2014 17th Euromicro Conference on Digital System Design

Data- and State-Dependent Power Characterisation and Simulation of Black-Box RTL IP Components at System Level

Daniel Lorenz, Kim Grüttner
OFFIS – Institute for Information Technology
Escherweg 2
26121 Oldenburg, Germany
{lorenz, gruettner}@offis.de

Wolfgang Nebel
Carl von Ossietzky Universität
Ammerländer Heerstr. 114-118
26121 Oldenburg, Germany
nebel@informatik.uni-oldenburg.de

***Abstract*—Due to the increasing algorithmic complexity of todays embedded systems, the consideration of extra-functional properties becomes even more important. Extra-functional properties such as timing, power consumption, and temperature need to be validated against given requirements on all abstraction levels. For timing and power consumption at RT- and gate-level, several techniques are available, but there is still a lack of methods and tools for power estimation and analyses at electronic system level (ESL) and above. Existing ESL methods in most cases use state-based methods for power simulation. This may lead to inaccurate results, especially for data-dependent designs. In this paper, we extend the Power State Machine (PSM) model for black-box RTL IP components with a mechanism that employs data-dependent switching activity using the Hamming distance (HD). In pipelined designs, we do not only consider the input HD but also the HDs of the internal pipeline stage registers. Since these registers of black-box IP are not observable from the outside, our model derives the internal HDs from previous input data. The results show that our extension achieves up to 38% better results than the previous PSM approach and up to 35% better results compared to a model considering only the input HD.**

I. Introduction

There is a high demand for mobile embedded applications with increasing complexity and performance, as well as long runtimes, which leads to increasing development time and costs. At the same time, limited battery capacitance reduces runtime of high performance mobile devices. To tackle these challenges, the system must be simulated as a whole, including timing and power properties. Furthermore, it is necessary to be able to analyse the expected power consumption of all system components for realistic system use-cases, e. g. in system-level simulation runs.

This includes the consideration of energy consumption of the processor cores and its integrated power management capabilities, plus the energy consumption of additional hardware components on the platform, e. g. buses, memories, dedicated hardware accelerators. Most of them are not developed from scratch, but reused from previous designs or bought from IP vendors. Usually, these IP components do not contain any information about their power consumption. In some cases data sheets are available that allow calculating an estimate of the power consumption in dependence of parameters such as clock frequency, supply voltage, and target technology. However, information of these data sheets can hardly be used in functional system-level simulations, as they do not consider any dynamic effects such as data and state dependencies.

To enable power management evaluation of the entire system on high abstraction levels, a holistic approach for energy estimation using virtual platforms on electronic system level (ESL) is required [1]. In this paper, an extension of the PSM concept (state-dependent power modelling approach) for data-dependent characterisation and simulation of black-box RTL IP component's power consumption is presented. Data-dependent switching activity is represented by the Hamming distance (HD) signal metric. For pipelined IP components, we propose to consider the input HD and the internal pipeline stage registers' HDs. The internal state and data-dependent switching activity and energy consumption of a component is approximated by observing and interpreting the interaction with its environment through its ports. This way, internal pipeline registers are not observable and thus lead to the necessary abstraction of deriving and predicting internal pipeline stage register HDs from previous input port data. This property allows us to use existing executable black-box IP components and plug our PSM model into commercial simulation environments. Our current SystemC-based proof-of-concept PSM implementation can be used and mixed with VHDL, Verilog and SystemVerilog components using co-simulation techniques.

Section II gives an overview of related work for modelling energy consumption in SystemC. Our basic approach for describing and simulating *Power State Machines* (PSM) is presented in Section III. We extend the PSM model to enable modelling of data-dependent power consumption in Section IV. The extended characterisation flow is explained in Section V. In Section VI the energy consumption of different IP components is characterised, modelled and simulated using PSMs. The resulting power traces, obtained from PSM model execution in SystemC, are compared against energy simulations on gate-level with respect to accuracy and simulation speed. Furthermore, the basic PSM approach is compared with the data dependent extension proposed in this work. Section VII closes with a summary and outlook on future work.

978-1-4799-5793-4/14 $31.00 © 2014 IEEE
DOI 10.1109/DSD.2014.89

II. Related Work

In the past, several methods for estimating the energy consumption on different abstraction levels have been investigated. It became evident that early design decisions offer the greatest potential on power savings [2]. In [3], Power State Machines was proposed for the first time in system-level modelling. State transitions are controlled by inputs of a power manager. Instead of a definite energy consumption, a state can have a maximal energy consumption, too. Its activity level is controlled by a power manager. Costs for state transitions are modelled by delays.

Since dynamic energy consumption emerges from switching activity, in [2] a concept for SystemC was presented where activity is observed at the communication interfaces. A process is notified about every `default_event` of the communication channels and traces the activity. This method is very transparent and produces little overhead, but neglects the activity inside the component which can definitely behave different than the activity in the communication channels.

With *Powersim*, a new approach was recently proposed in [4]. In this case, the model does not have to be changed because a modified SystemC simulation kernel is used. The power model of the system is loaded at the beginning of the simulation, which performs an activity estimation at every access on a SystemC data type. Due to the modified SystemC simulation kernel, this approach cannot be integrated in commercial design environments, which have a proprietary SystemC kernel that usually cannot be changed. Additionally, an estimation of communication, especially of abstract models (e. g. TLM), is possible only with restrictions because merely the assignment operator can be used for power annotation.

