Energy-Aware Scheduling of FIR Filter Structures using a Timed Automata Model

Wognsen, Erik Ramsgaard; Hansen, Rene Rydhof; Larsen, Kim Guldstrand; Koch, Peter

Published in:
2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)

DOI (link to publication from Publisher):
10.1109/DDECS.2016.7482468

Publication date:
2016

Energy-Aware Scheduling of FIR Filter Structures using a Timed Automata Model

Erik Ramsgaard Wognsen, René Rydhof Hansen, Kim Guldstrand Larsen, and Peter Koch
Aalborg University, Denmark
{erw,rrh,kgl}@cs.aau.dk   pk@es.aau.dk

***Abstract*—Software Defined Radio (SDR) devices are becoming increasingly popular due to their support for mode-, standard- and application-flexibility. At the same time however, the energy consumption of such devices typically suffers from the use of reconfigurable real-time platforms which are known to be severely power hungry. In this work we therefore show how to use tools and techniques developed by the formal methods community to minimize the energy consumption of Finite Impulse Response (FIR) filters which are extensively used in SDR front-ends. We conduct experiments with four different FIR filter structures where we initially derive data flow graphs and precedence graphs using the Synchronous Data Flow (SDF) notation. Based on actual measurements on the Altera Cyclone IV FPGA, we derive power and timing estimates for addition and multiplication, including idling power consumption. We next model the FIR structures in UPPAAL CORA and employ model checking to find energy-optimal solutions in linearly priced timed automata. In conclusion we state that there are significant energy-versus-time differences between the four structures when we experiment with varying numbers of adders and multipliers. Similarly, we find that idle power becomes an important parameter when a high number of functional units are allocated.**

I. Introduction

With the easy availability of cheap SDR devices, e.g., the RTL-SDR[1], it seems certain that the near future will see a proliferation of systems incorporating SDR at all levels. Especially, this is emphasised by the potential of the rapidly expanding "Internet of Things" technology where embedded systems and appliances, often low-power and low-cost, are connected through the Internet. Further afield, SDR applications have also found their way into energy critical autonomous systems such as (nano-)satellites, e.g., the Aalborg University student satellite AAUSAT3[2]. Common to most of such SDR applications is that they are built on low-energy platforms with a limited (re-)supply of energy, e.g., in the form of batteries or solar panels. Consequently, when designing and implementing SDR applications, it is important to take energy-consumption into account and to reduce the application's energy consumption where ever possible.

We employ a formal method, which originally is developed to find optimal solutions in large search spaces, to model, explore, and optimize the energy consumption of SDR front-ends. Specifically, we investigate four implementation structures for digital FIR filters which are often used in SDR front-ends, [1]. Over the years much work has been done in the context of energy reduction in real-time digital filter systems, e.g., [2] and [3]. These contributions however, rely on methods related to either 1) the circuit architecture, 2) the type of arithmetic employed, or 3) the specific program structure used for implementing the filter algorithm. Therefore, to our knowledge no previous paper has addressed the use of linearly priced timed automata to conduct energy-aware scheduling of real-time digital signal processing algorithms onto reconfigurable hardware (HW).

We discuss how the SDF notation, [4], can be used to derive precedence graphs (PG) which allows us to formally model these structures as *linearly priced timed automata* using the UPPAAL CORA tool suite [5]. The timed automata model can next be model checked and energy-optimal solutions can be found based on power and time estimates (for addition and multiplication) that we derive from actual measurement on the Altera Cyclone IV FPGA. Based on the model checking results we conclude that significant differences in energy and execution time can be observed for the different FIR structures. The paper, which is based on [6] and [7], is organized with an introduction to the FIR structures in Section II, a discussion on the experimental set-up in Section III, an analysis of the results in Section IV, and finally the conclusion in Section V.

II. FIR Filter Structures

In the time domain an FIR filter is described by its difference equation $y[n] = \sum_{l=0}^{N} b_l \cdot x[n-l]$ where N is the filter order and $x[n-l]$ represents a sequence of input tokens. The constants b_l are known as the filter coefficients (equivalent to the impulse response) which completely specify the behavior in terms of amplitude- and phase responses of the filter, see e.g., [8]. Various structures expressed as SDF graphs can implement the difference equation. First and foremost, the direct implementation which is known as the transversal filter, Figure 1 for $N = 5$. Figure 1 (a) illustrates a typical data flow graph for the filter which in (b) is redrawn using the SDF notation. Here, the arithmetic operations are identical to those shown in (a), but the delay elements (z^{-1} nodes) are substituted with direct arcs denoted D (no arithmetic operation). Further, the distribution of a token corresponds to an operation which is indicated as a fork (F) node.

Input to and output from every node is marked with a number specifying the amount of tokens being consumed and produced, respectively, each time the node is executed. This is also shown in Figure 1 (b) along with the fact that each node (red) and each arc (green) must be given a unique number (in any random order). Note that the input arc from the ADC and the output arc to the DAC are both omitted under

This work was funded by the EU FP7 project SENSATION, grant number 318490.

[1]http://www.rtl-sdr.com

[2]http://www.space.aau.dk/aausat3/

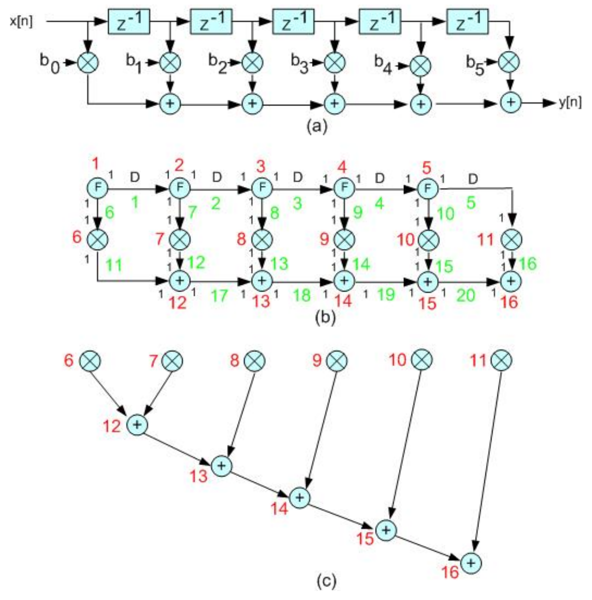

Fig. 1. The direct implementation of Equation **??** is shown here as a data flow graph (a), a synchronous data flow graph (b), and as a precedence graph (c).

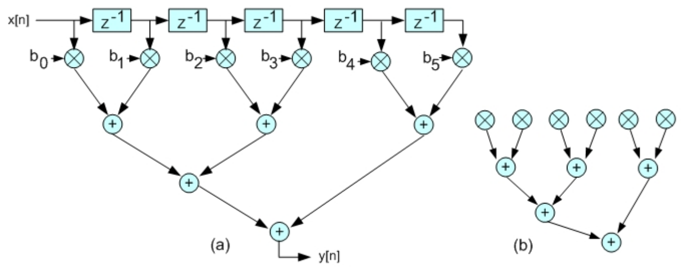

Fig. 2. The accumulation of the product terms is conducted in terms of an adder tree. The data flow graph and PG are shown in (a) and (b), respectively.

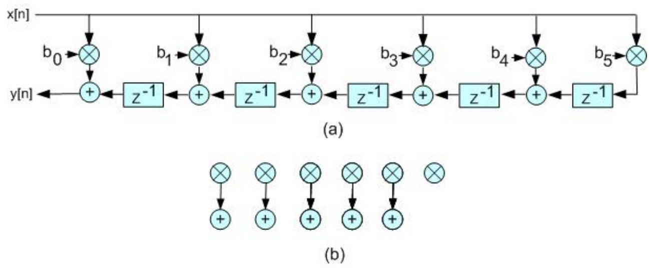

Fig. 3. Using retiming, all delay elements can be shifted into the adder chain. The data flow graph and PG are shown in (a) and (b), respectively.

the realistic assumption that data from the external world is always available and data to the external world can always be delivered. Similarly for the coefficients b_l.

It is possible to 1) decide if the FIR filter is executable, i.e., the amount of tokens on the arcs remains bounded and non-negative, and 2) to identify a possible execution order among the individual nodes. The latter is needed in order to transform the SDF graph into a PG, which then holds information on 1) the inter-precedence relations among the nodes, and 2) the amount of inherent parallelism, see Figure 1 (c).

For non-recursive algorithms such as FIR filters, the precedence relations are given directly from the SDF graph which is also obvious from Figure 1 (the fork nodes do not need to be considered). Note however, that the PG, e.g., Figure 1 (c) explicitly expresses the precedence relations without timing information which can next be derived by scheduling the graph according to a certain HW allocation.

