Investigating Energy Consumption of an SRAM-based FPGA for Duty-Cycle Applications

Khurram Shahzad and Bengt Oelmann
Department of Electronics Design
Mid Sweden University
Sundsvall, Sweden
{khurram.shahzad, bengt.oelmann}@miun.se

Abstract. In order to conserve energy, battery powered embedded systems are typically designed with very low-power modules that offer limited computational power and communication bandwidth and therefore, are generally applicable to low-sample-rate intermittent applications. On the other hand, enabling an embedded system with a high-throughput processing resource such as an FPGA, high-throughput processing performance that is typically required in high-sample rate monitoring applications can be achieved. However, the high power consumption associated with an FPGA poses a major challenge in attaining significant lifetime for a battery-powered embedded system. In this paper, we investigate energy consumption of an SRAM-based FPGA in relation to duty-cycle applications. In order to achieve long operational lifetime in an FPGA-based embedded system, the possible options to dynamically manage the power consumption are studied and discussed. The experimental results suggest that the SRAM-based FPGA, XC6SLX16 that provides ample logic resources in relation to typical high-sample rate monitoring applications, can be used in a battery operated embedded systems while minimizing the energy consumption to 2.56 mJ for inactive duration of 235 ms or above.

Keywords. Energy optimization; SRAM-based FPGA; High-sample rate; Dynamic power management; Duty-cycling

Introduction

With the recent advancement in electronics, battery-powered wireless embedded systems have emerged as a low-cost alternative to fixed wired-based monitoring solutions in many fields. However, the power consumption of these battery-powered systems has been a challenge to achieve a significant operational lifetime. In order to minimize power consumption, such systems are typically designed using modules with ultra low-power characteristics. These low-power modules, however, provide limited computational and communication capabilities. Therefore, such embedded systems are generally suitable for low-sample rate intermittent applications, in which a small amount of data is transmitted to a remote station without any significant processing locally in an embedded system.

Nevertheless, the high-sample rate applications can also take advantage of such potentially low-cost and autonomous wireless systems, provided that these are enabled to fulfill relatively high processing and communication requirements while maintaining low-power characteristics. For example, enabling a wireless embedded system with a

high-throughput processing resource such as an FPGA, high-sample rate applications, such as vibration-based industrial condition monitoring, structural health monitoring, and image-based tracking can be realized by processing data locally in an embedded system [1]-[3]. In this approach, raw data is processed in an embedded system and only the results are transmitted wirelessly and therefore, the communication activity and the associated energy consumption are reduced. On the other hand, the energy consumption associated with processing data locally becomes a challenge to achieve long operational lifetime. However, by exploiting the hardware parallelism in an FPGA, the actual processing can be completed in a much shorter period of time as compared to that for a low-power micro-controller that processes data in a sequential manner and, following on for the rest of the duration during which an FPGA is idle, it can be switched to a low-power state so as to conserve energy.

In relation to integrating an FPGA in an embedded system, it is the static random access memory (SRAM) based FPGAs that are commonly used [1]-[9]. This is mainly because these FPGAs not only offer embedded resources such as block RAM and Multiply-and-Accumulate (MAC) units in addition to basic logic cells, but they are typically built on a relatively more advance semiconductor technology as compared to their re-configurable counterparts such as flash-based FPGAs and therefore, provide a better performance [10]. However, their relatively high static power consumption, as compared to the flash-based FPGAs, often limits their potential advantages in battery-powered embedded systems. In addition, dynamic power management and duty-cycling techniques [11]-[12] that can be applied to conserve static power when an FPGA is idle are typically limited by the energy consumption associated with the resulting re-configuration process. Therefore, it is interesting to investigate SRAM-based FPGAs so as to evaluate their feasibility for duty-cycle applications, in which power to an FPGA is turned-off and thus, requires re-configuration when it is powered-on.

The literature related to battery-powered embedded systems that integrate SRAM-based FPGAs, such as [1]-[9] is generally focused on application specific performance issues. The energy consumption of these FPGAs during idle, low-power and re-configuration states that are required to determine the effective energy conservation through dynamic power management and duty-cycling and, hence, to determine operational lifetime, is least explored. Therefore, in this paper, we investigate the energy consumption of an SRAM-based FPGA in relation to idle, low-power, and configuration states so as to determine the feasibility with regards to using a SRAM-based FPGA for duty-cycle applications.

The remaining sections are organized as follows. In section 1, the theoretical aspects of energy optimization/conservation are discussed. In section 2, the factors that have an impact on the choice of an FPGA are discussed. In section 3, the experimental setup is explained. In section 4, the results are presented together with the supporting discussion. In section 5, the concluding remarks are given.

1. Theory

Energy consumption E_T, of an embedded system that consumes power, P watts for operational time duration of T seconds, can be calculated as given in Eq. (1),

$$E_T = P \times T \tag{1}$$

During this operational time T, an individual module of an embedded system can be in one or more of the following states and therefore, the energy consumption can be analyzed according to these states.

Active The FPGA performs desired task(s). The power consumption during this state is the sum of both the dynamic and static power consumption.

Idle There is no switching activity, and therefore the power consumption corresponds only to the static power.

Sleep A built-in state, in which power consumption of a module is reduced as compared to that of the idle state.

Power-off Power supply to a module is turned-off so as to minimize power consumption.

Active-to-sleep This is a transitional state, in which a module transitions from active to sleep state.

Sleep-to-active A transitional state during which a module is switched to active state from sleep state.

Active-to-power-off A transitional state that corresponds to switching a module from active to power-off state.

Power-off-to-active This is also a transitional state in which a module is powered-on and thus, transitions from power-off to active state.

All of the above states may not be applicable to each individual module in an embedded system. However, as all integrated modules are supposed to perform certain operations during operational time T and therefore, each module experiences at-least active and idle states. Accumulating energy consumption of all N modules of an embedded system associated with active and idle states, the Eq. (1) can be expanded to Eq. (2),

$$E_T = \sum_{i=1}^{N} \left(P_{active\_i} \times T_{active\_i} + P_{idle\_i} \times T_{idle\_i}\right) \quad (2)$$

The active energy consumption ($P_{active\_i} \times T_{active\_i}$) in Eq. (2) depends on the required performance in an application and therefore, provides limited options that can be exploited to optimize it in an application independent embedded system. One possible option is to integrate low-power modules that can be dynamically configured to different operating frequencies, so as to optimize the operating frequency and the associated power consumption for a module whenever possible. In addition, for certain high-energy consuming activities, integrating more than one module with different performance and power consumption specifications can assist in optimizing the energy consumption. For example, a processing unit may include a micro-controller and an FPGA in order to optimize the performance and power consumption according to the requirements of an underlying application.

In relation to the idle state, during which the energy consumption is the product of the application dependent idle time and the system dependent power consumption, minimizing static power consumption can lead to overall energy optimization. This can be achieved by integrating modules that support sleep states and, by designing an embedded system such that the individual modules can be switched to sleep states dynamically. With this method in practice, an individual module can either be in one of the four states, active, sleep, active-to-sleep, and sleep-to-active state. Based on these states, the resulting energy consumption can then be calculated as given in Eq. (3). In relation to an FPGA-based system, it is important to consider the energy consumption during all these states in general and during the sleep-to-active state in particular, as it

plays an significant role in deciding whether or not to switch an FPGA in the sleep state for a given time period.

$$E_T = \sum_{i=1}^{N}(P_{active_i} \times T_{active_i} + P_{sleep_i} \times T_{sleep_i} + P_{active\_to\_sleep\_i} \times T_{active\_to\_sleep_i} + P_{sleep\_to\_active\_i} \times T_{sleep\_to\_active_i}) \quad (3)$$

By switching individual modules to sleep states, energy consumption can be reduced significantly as compared to the case in which they were in idle states. However, the resulting energy consumption can still be a challenge for energy constrained applications. In order to minimize the energy consumption associated with the sleep state of a module, the power supply to the module can be turned-off. In this way, the energy consumption of a module can be reduced to zero joules. With this state in practice, which we call the power-off state in this paper, the total energy consumption E_T given in Eq. (3) can be calculated by replacing the power and timing parameters corresponding to the sleep state with those of the power-off state.

In addition to dynamically switching an individual module to sleep or power-off state when it is idle, whole embedded system can also be operated in a duty-cycle manner so as to conserve energy and to maximize the operational lifetime. In a duty-cycle approach, an embedded system periodically performs the required task(s) for a short period of time, τ, in relation to the total time period, T as shown in Figure 1. During time period (T-τ), which is called inactive duration in this paper, all integrated modules can be switched to the power-off state in order to conserve energy. This, however, requires one module to keep track of the time duration so that it has the ability to switch all modules to active states after inactive duration. For example a low-power micro-controller can be used for this purpose. In relation to an embedded system that integrates an FPGA, the effective energy conservation that can be achieved by means of dynamic power management and duty-cycling depends on the energy consumption during sleep, power-off, sleep-to-active, and power-off-to-active states and, is typically dictated by a number of factors related to an FPGA in use and are discussed in the next section.

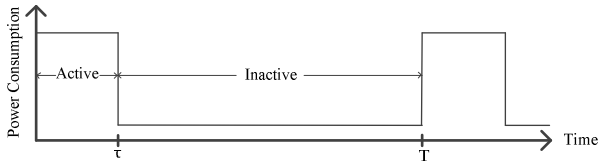

Figure 1. Duty-cycling; an embedded system performs desired operations during active duration, and then remains inactive for (T-τ) duration

2. Choosing an FPGA for duty-cycle applications

In an FPGA-based embedded system that is targeted to achieve high-performance, re-configurability, low-cost, compact size, and low-power consumption, the choice of an FPGA plays a crucial role. The factors that influence the choice include the size of an FPGA in terms of logic resources, the underlying technology, availability of low-power states and, support for easy and fast development of hardware and software.

In relation to logic resources, modern FPGAs typically consists of logic cells/elements (i.e. look-up tables, flip-flops, etc.), memory buffers, and embedded logic units such as multipliers, circuitry for synthesizing different clock frequencies, etc. As the number of logic resources in an FPGA has an impact on cost, physical size and power consumption, it is important to choose an FPGA with the right amount of resources so that, for a given set of applications, an integrated FPGA results in both an optimal logic resource utilization and power consumption. For an FPGA-based embedded system that is targeted to achieve high-throughput processing performance and low-power characteristics, example applications include vibration-based condition monitoring, structural health monitoring, image-based industrial and environmental monitoring. The actual type and amount of logic resources required to synthesize these applications will vary. However, to provide an indication to the readers, a list of major resources and their utilization (i.e. number of resources used to synthesize an application) as found in published literature [1]-[3] and [6]-[9] is complied in Table 1.

In addition to logic resources, the underlying technology of an FPGA also affects the performance, power-consumption, re-configurability, cost etc. and therefore, requires equal consideration in the selection of an FPGA. In relation to technology, modern FPGAs can be classified into three major categories, anti-fuse, flash, and SRAM-based FPGAs, and different pros and cons are associated with each category. Unlike anti-fuse FPGAs, both the flash and the SRAM-based FPGAs can be configured multiple times and thus, can be integrated in a generic embedded system in order to realize different applications. Therefore, we restrict our discussion to only these two categories. The major difference in these two categorizes is the manner in which the configuration data is stored in the device. In flash-based FPGAs, the configuration data is stored in flash memory cells and is retained even when power to the FPGA is turned-off. On the other hand, the configuration data in SRAM-based FPGAs is stored in SRAM cells and is lost when the power is turned-off. Therefore, an SRAM-based FPGA is required to be re-configured each time the power is turned-on. In relation to dynamic power management and duty-cycling in which the power supply to an FPGA is turned-off so as to conserve energy, a flash-based FPGA is likely to result in a shorter power-off-to-active time and the associated energy consumption. However, typical SRAM-based FPGAs not only offer additional embedded resources such as block RAM, and MAC units, but are built on relatively more advanced semiconductor process technology as compared to their contemporary flash-based counterparts, and thus provide a better performance [10]. Therefore, it is interesting to investigate their energy consumption during different states in relation to duty-cycle applications.

Table 1. Type of logic resources and their utilization in realizing certain monitoring applications on FPGAs

Resource Type	Four –Input Look-up Tables	Flip-Flops	Dedicated RAM (kb)	Multipliers (18x18 bits)
Number of resources used	7000 to 11000	2800 to 6900	24 to 300	4 to 35

In order to choose a reconfigurable, low-cost, and low-power FPGA able to provide ample amount of resources as described in Table 1, there are number of FPGAs from different vendors that can be considered. However, at present, both in terms of volume and revenue, the Xilinx and the Actel are the leading manufacturers of the SRAM and flash-based FPGAs, respectively and therefore, provided the motivation to consider the FPGAs from these two manufacturers. The Spartan-6 is a low-power and low-cost family of Xilinx's SRAM-based FPGAs, where as the IGLOO family can be

associated with the same attributes among the Actel's flash-based FPGAs. Among the Spartan-6 FPGAs, it is the XC6SLX16-2CPG196 [13] that has the desired amount of resources as given in Table 1 and therefore, it is used to investigate the energy consumption in relation to duty-cycling applications. Among the Actel's IGLOO family, it is the AGL1000V2FBGA144 [14] that provides a similar amount of resources as compared to that of the XC6SLX16-2CPG196 and therefore, we find it interesting to investigate its energy consumption so as to compare the SRAM-based FPGA with the flash-based FPGA.

3. Experimental setup

In order to obtain energy consumption parameters associated with SRAM-based FPGA, an FPGA-based embedded system, the SENTIOF [15] is used for experimental purposes. The SENTIOF is an FPGA-based high-performance embedded system that has been developed at Mid Sweden University, Sweden. The simplified architecture of the SENTIOF is shown in Figure 2. In addition to other modules such as micro-controller, radio transceiver, SRAM, etc. it integrates the Spartan-6 XC6SLX16-2CPG196 FPGA in order to achieve high-throughput computationally intensive processing within the embedded system. The low-power micro-controller performs control specific operations such as dynamic power management. With dynamic power management, each module including the FPGA can be switched to sleep and power-off state. The built-in sleep state of the FPGA, which is called the "suspend state" in relation to this FPGA, is activated by de-asserting the SUSPEND input signal of the FPGA as shown in Figure 3. When it is in the suspend state, the micro-controller can de-assert this signal to switch it back to the active state. The micro-controller is also able to monitor the AWAKE signal, which is asserted by the FPGA when it is ready to perform the desired operations. In order to switch the FPGA to the power-off state, the power-supplies to the FPGA that are VCCINT (1.2V), VCCAUX (3.3V) and VCCIO (3.3V) are turned-off. In the SENTIOF, it is only the FPGA's core that is powered through the 1.2V voltage regulator, therefore, in order to turn-off the 1.2V supply to the FPGA, the regulator is disabled. On the other hand, the 3.3V is supplied to other modules in addition to FPGA, therefore, P-type power MOSFET transistors, as shown in Figure 3, are used to turn-off VCCAUX and VCCIO of the FPGA.

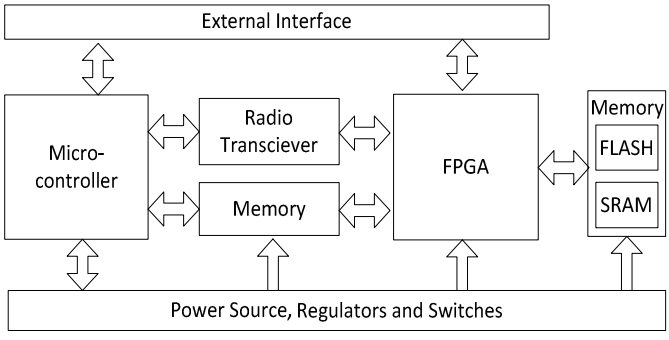

Figure 2. A simplified architecture of the SENTIOF that is used in experiments

The typical current drawn by the P-type transistor is 1uA, and can typically be turned-on and off in 1 µs and 3 µs, respectively. It should be noted that Figure 2 is included to show the reader a simplified implementation view related to the suspend and power-off state and therefore, does not include pull-up/pull-down circuits etc. that are required to provide a stable operation. In order to switch the FPGA back to the active state from power-off state, the micro-controller enables the 1.2V voltage regulator, turns-on the MOSFET and then asserts the PROGRAM signal such that the FPGA can load the configuration data from the associated flash memory. When the configuration is completed and the FPGA is ready to operate, it asserts the DONE signal.

The timing and power consumption parameters that are presented in this paper, in relation to the SRAM-based FPGA, are measured using the SENTIOF. The power consumption is measured by recording the current drawn through a 3.6 V main power source for the SENTIOF. The parameters associated with the flash-based FPGA, AGL1000V2 are extracted from the manufacturer's specifications and the software tools (LiberoIDE) provided by the manufacturer. The power consumption for the AGL1000V2 corresponds to the power supply voltage of 1.2 V and 3.3 V for the core and IOs, respectively.

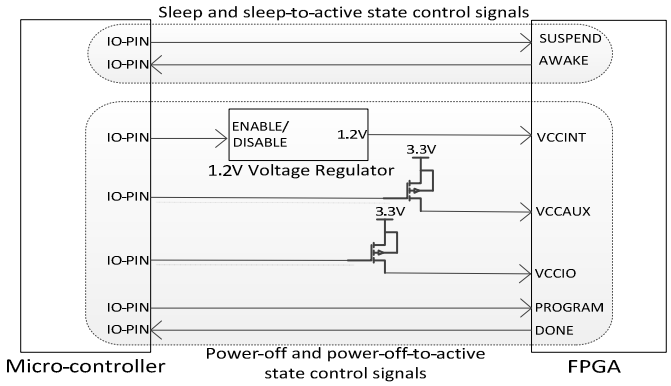

Figure 3. A simplified depiction of implementation details related to the suspend and power-off state

4. Results

The typical static power consumption of the SRAM-based FPGA, XC6SLX16 is 24 mW. In order to conserve static power during idle state, it can be switched to the suspend state. During the suspend state, the FPGA maintains all the design states and the configurations data while reducing the static power consumption to a lower level. The average power consumption measured during the suspend state is about 11 mW, which corresponds to more than a 50% reduction as compared to that of the idle state. The important feature of this state is that the FPGA can be switched to the suspend state in less than 14 ns, and back to active state in less than 20 µs. With this short transition time to and from the suspend state, it is possible to conserve energy by frequently switching the FPGA to the suspend state.

Based on this fast transition time and a more than 50% reduction in power consumption, the suspend state can be an ideal option to conserve energy when the

FPGA is idle for short durations. However, for longer durations that are typically associated with duty-cycle applications, the power consumption of 11 mW can lead to the wasting of a valuable amount of energy. For example, if the FPGA is switched to the suspend state for 5 minutes, the resulting energy consumption is 3.6 joules, which can otherwise be used to perform certain other functions. Therefore, when the FPGA is idle for long durations we opt to switch it to the power-off state so as to minimize the energy consumption. However, by switching the FPGA to power-off state, all the design states and the configuration data are lost. As a consequence, on the next power-up, the FPGA requires to be re-configured by loading the configuration data (bit stream) from the associated non-volatile memory.

The SRAM-based FPGA, XC6SLX16 can be configured through a serial peripheral interface (SPI) interface with a selectable data bus width of single, dual and quad-bits and a maximum transfer rate of 66 MHz. In order to achieve a fast configuration time for the FPGA, a non-volatile memory that provides quad SPI interface and an operating frequency of upto 85 MHz is used. In an uncompressed format, a configuration time of 15.16 ms is recorded for loading the bit stream of 3,731,264 bits from the associated non-volatile memory to the FPGA. During this configuration process, the SPI bus width and configuration speed were set to quad-mode and 66 MHz, respectively. In order to further minimize the configuration time, the FPGA can also be configured using a compressed bit-stream. However, the size of such a bit-stream depends on the synthesized application and therefore, results in an application dependent configuration time. It should be noted that during the power-off-to-active state, an additional delay of about 8 ms, apart from the actual configuration time, was measured. This delay corresponds to the time at which the power to the FPGA is turned-on to the instant at which it becomes ready to load the bit stream and, thus, leads to the power-off-to-active time to 23.5 ms.

During the power-off-to-active state, the instantaneous current drawn from a 3.6V power source is shown in Figure 4. It should be noted that during the configuration process, both the FPGA and the FLASH memory are ON and therefore, the current drawn corresponds to both of these modules. Upon power-up, the FPGA causes the instantaneous current to exceed a little over 1A. The average current consumption

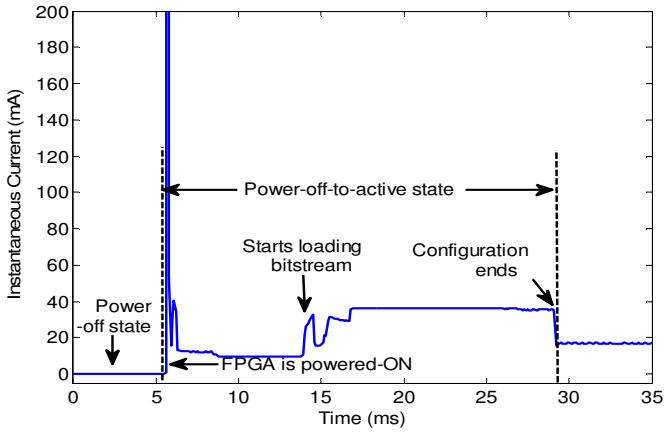

Figure 4. Instantaneous current consumption during power-off-to-active state of the FPGA

during power-off-to-active state is measured to be 30.3 mA, which leads to the energy consumption of 2.56 mJ in relation to the power-off-to-active of the FPGA. This, in other words, means that the FPGA should only be switched to power-off state if the energy conserved is more than 2.56 mJ. The resulting duration for which the energy conservation is more than 2.56 mJ is 82.5 ms or more. The results relating to different states of the FPGA are also summarized in Table 2.

Table 2. Average power consumption, switching time, and energy consumption during different states of the SRAM-based FPGA

	Idle state	Suspended state	Power-off state	Power-off-to-active state
Average power consumption	24 mW	11 mW	0	109 mW
Time required to switch the FPGA to	-	14 ns	< 10 ns	23.5 ms
Energy consumption	Time dependent	Time dependent	0	2.56 mJ

In order to compare the effective energy conservation that can be achieved by switching the FPGA to the suspend and power-off state in relation to that of idle state, the percentage energy conservation for a wide range of inactive duration is shown in Figure 5. For a given inactive duration, the energy conservation corresponds to the ratio of the energy consumption during the idle state and the suspend state or the power-off state. The energy consumption corresponding to the suspend and the power-off state also includes the energy consumed to switch the FPGA to the respective state and back to the active state. From Figure 5, we can observe that for inactive duration of less than 38 µs, the FPGA should not be switched to the suspend or power-off state. However, for inactive duration of 38 µs to 235 ms, switching the FPGA to the suspend state results in more energy conservation as compared to the power-off state. This is mainly due to the short transition time to/from the suspend state. On the other hand, for inactive duration longer than 235 ms, it is the power-off state that results in the maximum energy conservation. The maximum energy conservation during the suspend and power-off state is around 54.1% and 99.9%, respectively. The 99% energy conservation can be achieved for an inactive duration of 10 seconds, which suggests that the SRAM-based FPGA can be a suitable choice for a wide range of duty-cycle applications.

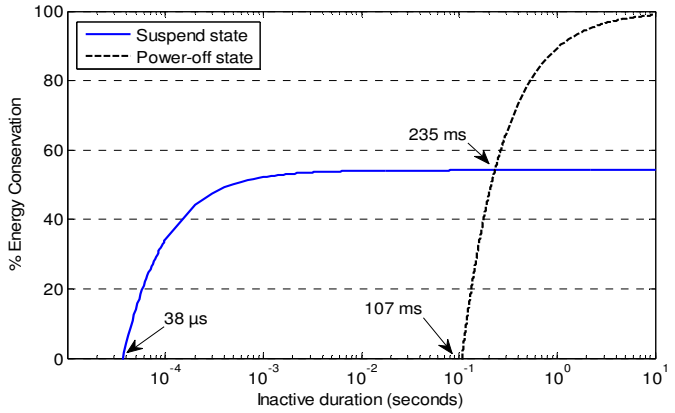

Figure 5. Percentage energy conservation during the suspend and power-off state in relation to idle mode

Flash-based FPGAs generally consumes less static power, and can also be switched to low-power states in less time as compared to SRAM-based FPGAs. Therefore, it is interesting to compare the SRAM-based FPGA with an equivalent flash-based FPGA. For this purpose, we estimated the energy consumption of the flash-based FPGA, AGL1000V2 for different inactive durations. The typical static power consumption of the flash-based FPGA is 75 µW [14]. This FPGA can also be switched to a low-power state, known as the FlashFreeze state. The typical power consumption during this state is 50 µW and, the time required to enter and exit from the FlashFreeze state is about 1 µs. The energy consumption of both the flash and SRAM-based FPGAs for different inactive durations is shown in Figure 6. For a given inactive duration, the energy consumption of the flash-based FPGA corresponds to the energy consumed during the FlashFreeze state and during switching the FPGA to FlashFreeze state and back to active state. The energy consumption of the SRAM-based FPGA corresponds to that of the suspend state for in-active duration of up to 235 ms, and to the power-off state for durations longer than 235 ms. From Figure 6, we can observe that for inactive duration of 100 µs to 10 ms, the difference in energy consumption between the two FPGAs is negligible. After 10 ms, the energy consumption associated with SRAM-based FPGA increases and leads to a maximum difference of 2.4 mJ at inactive duration of 234 ms. However, with inactive duration of 235 ms or more, the energy consumption of the SRAM-based FPGA remains almost constant where as the energy consumption associated with the flash-based FPGA tends to increase. For an inactive duration of more than 54 seconds, the energy consumption of the flash-based FPGA exceeds that of the SRAM-based FPGA.

In Figure 6, the flash-based FPGA was switched to the FlashFreeze state during which it consumed 50 µW and therefore, it can be argued that the power supply to the flash-based FPGA can also be turned-off so as to minimize the energy consumption associated with the flash-based FPGA. However, it should be noted that the time required for the power-off-to-active state and the associated the energy consumption is more than that of switching the flash-based FPGA from FlashFreeze to the active state. This means that by switching the flash-based FPGA to the power-off state, the maximum energy conserved in relation to the SRAM-based FPGA is likely to be less

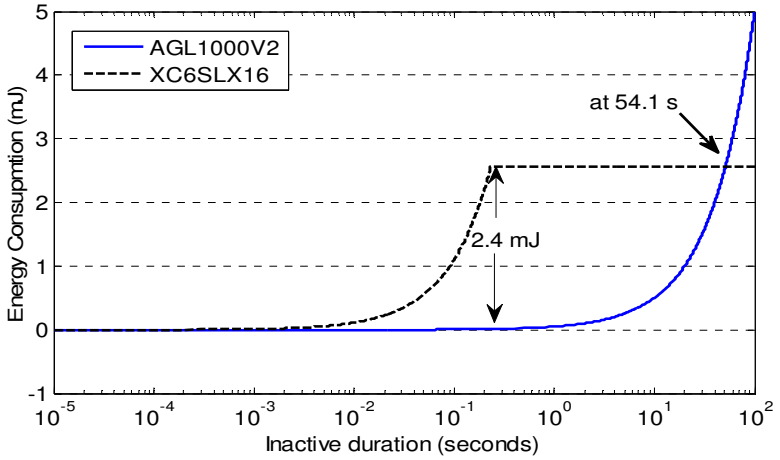

Figure 6. Energy consumption of XC6SLX16 and AGL1000V2 during low-power states for a range of inactive durations during which these FPGAs can be switched to their respective low-power states

than 2.4 mJ, which is otherwise achieved by switching the flash-based FPGA to the FlashFreeze state as shown in Figure 6.

In summary, with dynamic power management, the energy consumption associated with the idle state of the SRAM-based FPGA can be reduced significantly, and thus the FPGA can be used for duty-cycle applications with both very short as well as long inactive durations.

5. Conclusion

In this paper, we have investigated the energy consumption of an SRAM-based FPGA, XC6SLX16 in relation to the idle, suspend, power-off, and transitional states that include switching the FPGA from suspend to active, and power-off to active states, so as to determine its feasibility for duty-cycle applications. In addition, these results are compared with a flash-based FPGA that provides an equivalent amount of logic resources.

The SRAM-based FPGA typically consumes 24 mW during the idle state, and can be switched to the suspend state, in which it consumes 11 mW. The typical switching time to and from the suspend state are 14 ns and 20 µs, respectively. Therefore, for short inactive durations such as 38 µs to 235 ms, the power consumption can be reduced to more than 50%. For inactive durations exceeding 235 ms, the FPGA should be switched to the power-off state. In order to minimize both re-configuration time and the associated energy consumption after the power-on, the SPI interface associated with loading the bit stream can be operated at a maximum clock frequency of 66 MHz and a data bus width of quad-bits. The resulting energy consumption of the power-off-to-active state enables to switch the FPGA to the power-off state and thus conserves energy up to 99.9 % in relation to that of idle state.

In relation to an equivalent flash-based FPGA, AGL1000V2 that consumes 50 µW during its low-power state, we observe that by switching the SRAM-based FPGA to the suspend and power-off states, its energy consumption is almost equivalent to that for an inactive duration less than 100 ms. On the other hand, for an inactive duration greater than 54 seconds, the energy consumption associated with SRAM-based FPGA is less than that of the FLASH-based FPGA. Therefore, it can be concluded that the SRAM-based FPGA that provides more logic resources and a higher operating frequency as compared to the FLASH-based FPGA, can be used to achieve high-throughput computationally intensive processing locally in an embedded system while maximizing the operational lifetime through dynamic power management and duty-cycling.

References

[1] K. Shahzad, P. Cheng, and B. Oelmann, "Architecture exploration for a high-performance and low-power wireless vibration analyzer," *IEEE Sensors Journal*, vol.13, no.2, pp.670-682, Feb. 2013.

[2] K. Khursheed, M. Imran, A.W. Malik, M. O'Nils, N. Lawal, and B. Thörnberg, "Exploration of Tasks Partitioning between Hardware Software and Locality for a Wireless Camera Based Vision Sensor Node", *26th International Symposium on Parallel Computing in Electrical Engineering (PARELEC),* pp.127-132, 3-7 April 2011.

[3] C. H. Zhiyong, L. Y. Pan, Z. Zhenxing, and M. Q. H. Meng, "A novel FPGA-based wireless vision sensor node," *IEEE International Conference on Automation and Logistics, 2009,* pp.841-846, 5-7 Aug. 2009.

[4] Y. Sun, L. Li, and H. Luo , "Design of FPGA-Based Multimedia Node for WSN", *7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM),* pp.1-5, 23-25 Sept. 2011.

[5] J. Portilla, T. Riesgo, and A. de Castro, "A Reconfigurable FPGA-Based Architecture for Modular Nodes in Wireless Sensor Networks", *3rd Southern Conference on Programmable Logic*, pp.203-206, 28-26 Feb. 2007.

[6] J. Henaut, D. Dragomirescu, and R. Plana, "FPGA Based High Date Rate Radio Interfaces for Aerospace Wireless Sensor Systems," *4rth International Conference on Systems, ICONS '09*, pp.173-178, 1-6 March 2009.

[7] A.A.M Bsoul, R. Hoskinson, M. Ivanov, S. Mirabbasi, H. Abdollahi, "Implementation of an FPGA-based low-power video processing module for a head-mounted display system", *IEEE International Conference on Consumer Electronics (ICCE),* pp. 214,217, 11-14 Jan. 2013.

[8] E. C. Yepez, R. A Osornio-Rios, R. J. Romero-Troncoso, J. R. Razo-Hernandez, and R. Lopez-Garcia, "FPGA-Based Online Induction Motor Multiple-Fault Detection with Fused FFT and Wavelet Analysis," *International Conference on Reconfigurable Computing and FPGAs,* pp. 101,106, 9-11 Dec. 2009.

[9] M.L. Kaddachi, L. Makkaoui, A. Soudani, V. Lecuire, and J. Moureaux, "FPGA-based image compression for low-power Wireless Camera Sensor Networks," *3rd International Conference on Next Generation Networks and Services (NGNS),* pp. 68-71, 18-20 Dec. 2011.

[10] D. G. Bailey, "Design for embedded image processing on FPGAs", *2011 John Wiley & Sons (Asia) Pte Ltd*, ISBN 978-0-470-82849-6, pp. 54-72, 2011.

[11] R. Maheswar, P. Jayarajan, and F. N. Sheriff. "A Survey on Duty Cycling Schemes for Wireless Sensor Networks." *International Journal of Computer Networks and Wireless Communications (IJCNWC),* vol. 3, no. 1,pp. 37-40, February 2013.

[12] H. Yoo, M. Shim, and D. Kim, "Dynamic Duty-Cycle Scheduling Schemes for Energy-Harvesting Wireless Sensor Networks," *IEEE Communications Letters*, vol.16, no.2, pp.202-204, February 2012.

[13] Spartan-6 XCLS6X16-2CPG196 San Jose, CA: Xilinx Inc. 2013. [Online]. Available: www.xilinx.com.

[14] IGLOO AGL1000V2FPGA144 Aliso Viejo, CA: Microsemi Corporation. 2013. [Online]. Available: www.actel.com.

[15] K. Shahzad, P. Cheng, and B. Oelmann, "SENTIOF: An FPGA-Based High-Performance and Low-Power Wireless Embedded Platform" Accepted for publication *in International Conference on Wireless Sensor Networks (WSN'13)*, 8-11 Sep. 2013.