The past has shown that modelling and estimating the energy consumption through state machines with adaption to the environment is a promising concept. In [5], this approach is used with SystemC/TLM to invoke a state transition in a state machine by calling a method at corresponding source code location in the functional model. Energy consumption can be adapted to Dynamic Voltage and Frequency Scaling (DVFS). The disadvantage of this approach is its invasiveness in the functional model and thus it can hardly be used for IP components, especially if they are black-box IP components.

The authors of [6] present an approach that models the power consumption of interconnects at system level. Compared to our approach, they also take into account the data-dependent power consumption to get better results. To improve simulation performance, they characterise the input data stream and use the resulting switching probabilities. This has the disadvantage that for every new data stream the parameters have to be characterised again. In contrast, our model needs to be characterised only once.

In [7] a methodology was proposed to annotate white- and black-box components with state-tracing mechanisms. The idea is to find a factor for each state representing the resulting dynamic power. To determine the factors, these were calibrated with a linear regression to a post-layout gate-level power trace. The disadvantage of this method is that linear correlations can lead to states which are not annotated with power factors and thus induce a significant error in the resulting power trace. It is shown that the calibration data has a large impact on the accuracy, but it is not described how to generate calibration data to increase the accuracy.

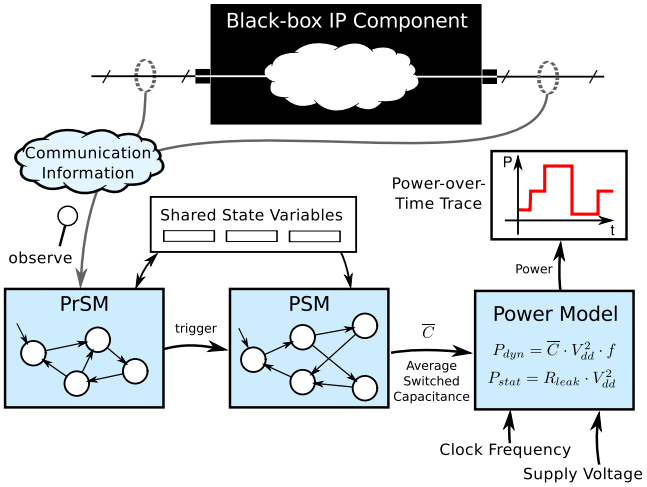

Fig. 1. PSM approach overview for non-invasive simulation of energy consumption

III. PSM Approach

An essential requirement for complete system power simulation is the power estimation of all components including black-box IP components. The internal structure of a black-box component cannot be annotated or modified to record switching activity in order to derive energy consumption. For this reason, the internal state of an IP component is abstracted, based on a correlation of its observable I/O behaviour with the estimated power consumption. The estimated energy consumption can be obtained from data sheets, inferred from an estimation of design size or the functional complexity of the design (top-down), or extracted from lower level (e. g. register transfer or gate level) simulations (bottom-up). With the latter approach, the most accurate model can be built [5]. For this reason, we abstract power values of gate-level simulations to create PSM models.

Figure 1 gives an overview of our I/O observation-based approach for annotating energy information at black-box IP simulation models. The ports of the IP component are observed over time to approximate the internal functionality. Based on these observations, a *Protocol State Machine* (PrSM) is controlled. The main task of the PrSM is to trigger state transitions in the PSM based on the observation and interpretation of the interaction between component and environment. By modelling the interdependencies between I/O and interal states the PrSM extracts the energetic relevant events to orthogonalise the communication and functional artefacts of the non-functional PSM model. This may lead to a reduction of complexity in the PSM because it only describes the different internal operation modes, whereas the PrSM covers the access protocol of the component. Furthermore, the separation of PrSM and PSM has the advantage that components with the same access protocol and different internal implementations could use the same PrSM, only the PSM has to be changed.

Furthermore, there may be some functional state transitions in the IP component that cannot be detected by observing the

interaction with the environment, e. g. if a component finishes its calculation, it may not send data indicating this through one of its ports. For this reason the PSM and PrSM are modelled as timed automata to executes state transitions after a given delay/timeout.

PSM and PrSM are each modelled as an Extended Finite State Machine (EFSM) which allows the extension of an simple FSM with (shared) state variables to reduce the complexity. EFSMs extend the state transition with enabling functions and update functions. Enabling functions check conditions on the state variables. Only when the condition is true, the transition is executed. Update functions modify the content of state variables if the related transition is executed. The input parameter for the update function can be both input symbols of the state machine and the values of state variables. The PSM has the restriction that it only allows enabling functions and not update functions, i. e. the PSM can only read the shared state variables but not modify them [8].

As described, PrSM and PSM allow for transitions to be executed after a defined delay. Therefore, the automata are modelled as an Extended Communicating Event Clock Automata (ECECA) which is a combination of EFSM with Communicating Finite State Machines (CFSM) [9] and Event Clock Automata (ECA) [10]. An ECA introduces clocks that are tightly associated with certain symbols of the input alphabet. An ECA may have event-recording and event-predicting clocks. An event-recording clock x_a is always reset when the automaton is triggered with the input symbol a. An event-predicting clock y_b is reset to a nondeterministic negative value when the automaton is triggered with the input symbol b. The value is checked for 0 at the next occurrence of input symbol b. Due to this association, every non-deterministic ECA can be transformed into a deterministic ECA which is necessary to implement it as executable model. The transitions of an ECA are annotated both with input symbols and clock constraints [10]. For enabling synchronisation between PrSM and PSM we use the channel mechanism of CFSMs presented in [9]. Instead of a full-duplex channel, we utilise an unidirectional channel.

The input alphabet of the PrSM is triple $\Sigma_{PrSM} = \{D, T, p\}$, where D is a finite set of payload data, T a finite set of timing events and p the observation port. The payload data is the data which is transmitted over the observed port, which includes metadata such as the bus address and byte enable signals as well as control and configuration data. The timing events are consecutive timestamps related to the simulation time. The port explicitly defines to which port the observed information belong. A transition in the PrSM is expressed as $\lambda_{PrSM} = \{s, e, g, a, o, s'\}$ with source state s, target state s', input symbol $e \in \Sigma_{PrSM}$, guard g, variable update a, and transmission message o. When the PrSM is triggered by an input symbol, the guard executes the enabling function and if it evaluates to true, the update function is executed and the related message is transmitted. Then the PSM is triggered by the received message. In contrast to the PrSM the PSM can only read the shared state variables and therefore does not have variable updates. Formally expressed the PSM transition is the tuple $\lambda_{PSM} = \{s, e, g, o, s'\}$. The output alphabet of the PSM Γ_{PSM} is a finite set of switched capacitances.

PSM and PrSM are sharing state variables which means that the state variables of the PrSM V_{PrSM} are the same as those of the PSM V_{PSM}. This leads to the following special relationship between PrSM and PSM. As previously mentioned, the state variables are used to share state variables and reduce complexity in the automata. Since the PSM is able to access the same variables, this implies that the transfer messages of the synchronisation channel between PrSM and PSM is extended by the same value range. Hence, a dynamic state variable dependent output can be applied, meaning that every extended PSM state, i. e. the PSM state in combination with the current values of the state variables, has a unique and deterministic output.

Based on the energy characteristics of the IP component, the dynamic power model can be created. To determine the dynamic power consumption $P(t)$ as function of discrete time $t \in \mathbb{N}$ the well-known formula

$$P(t) = \frac{1}{2} V_{\mathrm{dd}}^2 f C \cdot \alpha(t) \tag{1}$$

can be used, where C is the total switched capacitance, which is always constant, V_{dd} the supply voltage, f the clock frequency, and $\alpha[t] = \{x \mid 0 \leq x \leq 1\}$ the switching activity (i.e. fraction of overall circuit switching) for the clock cycle at time t.

Since the real switching activity $\alpha(t)$ cannot be obtained at system-level, a set of macro states $S = s_1, \ldots, s_n$ is defined in the PSM to distinguish between characteristic operation modes, which are time dependent. Thus, for a given use-case the time $t \in \mathbb{N}$ can be associated with the operation modes $s \in S$, $\varphi : \mathbb{N} \to S$. To each operation mode $s_i \in S$ an average activity is associated, describing the arithmetic mean activity $\alpha_i \in A$ for the time when functional mode s_i is active:

$$\alpha' : S \to A \quad \text{with } \alpha_i = \frac{1}{\Delta t(s_i)} \sum_{t=t_{\mathrm{beg}}(s_i)}^{t_{\mathrm{end}}(s_i)} \alpha(t) \tag{2}$$

Derived of (1), we get the average energy consumption $\overline{P}(t)$ over time:

$$\overline{P}(t) = \frac{1}{2} V_{\mathrm{dd}}^2 f C \cdot \alpha'(\varphi(t)) \tag{3}$$

Supply voltage and frequency are parameters depending on the power domain and the current strategy of the power manager. For this reason, we abstract from these parameters and get the average switched capacitance over time:

$$\overline{C}(t) = \frac{1}{2} C \cdot \alpha'(\varphi(t)) \tag{4}$$

In (4), the output of the PSM is described. This formula is the basis for deriving the average switched capacitance during simulation to generate a power-over-time trace.

As described above, it is possible to extend the PrSM and PSM with shared state variables V, which are modified only by the PrSM and can be used to store specific configuration data as the current function choice (e. g. MD5 vs. SHA256 in a hashing module). This modification facilitates an easier management of context-aware information being shared between PrSM and PSM. With this modification, the output of the PSM also may depend on the shared state variables. The determination of the average switching activity from (3) and (4) has then to be modified to $\alpha'(\varphi(t), V(t))$. I. e. state transitions, delays, and output of the state machines can depend on shared state variables. As an example, if a shared state

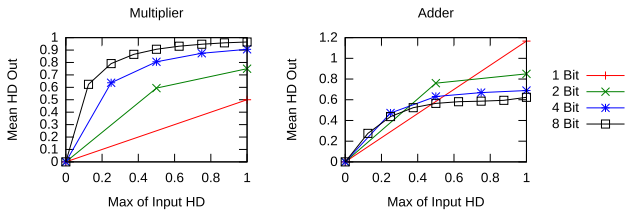

Fig. 2. Propagation of Hamming distances through multipliers and adders of 1 to 8 bit input bit width. As input Hamming distance the maximum of both input parameters is taken. The output Hamming distance is the mean of all occurrences.

variable describes the current clock frequency of the module, delays in the PSM can be adapted accordingly to the clock frequency. More details of the basic PSM approach can be found in [11].

IV. Data Dependency Extension

As stated in Section III the PSM model can only model state-based power consumption. Many designs have data-dependent power consumption. The reason for this is in most cases that the amount of switching activity in the component can heavily depend on the switching activity at the inputs of the component. To tackle this problem the extended PSM concept models this data dependency in the PSM model. The idea behind this concept is to extend the power states of the PSM with a data-dependent output instead of a constant output. That means that the output depends on the data switching at the inputs and/or outputs of the observed component. Activity at the inputs is the switching of the bits of two consecutive data vectors. This is known as Hamming distance (HD) which is the number of positions at which the corresponding value is different between two data vectors of the same size. Let Σ be a finite alphabet with $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in Σ^n, the HD between x and y is defined as:

$$\Delta(x, y) := \#\{j \in \{1, \ldots, n\} \mid x_j \neq y_j\} \tag{5}$$

Many designs process data in parallel in pipeline stages. Hence, when considering only the Hamming distance at the inputs of the component this could lead to a huge error. The problem is that the user has no knowledge about the internal structure in black-box components and thus about the internal data propagation and Hamming distance. We assume for internal components between two register stages that the output HD is linear dependent of the input HD. For two or more input components the maximum of the individual input HD has to be taken. Figure 2 exemplary shows the results for adders and multipliers with a bit width of 1 to 8 bit. The HD shown is relative to the input bit width, which is the reason for a mean output HD of 1.2 in the adder graph. Since it is not known which combinatorial blocks are used, it is legal to assume a linear dependency between input and output HD.

To enable data-dependent modelling of the power consumption, some assumptions were made related to the nature of the relationship between power consumption and switching activity at the inputs:

1) Many components need more than one clock cycle or even a dynamic number of clock cycles until the input data is processed and corresponding outputs become readable. In each clock cycle the data is processed in specific pipeline stage, i.e. the combinational logic between two register stages. Each of these pipeline stages contributes to the overall power consumption.
2) The amount of the contributed power consumption is linear dependent of the switching activity of the logic's input data. Even if data is modified in the logic blocks, the switching activity will strongly depend on the switching activity of the input data.
3) Based on the already described assumptions relating linear HD propagation through combinatorial logic, it is assumed that the switching activity keeps constant over all pipeline stages even if data is significantly modified. According to this assumption, the HDs inside the component can be derived of the previous input data for black-box components.
4) Even if no activity is at the inputs of the component, the component consumes dynamic power, which comes from the clock tree induced switching activity which is always present as long as the component is not clock gated.

These assumptions lead to the following formula which expresses the data-dependent output for a power state:

$$C = \sum_{i=1}^{m} (n_i \cdot hd_i) + c \tag{6}$$

where C is the average switched capacitance per clock cycle, n_i the characterised linear parameter and hd_i the Hamming distance of pipeline stage i, and c the characterised constant part.

To apply this data-dependent power consumption characteristic in the PSM model no modifications in the model are necessary. The model already brings all necessary features that are:

- PrSM can store input values in shared state variables
- the PrSM can read and modify shared state variables
- the PSM can read the shared state variables
- the output of the PSM can depend on the state and additionally on the shared state variables

These features are utilised as depicted in Figure 3.

As stated in assumption 3, the HD keeps nearly constant over the pipeline stages. Therefore the same data can be used to calculate the HD even if the data is changed in the component. Hence, the input data has to be saved for the subsequent clock cycles to derive the HD in the register stages which can not be accessed from outside the component. Therefore, for each register stage in the component a representative shared state variable exists in the PrSM, which is initialised with 0 because the exact initial value is not known. When new input data is available, all data is shifted into the next shared state variable using read and write operations of the PrSM. The new data is saved in the related shared state variable by the PrSM. The last data is removed by the shift operation just as it happens in the observed component. The PSM is triggered by the PrSM that new input data is available and either the power state changes or simply the output power has to be recalculated. In the last

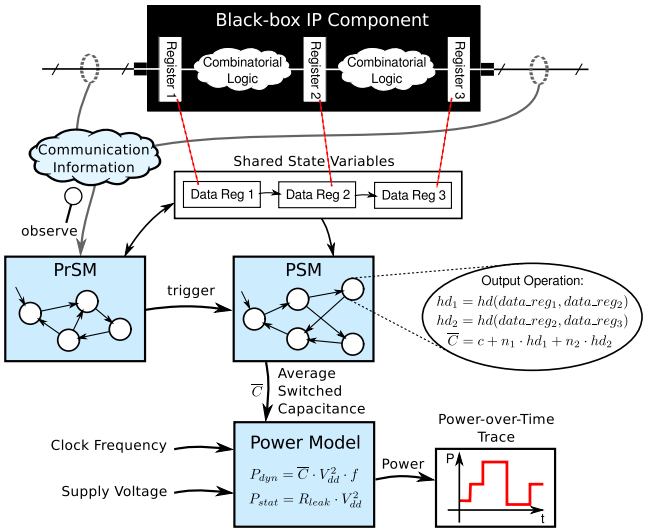

Fig. 3. Application of data-dependent power consumption in PSM model

case, a transition with equal source and target state is executed. The PSM then calculates the HDs of each pipeline stage based on the new register values. The reason is that one clock cycle before the value *a* of register stage X was stored in the previous register stage Y, which now stores the value *b*. So the PSM knows that the values switched from *n* to *m* in register stage Y just as in the combinational logic before. With the HDs and the characterised linear parameters the output value of the PSM can be calculated. It is important to mention that for different power states different or even no linear parameters need to be characterised because combinational logic in the pipeline stages may activated in another way. A simulation speed improvement can be achieved if the PrSM calculates the HDs because in the HD needs to be calculated only once. But this mixes up the semantic separation of power, which depends on the HD, represented by the PSM, and the functional behaviour, represented by the PrSM.

If a component needs an input- or state-dependent amount of clock cycles for processing one input vector, then these information has to be regarded in the PSM as well as in the characterisation. In most cases this is caused by a loop or conditional execution in the component, which must be reflected in the shared state variables and the PrSM to model the loop behaviour. Feedback loops have the disadvantage that HDs are more unpredictable because the data of the feedback loop and the input data are merged and thus the HD has to merged, too. In this case, the user would use the average or the maximum of both HDs or would try to reimplement the original merging functionality if it is known and derive the correct HD. Depending on the design, either the average or the maximum of both HDs can result in lower errors. Hence, always both strategies should be characterised to get the better one.

In the following, the proposed approach is denoted as pipeline stage HD model (PSHDM). If only the input HD is used, the resulting model is the input HD model (IHDM). The PSM model without data-dependent extension maps mean power to every power state, hence it is denoted hereafter as mean power model (MPM).

V. Characterisation of Data-Dependent Power

To be able to use the PSM model with data-dependent power consumption, the linear parameters first have to be characterised using the characterisation process from [11]. For this characterisation process, it is crucial to use data with less or no switching activity if possible to eliminate the data-dependent ammount of the power consumption. Based on this, the characterisation for the data-dependent power consumption can be done. If the power consumption does not change when input bits switching activity differs, characterisation is done because the error of the simulation model cannot be further reduced with the following described process.

The characterisation process is divided into the following tasks depicted in Figure 4:

1) Identify the number of pipeline stages;
2) create a power trace with different input data stimuli and identify the data dependent power states; and
3) characterise linear parameters for each data dependent power state.

Step 3 is further divided in the following subtasks:

3.a) Create power traces with different HD combinations in the pipeline stages;
3.b) save the HDs of the pipeline stages with the corresponding time stamps;
3.c) assign the power values to HD combinations and calculate the average; and
3.d) calculate all linear parameters and the constant part with linear regression.

The number of pipeline stages in the design is important because this determines the amount of input data that has to be saved and thus the number of shared state variables. It has to be determined how many clock cycles are necessary until the results can be observed at the output, which can be obtained from the documentation or from a simulation. For each clock cycle of the I/O Delay, a combinational logic exists between input and output, which has to be regarded.

In the next step, the data-dependent states have to be identified. In a simulation, all states are traversed with different input data and thus different HDs. The activity of simulation is traced and transformed in a power-over-time trace. This trace is mapped on the trace of the power states. The states with high standard deviation and very fluctuating power values have a high probability for data dependence. In the worst case, even all states can be data dependent. In this case the characterisation of the PSM could already fail because due to the fluctuating values no power states can be extracted from the power trace. In that case, the functional states respectively PrSM states are used instead of power states for the data-dependent characterisation.

To evaluate the data-dependent part of the power consumption, the following steps have to be performed for each state which emerged as a data-dependent state. The next step is to find out how much dynamic power consumption each pipeline stage contributes to the complete power consumption. Therefore, a simulation is used that generates equal distribution of HD combinations in the various pipeline stages. As previously mentioned, this is not the real HD but the

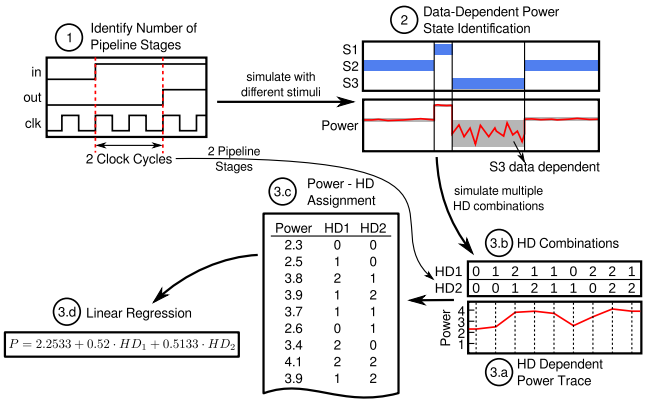

Fig. 4. Characterisation process for data-dependent power

assumed HD based on the previous input data. If possible, all available HD combinations should be simulated several times. For components with large bit width this is not possible in practical time, so these are simulated with a sufficient part of all HD combinations. The activity traced in this simulation is afterwards transformed in a power-over-time trace. Furthermore, the simulation also stores the HDs in the pipelines together with the corresponding time stamps.

Thus, the power values can be mapped to the individual HD combinations. If more than one power value for a HD combination exists, this is regarded in the linear regression to achieve the best fit.

According to assumption 2 that the power of each pipeline stage is linear dependent of the HD, a linear regression can be applied to relate power and HD data. Here it is important to consider a constant part in the linear regression because the assumption 4 is that always a constant part exists. After the linear regression, a parameter exists for the constant part and every pipeline stage.

Finally, these parameters can be tested against the power values used for the characterisation or new random power values. The more the input values are modified in the component and the more pipeline stages are present, the worse the resulting power values will be. To get better results, the only way is to reimplement the functions of the component in the PrSM and get correct values of the registers. But on one hand this produces very much overhead because functionality is duplicated and on the other hand this knowledge is not present in most cases, if black-box components are used.

VI. Experiments

In this section we show some examples of how the characterisation method is applied and the model is constructed. Afterwards we show the results of the proposed model denoted as pipeline stage HD model (PSHDM) in contrast to a gate-level power estimation and compare it with the models depicted in Section IV, input HD model (IDHM) and mean power model (MPM).

As an example we use an OTN (optical transport network) framer component, which is an industrial component of an internet routing module. This component analyses an unaligned input stream and tries to realign the stream depending on a key

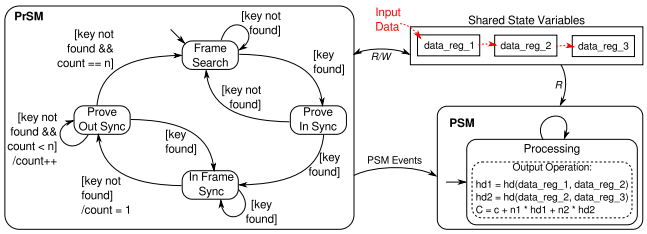

Fig. 5. PSM and PrSM model for the OTN framer module. The dashed arrows show the data flow through the shared state variables, which is accomplished by the PrSM.

which defines the begin of an OTN frame in the input stream. Each frame is divided in words numbered in lines and columns. The component has four states, *Frame Search*, *Prove In Sync*, *In Frame Sync*, and *Prove Out Sync*. Initially, the component is in the state *Frame Search*. If the key is found it is switched to state *Prove In Sync*. If another frame with key is found the state switches to *In Frame Sync*. When no key is found the component witches to the state *Prove Out Sync*. After n consecutive frames without key the component again switches to state *Frame Search*.

In the first step of the characterisation process the PSM is created without data-dependent states. Therefore, only input vectors without switching activity are used to determine the individual power states. Thereby it is considered that every of the above mentioned states is reached and that all state transitions are executed several times. The resulting power-over-time trace shows only little variations, which concludes that a single power state is sufficient for this component. In the next step the data dependencies have to be found. The analyses of the input/output behaviour reveal that the design has two pipeline stages but no feedback loops. Therefore, the HD between the input data and the previous input data is considered to for both pipeline stages. The design is simulated using various combinations of HDs in both pipeline stages. While simulation the resulting HDs of the pipeline stages are written into a text file and the activity is traced, which is converted to a power-over-time trace. The power estimation is done with a 65 nm ASIC technology library, a supply voltage of 1.1 V and a clock frequency of 50 MHz. The HD values are mapped to the power values and are calculated with the help of a linear regression the resulting model parameters. The resulting PSM/PrSM model for the design is depicted in Figure 5. It can be seen that the PrSM models the full functionality of the OTN framer, whereas the PSM contains only a single state which calculates the data dependent output capacitance. The shared state variables contain the input data of the current and the two previous clock cycles. The storing and shifting of the input data is done by the PrSM.

To test the resulting power model the RT level design is abstracted to a functionally identical SystemC module [12]. We implemented a library in SystemC to be able to annotate existing designs with the PSM/PrSM Model. The abstracted design is annotated with the PSM, PrSM and the shared state variables. Then the model is configured with the characterised parameters. To evaluate the data-dependent model, we simulate the SystemC model with the data used for the characterisation and three additional test cases. The stimuli data for the first test case have the same characteristics as the characterisation data,

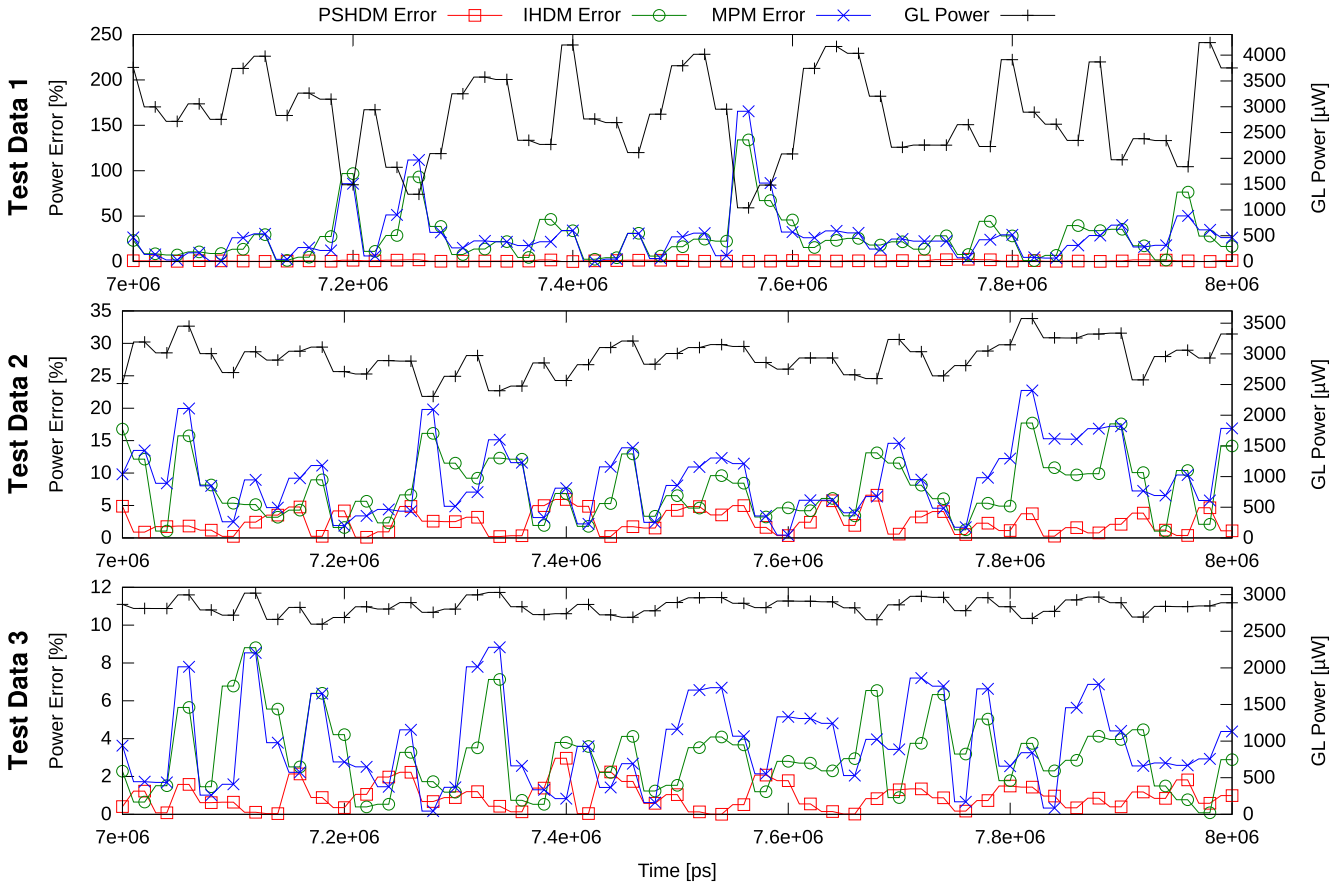

Fig. 6. Comparison of PSHDM, IHDM, and MPM error for the OTN framer in relation to the gate-level power trace for test data 1 to 3. The error rates are given in ralation to the gate-level power on the left y axis. The gate-level power is given the right y axis.

TABLE I. SIMULATION RESULTS FOR OTN FRAMER

Stimuli	Mean Relative Error			Execution Times				PSHDM to GL Speed-Up	PSHDM to MPM Overhead
	PSHDM	IHDM	MPM	PSHDM	IHDM	MPM	GL		
Characterisation Data	1.0085%	36.100%	39.504%	0.573 s	0.573 s	0.556 s	37 min	3874×	3.06%
Test Data 1	0.99592%	35.102%	37.836%	0.575 s	0.564 s	0.555 s	37 min	3860×	3.60%
Test Data 2	2.0443%	6.7122%	7.9280%	9.812 s	9.791 s	9.661 s	668 min	4085×	1.56%
Test Data 3	1.1541%	6.7575%	8.6614%	0.655 s	0.654 s	0.627 s	41 min	3755×	4.47%

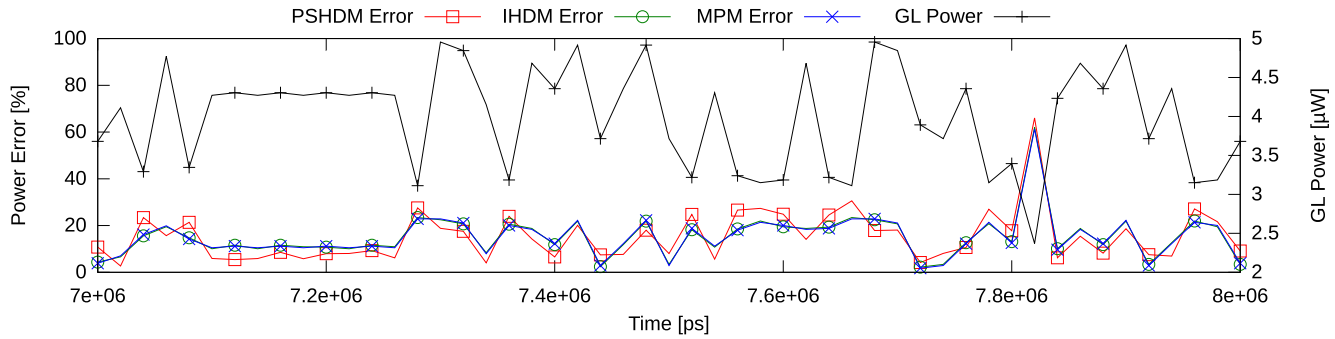

Fig. 7. Comparison of PSHDM, IHDM, and MPM error for the convolution encoder in relation to the gate-level power trace for the characterisation data. The error rates are given in relation to the gate-level power on the left y axis. The gate-level power is given on the right y axis.

i. e. a stream with randomly chosen HDs to hit a wide range of HDs. The second test case includes only stimuli data within a specific range of the value range and which are related to each other. That means that only small differences are between

two consecutive values as it appears in raw audio data. In the third test case the stimuli data is a digitally compressed data stream. Characteristically for these streams is that almost every bit has a switching probability of 50% and values are in the complete value range. The power of each simulation is traced and compared to the gate-level power data.

The results are shown in Table I. As it can be also seen in Figure 6, the power error is very low for this component compared to IHDM and MPM. This shows that not only the power error can be reduced drastically but also that the system-level power has a very high correlation with the gate-level power. The system-level power reflects the gate-level power behaviour very well if data-dependent power consumption is present and all pipeline stages are modelled. It can be seen that the power for the real-world examples are about $8\,\%$ in the MPM. Hence, in early design stages, when simulation performance is more important and highly accurate power values are not necessary, the MPM model can be used with an appropriate error. In later design stages, when power gains importance, the data-dependent model can be used.

Comparing the execution times for simulating the power consumption of the component, a significant speed up is achieved in the SystemC simulation in contrast to gate-level simulation. The most consuming task at gate level is the power estimation because for every single clock cycle the power is estimated individually. Comparing the execution times of the system-level models, it can be observed that the data-dependent extension only lead to a minimal overhead for this component.

As a second example we took a recursive convolution encoder with eight states and two outputs. First, a power-over-time trace was generated, which showed a fluctuation of power. We assumed data dependencies of the individual combinational blocks and used the knowledge of the functionality that data is fed back to merge the HD as described in Section IV. After building the model and characterising the required parameters, we tested the resulting model against the real gate-level simulation power trace which is shown in Figure 7. As can be seen, the error was very high and nearly no correlation between the values of the model and the gate level power exists. We compared the mean relative error of PSHDM, IHDM, and MPM in relation to the gate level power. Here the mean power model performed best with a relative mean error of 16.648% (PSHDM: 17.373%, IHDM: 17.313%). The reason is that most data is fed back through loops bluring the input data, which reduces the data dependency to a minimum and leads to an average power value, which has a smaller standard deviation. Consequentially, if there exist multiple feedback loops like in encoding and encryption components, the data dependency and the standard deviation of the average power value decrease. In this case, a usual power state with constant output is sufficient. Furthermore, the linear regression tries to reduce the square error of all data, which can lead to worse power value results if few power values reveal a huge deviation from the linear curve.

VII. Conclusion

In this paper, we presented an extension of the PSM model for data- and state-dependent power characterisation and simulation at system level. The model can be used especially for existing black-box RTL IP components. The key feature of this model is that it not only considers the data-dependent switching activity at the inputs of the component, but also the assumed switching activity in the internal pipeline stages which are not observable from outside. The experiments show that this could reduce the error of the model significantly if the examined component reveals data-dependent power. We also showed for which class of components the "standard" PSM model is sufficient to get acceptable results.

Acknowledgment

This work has been supported by the EnerSave Project, funded by the German BMBF under Grant Agreement 16BE1102.

References

[1] K. Grüttner, P. A. Hartmann, K. Hylla, S. Rosinger, W. Nebel, F. Herrera, E. Villar, C. Brandolese, W. Fornaciari, G. Palermo, C. Ykman-Couvreur, D. Quaglia, F. Ferrero, and R. Valencia, "The COMPLEX reference framework for HW/SW co-design and power management supporting platform-based design-space exploration," *Microprocessors and Microsystems*, vol. 37, no. 8, Part C, pp. 966 – 980, 2013, special Issue on European Projects in Embedded System Design: EPESD2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933113001221

[2] C. Walravens, Y. Vanderperren, and W. Dehaene, "ActivaSC: A highly efficient and non-intrusive extension for activity-based analysis of SystemC models," in *46th ACM/IEEE Design Automation Conference (DAC'2009)*, Jul. 2009, pp. 172–177.

[3] L. Benini, R. Hodgson, and P. Siegel, "System-level power estimation and optimization," in *International Symposium on Low Power Electronics and Design*, ser. ISLPED'98. Monterey, CA, USA: ACM, Aug. 1998, pp. 173–178.

[4] M. Giammarini, S. Orcioni, and M. Conti, "Powersim: Power estimation with SystemC," in *Solutions on Embedded Systems*, ser. Lecture Notes in Electrical Engineering. Springer Netherlands, vol. 81, pp. 285–300.

[5] H. Lebreton and P. Vivet, "Power modeling in SystemC at transaction level, application to a DVFS architecture," in *IEEE Annual Symposium on VLSI*, ser. ISVLSI'08. IEEE Computer Society, Apr. 2008, pp. 463–466.

[6] M. Gag, T. Wegner, and D. Timmermann, "System level power estimation of system-on-chip interconnects in consideration of transition activity and crosstalk," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*, ser. Lecture Notes in Computer Science, R. Leuken and G. Sicard, Eds. Springer Berlin Heidelberg, 2011, vol. 6448, pp. 21–30.

[7] S. Schürmans, D. Zhang, D. Auras, R. Leupers, G. Ascheid, X. Chen, and L. Wang, "Creation of ESL power models for communication architectures using automatic calibration," in *Proceedings of the 50th Annual Design Automation Conference*, ser. DAC '13. New York, NY, USA: ACM, 2013, pp. 58:1–58:58.

[8] G. J. Holzmann, *Design and Validation of Computer Protocols*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991, pp. 176–178.

[9] D. