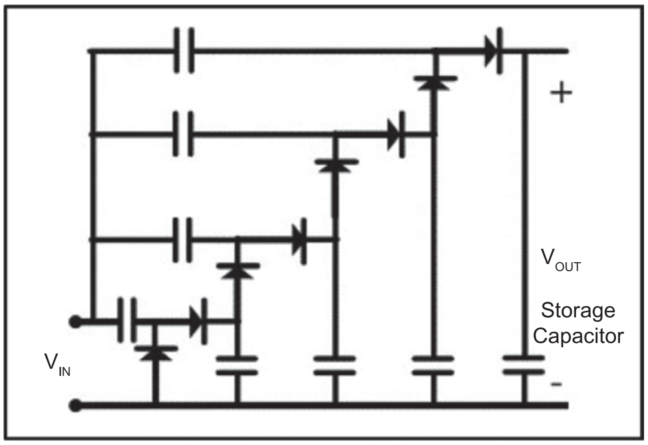


Figure 4b: Charge pump for activity detection.

3.4 *Radio driven wakeup*

As compared to sensor driven wakeup, designing a low power wakeup scheme from the radio receiver has received significantly more attention.[9] In the WLAN space, the use of an "out of band" radio channel to wake up the main WLAN card has been proposed for energy saving.[35] Many of the wireless communication systems use CSMA (Carrier Sense Multiple Access) in which each device would need to first check the channel to see if someone else is transmitting before they begin to transmit on that channel. Zigbee too needs a CSMA scheme especially for the non-beacon mode (in the beacon mode it is possible to avoid contention by using a TDMA scheme where each node is assigned a distinct time slot for transmitting). Since the wireless transceiver supports carrier sense, it should be possible to use this circuitry also to wake up the transceiver. However, this means that the carrier sense circuit needs to be kept on all the time, which may drain the power source.

The focus is therefore to design these wake-up modules in such a way that they are highly power efficient.[9] A simple way to implement a power efficient scheme would be based on the bootstrap architecture described in figure of section 3.2. The ActivityDetect would need to detect energy level of the carrier and the EventDetect will need to process the carrier signal a bit more to determine if the incoming signal is legitimate. The most basic circuit for detecting radio activity is shown in Figure 5a.[9] The antenna reacts to the incoming radio wave, producing a signal that is rectified by the diode D (having a low bias threshold) to produce the output voltage V_{OUT}. The problem with this basic circuit is that the radio strength needs to be pretty high (0.08 mW) for the output voltage to be sufficient to trigger the next stage (say 0.6 V).

Modern transformer technology can be leveraged to improve upon this as depicted in Figure 5b.[9] The circuit at the input of the transformer acts as a tuner which is set to the carrier frequency. This reduces the probability of false positives (i.e. when an activity is detected that is not an event). Over a period of several milliseconds the storage

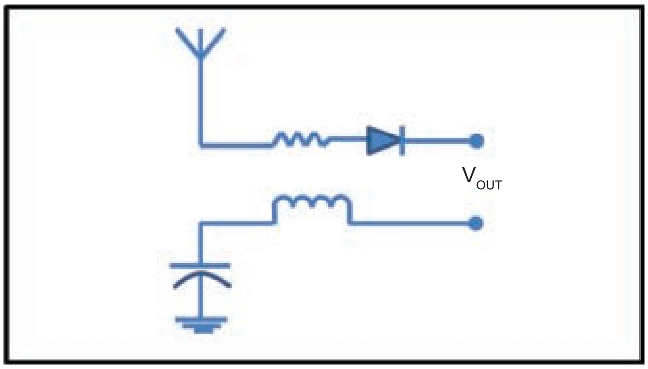


Figure 5a: Simple activity detection circuit.

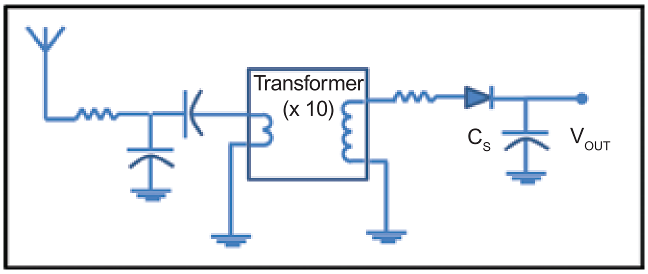


Figure 5b: Improved activity detection circuit.

capacitor C_S accumulates sufficient voltage to be able to trigger a transistor that turns on the next stage.

A much better designed wake up circuit operating at just 2.3 μA from a 1 V supply is described in.[36] It realises a RF voltage transformer using a metal pattern on 0.508 mm thick FR4 laminate and uses a transistor based envelope detector whose signal is amplified by an extremely low noise baseband amplifier to produce the desired triggering signal. However, even this best-in-class design requires a radio input at −47 dbm to generate the necessary triggering signal; this means that considerable research is still required to make this workable at the much lower signal levels (< −90 dbm) that are received by the radio receivers.