A. Alternative FIR Structures

Theoretically, there exists infinitely many structures for a given difference equation. Designers normally use the transversal filter which fits conveniently into the architecture and instruction set of most digital signal processors (DSPs). In the context of reconfigurable HW architectures, the computing paradigm is different since it supports exploitation of parallelism, thus making it essential to discuss other structures with a potentially higher degree of parallelism.

1) Transversal Filter with Adder Tree: The transversal filter is characterized by a long adder chain which prohibits a noticeable speed-up, even in the context of an increased number of HW units. One option is to replace the adder chain by an adder tree, Figure 2 ($N = 5$). In this figure (and onwards) we show only the original data flow graph together with the PG. Comparing Figure 2 (b) to Figure 1 (c) we clearly see that the critical path has been reduced by introducing more parallelism, thus enhancing the solution space in terms of scheduling and assignment possibilities for a given HW allocation. The overall input-output relation is maintained for floating point computation but for fixed point scenarios there might be numerical deviations leading to different signal-to-noise ratios (SNR).

2) Transposed Form: The two structures are characterized by a delay-line at the input, i.e., the input sample $x[n]$ is delayed before it is fed to the multipliers, and thus all multiplications are computationally independent and can execute in parallel. At the same time however, there is more or less dependency among the additions as already illustrated, which leads to an adder dominated critical path. Using the theories for retiming, [9], it is possible to reorganizing the delay elements such that they are all shifted to the output, also know as the Transposed FIR filter, Figure 3.

The salient feature of the transposed form is that the delay elements now serve to decouple the adder operations and since the multipliers all receive the same input data ($x[n]$), the multiplications can still execute independently. The transposed form therefore is characterized by a series of data independent multiply-accumulate operations as shown in Figure 3 (b), enabling a larger solution space for allocation, scheduling and assignment combinations.

3) Symmetric Impulse Response: FIR filters can be designed to have exact linear phase response which in many cases (SDR front-ends included) is a most wanted property. The designer must determine if the filter should have this characteristic, and then include the linear phase requirement into the specification. The resulting filter has symmetry in the filter coefficients, i.e., $b_{N-i} = b_i$ for $i = 0..(N-1)/2$, N odd. The symmetry can be utilized efficiently in order to reduce the number of multiplications by a factor of two as shown in Figure 4. Concerning the numerical properties in a fixed point

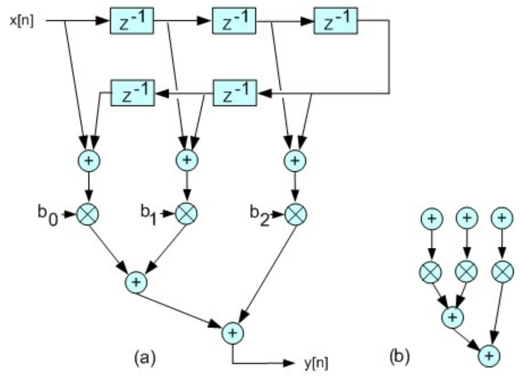

Fig. 4. A linear phase response can be utilized to reduce the number of arithmetic operations. The data flow graph and PG are shown in (a) and (b).

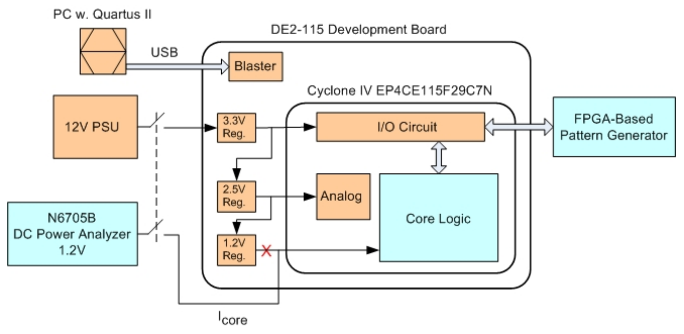
PC w. Quartus II
USB
DE2-115 Development Board
Blaster
Cyclone IV EP4CE115F29C7N
12V PSU
3.3V Reg.
I/O Circuit
FPGA-Based Pattern Generator
2.5V Reg.
Analog
Core Logic
N6705B DC Power Analyzer 1.2V
1.2V Reg.
I_core

Fig. 5. Measuring computational-dependent power overheads on the Cyclone IV FPGA is possible due to a modified Altera DE2-115 development board where the current flowing into the FPGA core logic is supplied directly from an Agilent N6705B DC Power Analyzer.

TABLE I. AVERAGE POWER CONSUMPTION AND EXECUTION TIMES FOR THE ALTERA CYCLONE IV MULTIPLIER AND ADDER CIRCUITS.

Functional Unit	Active Power, Measurement [6]	Propagation Time, Simulation [6]	Idle Power, Estimation
Embedded Multiplier	338 μW	10.32 ns	169 μW
Embedded Adder	324 μW	6.75 ns	162 μW

implementation, the symmetric structure is likely to exhibit a different SNR as compared to the transversal filter since addition is conducted prior to multiplication.

III. THE EXPERIMENTAL SET-UP

With the purpose of studying energy conditions for FIR filters implemented on an FPGA platform, we next investigate fundamental power- and timing metrics. For an FPGA, the total power consumption originates from many different sources, e.g., I/O, clock distribution, analog functions (e.g., PLLs), and from the logic conducting the computation. Since we are interested only in the actual computation, we first study the core logic power consumption. Our set-up consists of an Altera/terasIC DE2-115 development platform equipped with the Cyclone IV EP4CE115F29C7N FPGA, [10]. This platform is supplied from an external 12V DC source which feeds several on-board voltage regulators. Doing power measurement directly from this source would be meaningless. The arithmetic computations are conducted in the logic elements which belong to the core of the total FPGA fabric. The core logic is supplied from a 1.2V DC source which on the board is derived directly from a 1.2V voltage regulator. Since power is also participated in this regulator we cannot just measure the current flowing into this regulator. One may consider to conduct a current measure on the output of the regulator using a highly accurate stand-alone Amp-meter.

We have however, opted for an alternative strategy where we modify the board circuitry in order to get direct access to the core logic supply pins on the FPGA. Figure 5 shows a block diagram of the power supply scheme used in our set-up. The core logic is supplied 1.2V DC from an Agilent Technology N6705B DC Power Analyzer [11], which at the same time is able to conduct current measurements with a 0.025% + 8nA accuracy. We have chosen a second FPGA to provide the input stimuli to the device under test.

With this set-up we can measure the energy consumed by the individual arithmetic units in the filter. However, a variety of parameters are subject for variation, e.g., different circuit topologies, the input data values and rate (input stimuli pattern and frequency), and the number representation used. We expect that different adders and multipliers will exhibit varying energy consumption and thus we first investigate power- and time consumption of different adder and multiplier topologies. All circuits are coded in VHDL, 12 bit 2's complement. The adder types explored are 1) ripple-carry adder, 2) carry look-ahead adder, 3) carry-select adder, and 4) the FPGA pre-defined embedded adder. Similarly, the multiplier types investigated are 1) combinational multiplier, 2) shift-and-add multiplier, and 3) the FPGA pre-defined embedded multiplier. Our experimental campaign is discussed in [6] where we conclude that both the embedded adder and the embedded multiplier cannot be outperformed in terms of energy consumption, thus being those used here. The average (active) power and the propagation time for these two functional units are shown in Table I. These figures represent the power contribution which originates from the dynamic behavior of the total CMOS power consumption $P_{total} = P_{dyn} + P_{stat} = (P_{switch} + P_{short}) + P_{leak}$,

Since it is impossible to conduct an actual measurement of the leakage current associated with the individual adder and multiplier, the challenge therefore is to provide a realistic ratio between the dynamic and the static power consumption. According to trend reports published by International Technology Roadmap for Semiconductors (ITRS), [12], we estimate realistically that the "Logic Static Power" is equivalent to (or even higher than) the "Logic Dynamic Power". We therefore model $P_{dyn} \sim P_{stat}$ which means that the active power figures shown in Table I represent a twice as high power consumption as compared to the circuits' idle power, also shown in Table I.

Using these figures we next conduct Time-and-Energy driven Design Space Exploration (DSE) for the FIR structures. We explore how different HW allocations will impact the execution time and the energy consumption within one sample period. Basically, this is done by scheduling the structures under given HW allocations. We employ model checking in terms of UPPAAL CORA [13], [14], designed to perform cost-optimal reachability in linearly priced timed automata. A timed automata describes a state space, and model checking is the discipline of exhaustively searching such a space to determine with certainty if a given final state is reachable from the initial

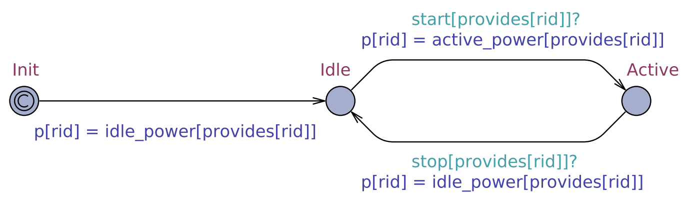
Init
Idle
Active
start[provides[rid]]?
p[rid] = active_power[provides[rid]]
p[rid] = idle_power[provides[rid]]
stop[provides[rid]]?
p[rid] = idle_power[provides[rid]]

Fig. 6. UPPAAL timed automaton modeling a physical functional unit.

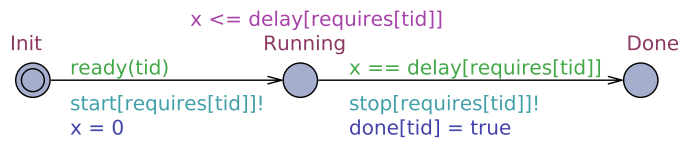

Fig. 7. UPPAAL timed automaton modeling an arithmetic operation.

state. By extending timed automata with prices (cost rates), the problem can be extended to cost-optimal reachability.

Our model consists of two different types of timed automata – one denoted "resource" representing the functional units, and another denoted "task" which models the arithmetic operations. The resource- and task-automata are shown in Figure 6 and Figure 7, respectively. The actual price marks applied in the resource automata are the $active\_power$ and $idle\_power$ as listed in Table I. Similarly, the timing for the task automata is the propagation time ($delays$) which are also indicated in Table I. All individual resources and tasks are identified by a unique id-number (rid and tid, respectively). The total system is composed of a network topology of all automata, thus representing the FIR algorithm as well as the given HW allocation.

The cost associated with executing a given filter structure is defined as 1) the sum of the power, and 2) the sum of the time being consumed by the functional units until all tasks are complete. Therefore, in order to conduct the model checking and find a cost-optimal schedule, we ask the question; "does a state exists for the system where all tasks are completed, and if yes, then calculate the energy optimal trace which brings the system into that particular state". Thus, we ask UPPAAL CORA to find the "Best Trace" for the query;

$$E <> \; forall \; (tid : tid_t) \; Task(tid).Done \qquad (1)$$

IV. Results

Since an FIR filter executes in an infinite loop, it makes sense to evaluate just one such iteration. We are interested in the shortest possible iteration period, and at the same time aiming for the smallest possible energy consumption. Since the power metrics are fixed (inclusive idling), the only tunable parameter is the HW allocation. Our experiments ignore architectural components such as memories, registers, busses, and control entities. One may argue that by including only arithmetic units our model will be too simple to capture the overall working of the register transfer level (RTL) architecture. We do hide some power consumption information originating from signal communication and datapath control, but in order to verify our idea, we have decided, as a first approximation, to rely only on the adder/multiplier contributions.

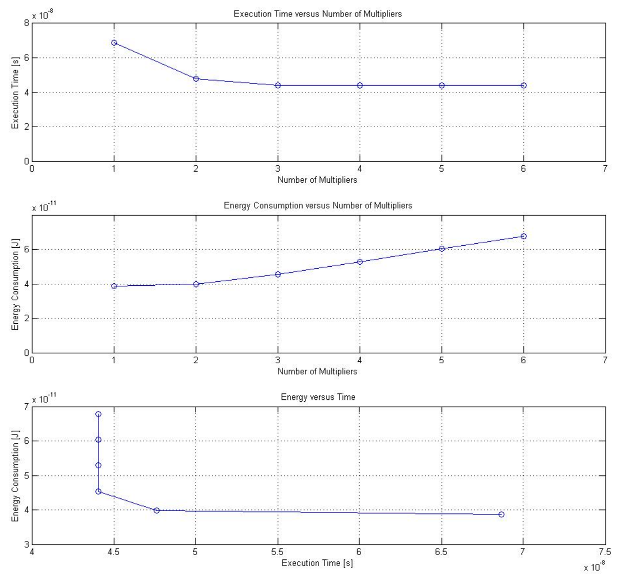

Fig. 8. Results of the energy optimal scheduling of the transversal FIR filter. Here, as well as in the Figures 9, 10, and 11, the top figure shows the total execution time as a function of multipliers allocated, the middle figure shows the energy consumption and the bottom figure presents the Pareto curve for various combinations of multipliers/adders allocation.

A. The transversal FIR filter

Due to its sequential adder chain, the transversal filter is not able to benefit from a HW allocation beyond one adder. The high degree of parallelism associated with the multiplications however, makes it interesting to experiment with a variable number of multipliers ranging from 1 to 6.

Figure 8 illustrates the time- and energy-consumption as well as the Pareto curve for 6 different HW combinations. The execution time decreases for an increased number of multipliers but only up to 3 multipliers after which no further speed-up is possible, the reason being that the adder chain becomes the bottleneck. Concerning the energy consumption we see a monotonic increasing curve which becomes essentially linear from 3 multipliers and onwards. Although we do not see any improvement in the execution time beyond 3 multipliers, the energy continues to increase which is due to the idle cost associated with the inactive multipliers. The Pareto curve discloses that 2-3 multipliers provide the best time-energy trade-off.

B. Transversal filter with adder tree

The transversal filter with an adder tree enhances the overall amount of parallelism. The parallelism associated with the multiplications remains unchanged but now potentially 3 additions can be conducted simultaneously. However, allocating 3 adders is inferior to the solution with only 2 adders because the critical path is still associated with the adder tree. Therefore, we have chosen to experiment with 1-2 adders and 1-6 multipliers, see Figure 9.

We see a monotonic decrease in the execution time (for 1 and 2 adders) when the number of multipliers is increased.

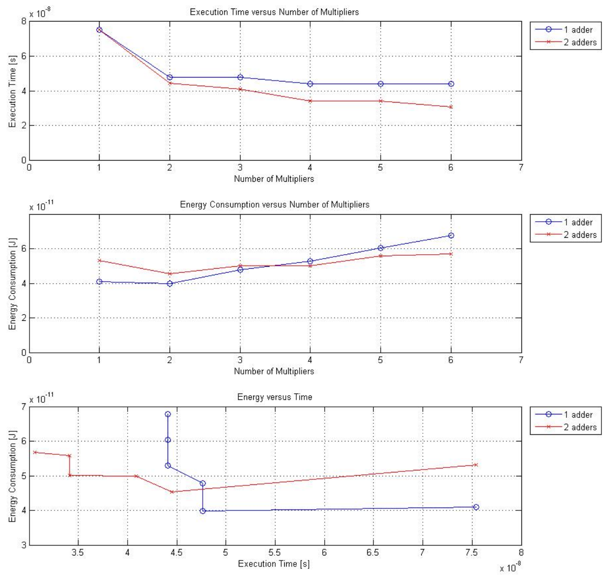

Fig. 9. Scheduling results for the transposed filter with adder tree structure.

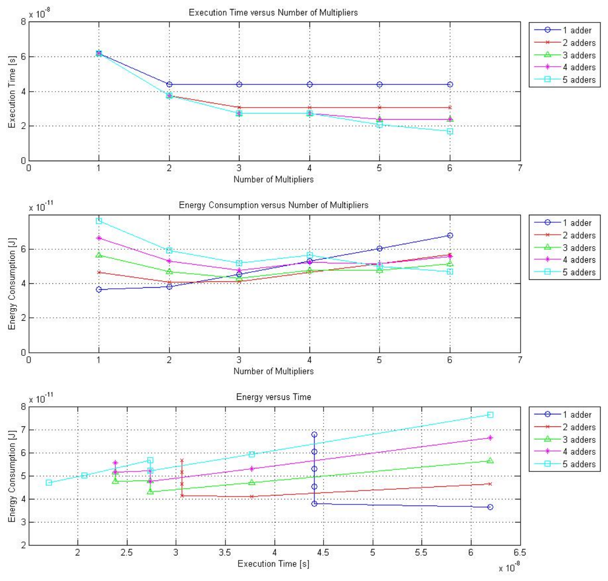

Fig. 10. Scheduling results for the transposed filter structure.

Interestingly, an even larger execution time deviation between the 1-adder and the 2-adder solution is observed as more multipliers are allocated. Explaining this behavior, we compare the observation against the energy consumption results. Here we also see monotonic increasing curves, the exception being the allocation with 1 multiplier only. In this particular case we will have 1-2 adders idling while waiting for yet another product to be computed, and due to the idle power consumption this ”waiting time” actually turns out to be ”energy-wise” more costly as compared to other scenarios with more multipliers. As the number of multipliers continues to grow, the 2-adder solution gradually becomes the better one, which can be explained by the fact that more products are available for addition, and thus more flexibility exists in terms of pairing two products for addition. The Pareto curve discloses that the best time-energy trade-off is a combination of the 1-adder and the 2-adders solution thus making the decision a matter of time or energy being the most critical metric.

C. The transposed FIR filter

The transposed FIR filter has a significant amount of inherent parallelism, and therefore it also makes sense to conduct experiments with a high number of both multipliers and adders, i.e., 1-6 multipliers and 1-5 adders. The results are shown in Figure 10 where the overall conclusion is that more allocated adders decreases the overall execution time when increasing the number of multipliers. Further, we discover the notable shift from the highest to the lowest energy consumption (for a high number of adders and differently for a low number of adders) when the number of multipliers is increased.

We can explain this behavior by a high degree of idle time among the adders when only a few multipliers are allocated. Combined with the fact that more allocated multipliers and adders enhances the scheduling/binding solution space leads to the evident observation of decreased execution time as well as energy consumption for an increase in the HW resources. This behavior is also reflected in the Pareto curves which illustrates that a shift towards lower execution time for high allocation solutions not necessarily increases the energy cost.

D. The symmetric impulse response FIR filter

Our final experiment involves the symmetric structure. This structure has 1) the number of multiplications decreased by a factor of two, and 2) operations shuffled such that (some) additions now are executed prior to the multiplications. We conduct experiments with up to three multipliers and two adders, the reason being the fact that executing three multiplications in parallel will lead to a substantial overhead in terms of idle power, thus making it more beneficial to execute only the two multiplications (in the critical path) in parallel and thus delaying the multiplication as well as the addition which are outside of the critical path. The result are shown in Figure 11.

The experimental model checking confirms that more allocated HW leads to a reduction in the overall execution time, and a derived consequence is a relative reduction in the energy consumption as well, due to decreased idle power consumption. The absolute energy consumption however, increases with more allocated HW. The Pareto curve indicates that the choice of optimal adder/multiplier allocation must depend on the requirements given to the time- and energy-consumption.

V. Conclusion

We have demonstrated the UPPAAL CORA model checking tool for energy-aware scheduling of FIR filters. We used core logic time- and power metrics for the Altera Cyclone IV FPGA derived empirically for the embedded adder/multiplier. We found that allocating more HW leads to a decrease in the overall execution time until the speed-up rate starts to level off due to an imbalance between HW resources and inherent

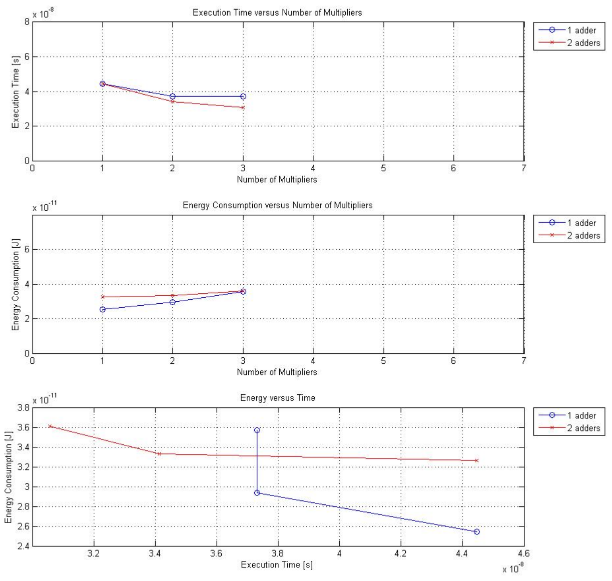

Fig. 11. Scheduling results for the symmetric structure.

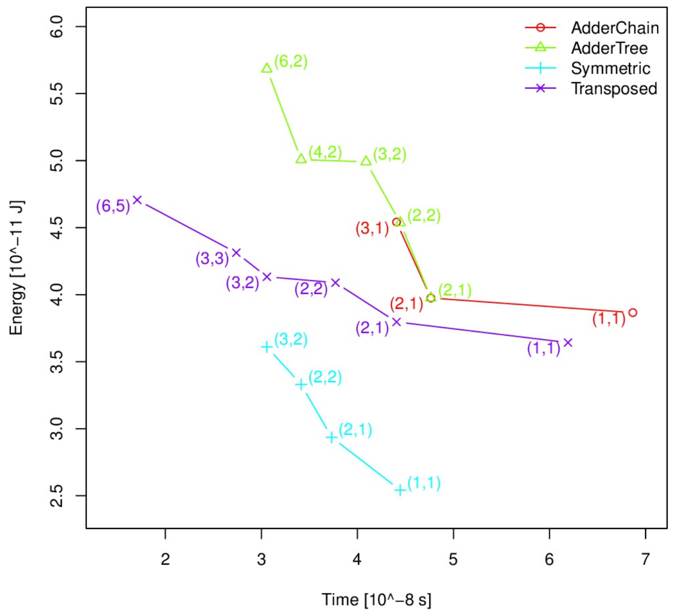

Fig. 12. Pareto Fronts for the four different structures, (# mult, # add).

parallelism. We found that the absolute energy consumption increases with more allocated HW which basically relates to the idle-power overhead. At the same time, we saw that the relative energy consumption among the various HW allocations is decreasing with higher number of functional units. This is a very interesting finding as it emphasizes that idle power should be seriously considered during HW allocation. Increasing HW allocation should therefore be done such that we reach a proper trade-off between "as much HW as possible doing specific computations", and "as little HW as possible sitting idle burning static power".

Concerning the filter structures, we found that there are significant differences between their time-and-energy characteristics. In order to do a fair comparison, we derived the Pareto Optimal time-and-energy curves for the structures, Figure 12. We conclude that the two transversal filters are those with the lowest (but also somewhat comparable) performance – energy as well as execution time. The transposed structure is superior to both of the transversal filter structures, and it spans a quite large interval, both on the energy- but in particular on the time axis. It is therefore fair to conclude that the transposed form is a sound candidate when it comes to choosing a structure which can trade off time and energy over large intervals. Further, among all structures the transposed form is the one which enables the shortest execution time.

Finally, the symmetric structure represents the best solution in terms of energy consumption. This is somewhat expected as the structure holds half the amount of multiplications as compared to the other structures, and thus it is an obvious candidate to choose for an SDR front-end, given that the filter can be designed with a linear phase response. All our experiments are conducted for filters with order $N = 5$ but since all four structures scales regularly in their topology with increased N, we can expect our results to also apply for higher N-values.

REFERENCES

[1] M.-U.-R. Awan *et al.*, "Polyphase filter banks for embedded sample rate changes in digital radio front-ends," *Communications, ZTE*, vol. 9, no. 4, pp. 3–9, 2011.

[2] M. Mehendale *et al.*, "Low-power realization of fir filters on programmable dsps," *Trans. on VLSI Systems, IEEE*, vol. 6, no. 4, pp. 546–553, 1998.

[3] G. Xu *et al.*, "Low power design for fir filters," *10th Int. Conf. on ASIC (ASICON), IEEE*, pp. 1–4, 2013.

[4] E. A. Lee *et al.*, "Static scheduling of synchronous data flow programs for digital signal processing," *Computers, IEEE Transactions on*, vol. 36, no. 1, pp. 24–35, 1987.

[5] UPPAAL, "Uppaal - an integrated tool environment for modeling, validation and verification of real-time systems," http://www.uppaal.org/.

[6] P. Koch, "FPGA energy profiles for arithmetic functions," Deliverable D4.2, SENSATION Project, EU, Tech. Rep., Oct. 2014.

[7] E. R. Wognsen *et al.*, "Energy-aware scheduling of FIR filter structures," Deliverable D4.3, SENSATION Project, EU, Tech. Rep., Oct. 2015.

[8] A. V. Oppenheim *et al.*, *Discrete-time signal processing*. Pearson, 2014.

[9] N. Shenoy, "Retiming: Theory and practice," *Integration, the VLSI journal*, vol. 22, no. 1, pp. 1–21, 1997.

[10] Altera, "De2-115 development board, users manual," http://www.altera.com/education/univ/materials/boards/de2-115/unv-de2-115-board.html.

[11] A. Technologies, "N6705b dc power analyzer, users guide," http://www.keysight.com/en/pd-1842303-pn-N6705B/dc-power-analyzer-modular-600-w-4-slots.

[12] ITRS, "International technology roadmap for semiconductor," http://www.itrs.net/.

[13] K. G. Larsen *et al.*, "As cheap as possible: Efficient cost-optimal reachability for priced timed automata," in *Computer Aided Verification, Proc. 13th International Conference, CAV*, vol. 2102. Springer, 2001, pp. 493–505.

[14] L. Aceto *et al.*, "A cost/reward method for optimal infinite scheduling in mobile cloud computing," in *Proc. 12th Int. Conf. on Formal Aspects of Component Software*, 2015, pp. 95–112.