4 Asynchronous Design Methodologies for IoT

The IoT semiconductor architecture given in Fig 1b and further detailed for the smart home energy application in section 2.8.1 shows the presence of multiple clock domains within the chip. At the very least, there is a low speed (and low power) internal clock required for the deep sleep mode. The network interface (for example Zigbee) has its own clock requirement. The ADC used in the sensing interface needs a clock operating at its sampling rate. The actuation may need to generate pulses which may require its own clock. In principle, many of these clocks can be derived from a master clock, but this may not be the most optimal design from an energy use perspective.

The notable difference between IoT systems and other embedded systems is that they operate at a very low duty cycle—i.e. they are asleep for most of the time; they wake up to do tasks for a very short period and then go back to sleep. In the sleep mode, the IoT system is either completely switched off, or only its low speed clock is kept running. When the system wakes up, the main clock domains of the chip would need to be started up and only then can the device put to use.

Wake up of the clock domains could be implemented using basic circuits described in sections 3.2 to 3.4, but it is likely that further advances in wakeup logic will use complex digital logic. If so, then this logic would need to be clockless, and so have to be designed according to the principles of asynchronous design.[37] This gives rise to the question whether asynchronous design principles[38] could be used to realize more parts of the IoT system than just the wakeup circuit, thereby avoiding the use of many or all of the clock domains.

One advantage of an asynchronous design methodology is that it does not use a clock and is hence insensitive to conditions which normally affect the clock speeds—environmental conditions such as temperature being a major one. A more significant one is its natural fit with dynamic voltage and frequency scaling (DVFS) methods used to conserve energy.[39,12] In DVFS, the supply voltage of the circuit is reduced which in turn reduces the frequency of operation of the circuit and these together contribute to significantly scaling down the power required to operate the circuit.[39] Asynchronous circuits could thus be designed to gracefully reduce and increase power according to the operational needs. Finally, asynchronous circuits are more tolerant to working with unreliable power sources (for example a rundown battery or harvested energy).

The following sections review two examples of the use of asynchronous methodologies for IoT design which serve to highlight that this could be a fruitful direction for future research.

4.1 *Asynchronous microcontrollers fare better under volatile workloads*

The use of asynchronous design methodologies for digital circuit design has received plenty of attention over several decades starting from the classical work of Muller[40] in the 50's, and came centre stage with Charles Seitz's chapter on system timing in the popular book by Mead and Conway on VLSI systems.[41] Asynchronous methodologies have found practical applications in designing systems where clocked local subsystems communicate asynchronously with each other—also called globally asynchronous and locally synchronous (GALS) systems.[38] On the other hand, academic interest has been high in the design of clockless asynchronous circuits, specifically in a class of circuits called quasi-delay insensitive (QDI) because these circuits can be constructed compositionally and be derived automatically from high level specifications.[38]

QDI circuits have found relatively little commercially success, mainly because synchronous circuits are so much easier to design and analyse using conventional CAD tools. The most focussed effort to find commercial use has been in the area of QDI microcontrollers which have gone through three generations of evolution from the earliest ones designed at Caltech in the late 80's to the more recent ones that implement the full functionality of the 8051, MIPS R3000 and the ARM microcontrollers.[42]

A recent paper[12] explores the reasons why an asynchronous microcontroller might be superior to a synchronous one. In order to do an equitable comparison, the authors fabricate both a synchronous S8051 and an asynchronous A8051 on the same die using 130 nm CMOS technology. A significant observation of their experiment is that under normal conditions both versions have comparable speeds and energy, with the A8051 producing ~12 dB lower electromagnetic interference but occupying ~2x the die size of the S8051 (i.e. no significant energy savings are seen under normal conditions). On the other hand, when there were wide variations in process, voltage or temperature (PVT) conditions or when there was large volatility in the processor load, then the A8051 significantly outperformed the S8051 in energy dissipation. Figure 6 (taken from)[12] compares the energy per computation (EPC) under different loading conditions which shows 40–50% improvement for the asynchronous logic at low loading.

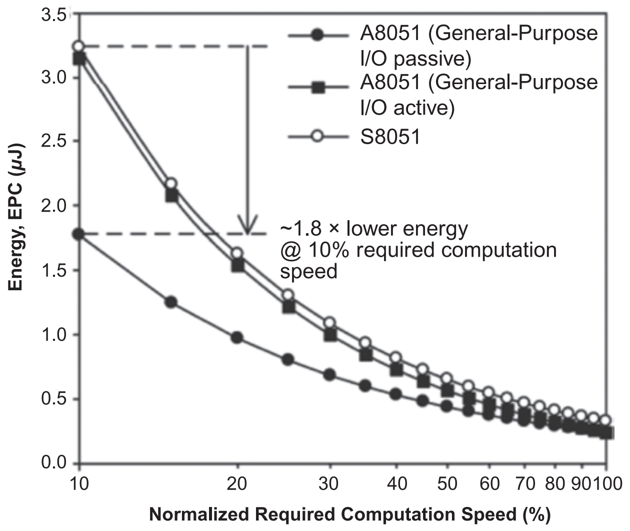


Figure 6: Performance of A8051 and S8051 at different loads.

4.2 *Asynchronous method for voltage sensing*

Sections 3.2 and 3.3 discussed a method of bootstrapping the detection of an external event that would then trigger the wakeup of the ADC circuitry within the IoT device so that it can carry out a precise measurement of the sensed input. What if however, a precise measurement is not really required and an approximate estimate of the sensed input is sufficient? In this case, would it be possible for the entire ADC process to be completed using only the energy generated from the external transducer? This interesting possibility is explored further in,[37] though the authors only propose its use for sensing the voltage level of an unpredictable power source—for example the storage capacitor in the output stage of an energy harvesting circuit (such as the charge pump in Figure 4b).

The way the circuit works is as follows: A (small) proportion of the input energy is captured in a capacitor C_{sample}, with this captured energy being directly related to the voltage being measured (sensed). The energy stored in C_{sample} is then used to run an asynchronous counter which drains C_{sample}. As energy is getting drained, voltage level on C_{sample} falls, and the asynchronous counter automatically slows down to adjust to the lower voltage. When the voltage falls below a certain threshold, the counter is stopped. The expectation is that the counter value would reflect in some way the voltage level on C_{sample}, which being related to the voltage on the sensed external transducer would therefore give an approximate measurement of the sensed input. The asynchronous counter itself is realized by sequencing four asynchronous toggle logic units. Figure 7 (taken from)[37] shows the measured output signals at the outputs of these toggle units. The rapid slowing down of the counters is clearly visible. Since the count value will have many dependencies on process and environmental factors, the authors don't propose this as a way to replace the ADC itself, but as a means to measure energy levels. However, with further research focussed on designing circuits which can compensate for these dependencies, it may be a contender to replace the ADC at least in applications where a precise measurement is not needed.

5 Harvesting Energy from the Environment

The prospect of having 100s of billions of IoT devices scattered around the world each running with a battery is a scary one from an environmental point of view. What happens to those batteries when they run out? This raises the question whether it is possible to avoid batteries by running the IoT system entirely using energy harvested from its operating environment.

What makes this proposition a worthwhile one to look at is the fact that the IoT system operates with a low duty cycle, which means it needs its energy in small bursts while it may be able to draw a sustained amount of energy from the environment all the time. For instance an IoT device that is awake 0.1% of the time, consuming 50 mW average power in its wake mode but only 1 μW of average power in its sleep mode, would need to be supplied with an average total power of 51 μW (see[14] for a more detailed analysis of such energy tradeoffs). As discussed in section 2.8

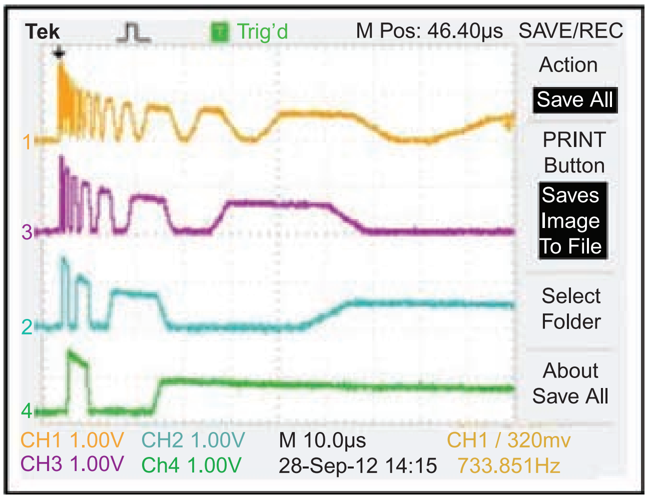


Figure 7: Output of the 4-bit asynchronous counter.

these values are typical for the present generation of IoT semiconductor systems, and the general consensus is that a power source producing about 100 μW would serve the purpose of keeping the IoT device running without batteries. This is the kind of power produced by the solar panel in a cheap calculator in a normally lit office or home environment, which gives encouragement that a cheap solution for energy harvesting should be possible.

5.1 *Wearable computing and energy harvesting*

The design of wearable biosensor systems that can assist in healthcare have been an active area of research for many years.[43] Mainly motivated by rising healthcare costs, these systems envisage IoT devices to be seamlessly integrated into items of daily wear to facilitate unobtrusive all day and any place health, mental and activity status monitoring.

More recently, wearable computing has taken a more consumer oriented turn with the advent of augmented reality devices like Google glasses that allows a whole host of context metadata to be overlaid on physically observable objects so that one can be much better informed about these objects before interacting with them.[44] Another interesting direction is the use of wearable computing in human robot interaction.[45] However, these consumer oriented applications seem to fall outside the intended scope of IoT.

Wearable computing is the one place where energy harvesting is most likely to be immediately applicable. That the human body is a tremendous source of energy was recognized many years ago—the author in[46] estimated nearly two decades ago that 200–900 mW can be harnessed from body heat, 400 mW from human breath, 20–30 mW from blood pressure, 300 mW from hand motion, 10s of mW from finger motion and several watts from leg motion. All these sources are clearly sufficient to power a low duty cycle IoT device embedded within an item of daily wear.

5.2 *Sources for energy harvesting*

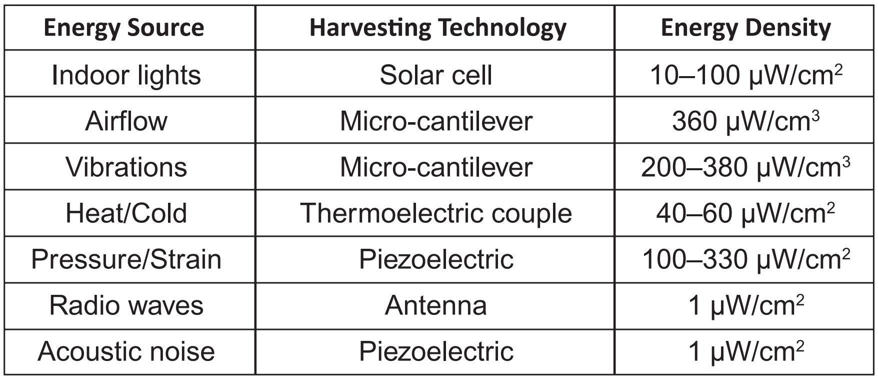
A number of sources have been investigated for energy harvesting, especially keeping in mind the places where the IoT device is likely to be implanted.[15,47] The possible energy harvestable from different sources is tabulated in Figure 8 below (other than the human sources described in the last section):

Sources such as sound and radio waves are not particularly attractive for energy harvesting generating barely 1 μW at dimensions that are typical for IoT devices. The exceptions are when these energy sources are enhanced with the specific aim of transferring energy as in the case of RFID where the short distance operation induces a much higher electromagnetic field at the receiver (−10 to −20 dbm) than would commonly be available during communication (−70 to −110 dbm).

5.3 *Schemes for harvesting energy*

A scheme of using a charge pump to harvest energy directly from an external transducer has been discussed earlier in section 3 (see figure 4b). Similarly, a scheme to pick up energy from a radio signal is described in Figures 5a and 5b.

Since micro-cantilevers are attracting a lot of recent research attention, this section briefly reviews a scheme described in a recent paper[48] for harvesting energy from airflow or vibrations using a micro-cantilever. A key point that should be carefully observed is that there is a significant difference between the energy density measurements



Energy Source	Harvesting Technology	Energy Density
Indoor lights	Solar cell	10–100 μW/cm^2
Airflow	Micro-cantilever	360 μW/cm^3
Vibrations	Micro-cantilever	200–380 μW/cm^3
Heat/Cold	Thermoelectric couple	40–60 μW/cm^2
Pressure/Strain	Piezoelectric	100–330 μW/cm^2
Radio waves	Antenna	1 μW/cm^2
Acoustic noise	Piezoelectric	1 μW/cm^2

Figure 8: Energy harvesting potential of different sources.

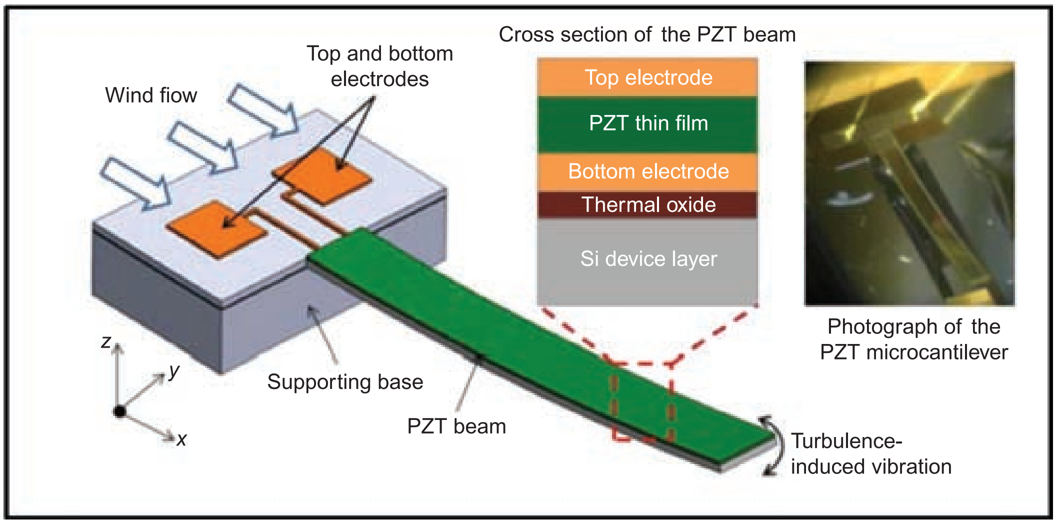

Cross section of the PZT beam
Top and bottom electrodes
Wind flow
Top electrode
PZT thin film
Bottom electrode
Thermal oxide
Si device layer
Photograph of the PZT microcantilever
z
y
x
Supporting base
PZT beam
Turbulence-induced vibration

Figure 9: Micro-Cantilever construction.

reported by different researchers (and tabulated in Figure 8 above) and the actual energy delivered by the system. For instance the statement that a cantilever system can be designed with an energy density of 300 μW/cm^3 does not automatically mean that if space of 1 cm^3 is available within the IoT device then a harvester could be squeezed into that space to produce 300 μW.

Figure 9 (taken from)[48] shows a 3 mm x 0.3 mm piezoelectric (PZT) beam of thickness 8 μm forming the cantilever. When air flows along the length of the cantilever beam it starts to vibrate. At its resonance frequency of 650 Hz, the peak rms voltage is achieved, which is able to deliver a power of about 1 nW at a matched resistance of 100 kΩ. This would mean that to produce a power of 100 μW, 100,000 such micro-cantilevers would be needed! The reason the energy density is reported at 100 + μW is because the volume of the device taken for calculating energy density is the volume of only the PZT beam which is 0.000072 cm^3. Since allowance would need to be given for the remaining structures, or for the vibration of the beam, or to provide channels for the airflow, the expected number of cantilever beams that could be placed in a 1 cm^3 may be no more than 100, producing a much lower power of 100 nW, i.e. only 1/1000 of what we desire. Clearly, this is an area that still needs significant more research before it becomes a viable scheme for energy harvesting.

6 Printed Electronics and IoT

A key aspect of IoT devices is that they are expected to be embedded within objects of daily use. For example, in a "smart packaging" application,[16] an IoT device is to be embedded into the package that carries food and beverage items of daily consumption; the packaging is "smart" because it can signal the level of freshness of the food stored inside.[51] The traditional method of providing such an indication is to print the manufacturing and expiry date on the package—however, this does not say much about the conditions under which the item was stored and transported which may have significantly degraded its "freshness". To make the example more specific, say Rs 100 (about 2 US$) is the price of the consumable, then its packaging cost will be a fraction (say Rs 5 or 10 cents); cost of making it smart would have to be a smaller fraction (say Rs 2 or 4 cents). Conventional silicon based technology would find it hard to meet these price points unless the parts required are standardized and produced in billions of units.

This is where printed and flexible electronics could come out as a winner. In printed electronics, conventional printing processes are modified and used with specially prepared inks to print electronic circuits, much as the printing of a book in a bulk printing press or a document on the domestic printer. Since the labels (and expiry date markings) on the packages would need to be printed in any case, the question is: can this process be augmented so that the labels are made "smarter" by adding electronics to it by using inks whose costs are only marginally higher than conventional inks. Note that the electronic circuit is printed on materials traditionally used in packaging—such as plastic, paper, cloth, etc. and hence would not in any way impair its flexibility or handling. Smartness is thus seamlessly embedded into the object—a feature that could become a major selling point for IoT.[17]

There is another benefit of using printed electronics. The issue of leaving trillions of hazardous spent batteries lying around in the environment was discussed in the previous section which argued that energy harvesters could be used instead. But what if the environment of operation does not have any steady source from which energy can be harvested cheaply? Printed electronics may come in handy here to produce cheap disposable batteries using environmentally friendly material. There is a lot of groundwork that has been done in micro-combustion systems including microreactors[49] which could also play a role in the creation of alternative energy sources for such devices.

6.1 *Limitations of printed electronics*

Printed electronics has a very large number of limitations at the current state of the art which all need to be addressed through more research before it can become a viable technology that lives up to the promise discussed in the previous section.

First consider the main value proposition of cost: while the printers needed for producing the circuits have become cheaper and can print on low cost commonly available substrates (such as PET and PEN sheets), the organic and inorganic materials and solvents required for printed electronics are still very expensive with prices many orders of magnitude higher than those of the inks that are used for printing.[54] However, with wider usage of printing processes, material costs are likely to come down dramatically over a period of time.[17]

Coming now to the question of performance: while a reasonable scale has been demonstrated in terms of number of transistors in a single printed integrated circuit, the speed of these circuits are still in 10s of Hz.[50] The only instances where higher levels of performance have been demonstrated is when the traditional (and expensive) photolithography technique is used,[53] or if one falls back to silicon circuits (but produced using printing techniques).[54] The speed of printed electronics can be improved manifold if high mobility materials are used (synthesized from graphene for example),[52] or if the deposited material is made more crystalline through post printing treatment, or if the feature sizes are reduced by using innovative printing processes.[54] Even with the current level of performance, a number of simple circuits are realizable, such as the one considered here—smart packaging.[51]

Thirdly, printed electronics does not produce devices with uniform characteristics and there is considerable variation between the performances of different devices printed on one circuit. However, process variations are now a problem even with small feature silicon devices[55] and when circuits are operated near the threshold voltage to reduce power dissipation.[56] Clever design methodologies including asynchronous ones would be required to overcome the negative impact of process variation.

Finally, printed and organic electronics are not stable due to oxidation by the atmosphere and other chemical effects. This means further treatment of printed electronic surfaces may be necessary to increase their life. In applications like smart packaging this is not likely to be a concern, since the shelf life of the packaged item itself may only be a few weeks.

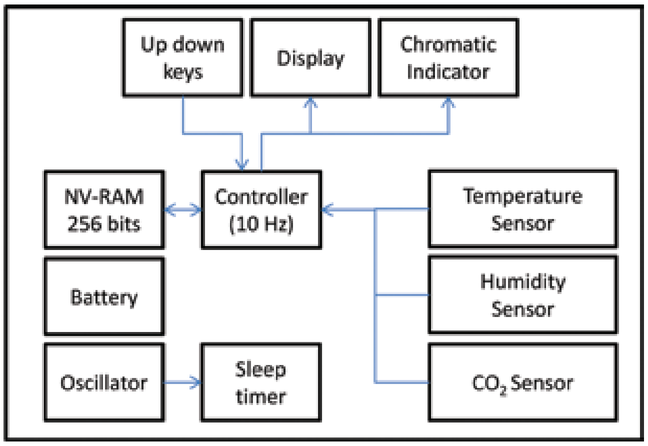


Figure 10: Architecture of a smart packaging system.

6.2 Electronics required for the smart packaging application

A rough architecture of the electronics that realizes a smart packaging application is depicted in Figure 10.

At the current state of the art, each of the components that are required in this application can just about be realized using printing techniques (see for example).[50,51]

Any circuit more complex than this would require more research on ink materials, printing processes and design techniques.

7 Conclusions

This paper develops the top level requirements for an IoT semiconductor and the modular architecture needed to realize it. IoT design needs a multidisciplinary approach, and so all the different topics are introduced at an introductory level so that any reader can get a quick understanding of the issues involved in realizing an IoT semiconductor. Four specific topics are then taken up for discussion at some depth keeping in mind two considerations—that these should have a strong bearing on IoT semiconductor evolution, and should have significant potential for future research. These four areas—bootstrap wakeup, asynchronous design, energy harvesting and printed electronics look very disparate and unrelated at first sight; however, there is strong inter-relationship between these disciplines which would need to be explored if they have to collectively make a much stronger impact on the domain of IoT semiconductors. Bootstrap wakeup and energy harvesting would both need to be clockless and would have to operate under volatile and uncertain environments thus needing asynchronous design. The printing processes used in printed and flexible electronics do not yield devices with uniform characteristics, and this process variability would again drive the design towards asynchronous methodologies. Printing allows exploitation of novel materials to realize superior devices, and these could be better integrated with the objectives of event detection and energy harvesting. Consider for example the use of specific devices for event detection and recording, or embedding energy storing devices all over the circuit for reclaiming spent energy, which is yet another source for energy harvesting.

Received 7 July 2013.

References

1. J. Manyika et al., "Disruptive Technologies: Advances that will transform life, business and the global economy", McKinsey Global Institute, May 2013. http://www.mckinsey.com/insights/business_technology/disruptive_technologies
2. OECD (2013), "Building Blocks for Smart Networks", OECD Digital Economy Papers, No. 215, OECD Publishing. http://dx.doi.org/10.1787/5k4dkhvnzv35-en
3. L. Atzori, A. Iera, G. Morabito, "The internet of things: A survey", Computer Networks, Volume 54 Issue 15, October, 2010, pp. 2787–2805.
4. MQ Telemetry Transport, http://mqtt.org
5. David Boswarthick et al. (eds), *M2M Communications: A systems approach,* John Wiley and Sons, UK, 2012.
6. G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, "Energy conservation in wireless sensor networks: A survey", Ad Hoc Networks, No 7, 2009, Elsevier, pp. 537–568.
7. Jia-Ming Liang et al., "An Energy-Efficient Sleep Scheduling With QoS Consideration in 3GPP LTE-Advanced Networks for Internet of Things", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Vol. 3, No. 1, March 2013, pp. 13–22.
8. P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," in Proc. 4th Int. Symp. Inf. Process. Sensor Networks, 2005.
9. L. Gu, J.A. Stankovic, "Radio-triggered wake-up for wireless sensor networks", Real-Time Systems, 2005, Springer.
10. David Jun and Douglas L. Jones, "Cascading Signal-Model Complexity for Energy-Aware Detection", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Vol. 3, No. 1, March 2013, pp. 55–74.
11. Jason Hill et al., "System architecture directions for networked sensors", ACM SIGOPS Operating Systems Review, Volume 34 Issue 5, Dec. 2000, pp. 93–104.
12. Kok-Leong Chang et al., "Synchronous-Logic and Asynchronous-Logic 8051 Microcontroller Cores for Realizing the Internet of Things: A Comparative Study on Dynamic Voltage Scaling and Variation Effects", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Vol. 3, No. 1, March 2013, pp. 23–34.
13. Vipul Chawla and Dong Sam Ha, "An overview of passive RFID", IEEE Applications & Practice, September 2007, pp. 11–17.
14. A. Kansal, et al., "Power Management in Energy Harvesting Sensor Networks", ACM Transactions on Embedded Computing Systems, Vol. 6, No. 4, September 2007, Article 32.
15. S. Sudevalayam and P. Kulkarni, "Energy Harvesting Sensor Nodes: Survey and Implications", IEEE Communications Surveys & Tutorials, Vol. 13, No. 3, 3rd Quarter, 2011, pp. 443–461.
16. Mantysalo M., "System integration of smart packages using printed electronics", Proceedings of 62nd IEEE Electronic Components and Technology Conference (ECTC), 2012, pp. 997–1002.
17. Lupo D. et al., "OE-A Roadmap for Organic and Printed Electronics", in *Applications of Organic and Printed Electronics*, ed. E. Cantatore, Springer, 2013.

18. Eugene Y. Song and Kang Lee, "Understanding IEEE 1451—Networked Smart Transducer Interface Standard", IEEE Instrumentation & Measurement Magazine, April 2008, pp. 11–17.
19. http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html
20. Robert H. Walden,"Analog-to-Digital Converter Survey and Analysis", IEEE Journal on Selected Areas in Communications, Vol. 17, No. 4, April 1999. pp. 539–550.
21. Naveen Verma and Anantha P. Chandrakasan, "An Ultra Low Energy 12-bit Rate-Resolution Scalable SAR ADC for Wireless Sensor Nodes, IEEE Journal of Solid State Circuits", Vol. 42, No. 6, June 2007. pp. 1196–1205.
22. D.C. Gazis et al., "Ultrasound Liquid Level Gauge for Tanks Subject to Movement and Vibration", US Patent No. 5,793,705, Aug. 11, 1998.
23. Michael LeMay, Rajesh Nelli, George Gross, and Carl A. Gunter, "An Integrated Architecture for Demand Response Communications and Control", Proceedings of the 41st Hawaii International Conference on System Sciences—2008.
24. F. Balarin, L. Lavagno, P. Murthy and A. Sangiovanni-Vincentelli, "Scheduling for embedded real time systems", IEEE Design and Test of Computers, Jan-Mar 1998. pp. 71–82.
25. Qian Zhu et al., "IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things", IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC), Hong Kong, Dec. 2010, pp. 347–352.
26. M. Feldhofer, S. Dominikus, J. Wolkerstorfer, "Strong authentication for RFID systems using AES algorithm", Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, Cambridge, MA, USA, August 2004.
27. Roman R. et al., "Securing the Internet of Things", IEEE Computer, Vol. 44, Issue 9, Sep. 2011. pp. 51–58.
28. Akyldiz I.F. et al., "A Survey on Sensor Networks", IEEE Communications Magazine, August 2002, pp. 102–114.
29. "Advanced Metering Infrastructure", National Energy Technology Laboratory, US Dept of Energy, Office of Electricity Delivery and Energy Reliability, Dept of Energy, US Govt., Feb. 2008 (http://www.netl.doe.gov).
30. "Smart Energy Profile 2: Application Protocol Standard", Zigbee public document no 13-0200-00, April 2013. (http://www.zigbee.org).
31. "Co-ordination of Energy Efficiency and Demand Response: A resource of the national action plan for energy efficiency", US Dept of Energy, Jan 2010. (http://www.epa.gov/eeactionplan).
32. Sarah Darby, "The Effectiveness of Feedback on Energy Consumption", Environmental Change Institute, University of Oxford, April 2006. (http://www.eci.ox.ac.uk).
33. Eduardo Casilari, Jose M. Cano-García and Gonzalo Campos-Garrido, "Modeling of Current Consumption in 802.15.4/ZigBee Sensor Motes", Sensors, Vol. 10, 2010. pp. 5443–5468. Open access—http://www.mdpi.com/journal/sensors
34. S. Nawab, A. Oppenheim, A. Chandrakasan, J. Winograd, and J. Ludwig, "Approximate signal processing", Journal of VLSI Signal Processing Systems, Vol. 15, No. 1/2, pp. 177–200, 1997.
35. E. Shih, P. Bahl, and M.J. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices", MOBICOM, Sep. 2002.
36. Emil Nilsson and Christer Svensson, "Ultra Low Power Wake-Up Radio Using Envelope Detector and Transmission Line Voltage Transformer", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Vol. 3, No. 1, March 2013, pp. 5–12.
37. R. Ramezani et al., "Voltage Sensing Using an Asynchronous Charge-to-Digital Converter for Energy-Autonomous Environments", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Vol. 3, No. 1, March 2013, pp. 35–44.
38. A.J. Martin and Mika Nystrom, "Asynchronous Techniques for System-on-Chip Design", Proceedings of the IEEE, Vol. 94, No. 6, June 2006, pp. 1089–1120.
39. L.S. Nielsen et al., "Low-Power Operation Using Self-Timed Circuits and Adaptive Scaling of the Supply Voltage", IEEE Transactions on VLSI Systems, Vol. 2, No. 4, Dec. 1994, pp. 391–397.
40. D.E. Muller and W.S. Bartky, B., "Theory of asynchronous circuits", in the Proceedings of the International Symposium on the theory of switching, 1959, pp. 204–243.
41. C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.
42. A.J. Martin, M. Nystrom, and C.G. Won, "Three generations of asynchronous microprocessors", IEEE Design and Test of Computers (Special Issue on Clockless VLSI Design), Vol. 20, No. 6, pp. 9–17, Nov./Dec. 2003.
43. A. Pantelopoulos and N.G. Bourbakis, "A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis", IEEE Transactions on Systems, Man and Cybernetics—Part C: Applications and Reviews, Vol. 40, No. 1, Jan. 2010.
44. D.W.F. van Krevelen and R. Poelman, "A Survey of Augmented Reality Technologies, Applications and Limitations", The International Journal of Virtual Reality, 2010, 9(2), pp. 1–20.
45. "Wearable computing: Designing and Sharing Activity-Recognition Systems Across Platforms", IEEE Robotics & Automation Magazine, June 2011, pp. 83–95
46. T. Starner, "Human-powered wearable computing", IBM Systems Journal, Vols. 35, Nos. 3 & 4, 1996.
47. V.C. Gungor and G.P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches", IEEE Transactions on Industrial Electronics, Vol. 56, No. 10, Oct. 2009, pp. 4258–4265.
48. Huicong Liu et al., "Development of piezoelectric microcantilever flow sensor with wind-driven energy

harvesting capability", Applied Physics Letters 100, 223905 (2012).

49. Yiguang Ju and Kaoru Maruta, "Microscale combustion: Technology development and fundamental research", Progress in Energy and Combustion Science, Volume 37, Issue 6, December 2011, pp. 669–715.
50. Kris Myny et al., "An 8-Bit, 40-Instructions-Per-Second Organic Microprocessor on Plastic Foil", IEEE Journal of Solid-State Circuits, Vol. 47, No. 1, January 2012, pp. 284–291.
51. Intelligent Sensor Label Products from ThinFilms, http://www.thinfilm.no/products/sensor-labels/
52. Y. Wu et al., "State-of-the-Art Graphene High-Frequency Electronics", Nano Letters 12, 2012, pp. 3062–3067.
53. T. Zaki et al., "A 3.3 V 6-Bit 100 kS/s Current-Steering Digital-to-Analog Converter Using Organic P-Type Thin-Film Transistors on Glass", IEEE Journal of Solid-State Circuits, Vol. 47, No. 1, Jan. 2012, pp. 292–300.
54. J.R. Sheats, et al., "Printing Technology for Ubiquitous Electronics", Circuit World, Vol. 36, 2010, pp. 40–47.
55. S Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation", IEEE Micro, Nov.-Dec. 2005, pp. 10–16.
56. Ronald G. Dreslinski, et al., "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits", Proceedings of the IEEE, Vol. 98, No. 2, February 2010, pp. 253–266.

Dr. Venkatesh recently joined the Department of Electrical Engineering IIT Madras as the Analog Devices Chair Professor where his teaching and research interests are focused on hardware-software co-design of complex embedded systems and in the emerging area of printed electronics. He is also the Chief Technology and Strategy Officer of Sasken Communication Technologies Ltd, which is a leading R&D services provider for producers of embedded products.

Dr. Venkatesh is a graduate in electronics from IIT Madras, Ph.D. in Computer Science from TIFR, Mumbai and was a faculty member of the Computer Science & Engineering Dept of IIT Bombay for 8 years where his research interests revolved around declarative languages and their application to the design of embedded systems. He moved to the industry when Sasken was being formed, where he has been leading their technology activities over the last two decades. He has also served as an adjunct faculty of IIM Bangalore, with interests in game theory and applications to the technology industry.

Dr. Venkatesh is a fellow of the Indian National Academy of Engineers, and serves on a number of Govt and Industry committees in the areas of microelectronics and telecommunications.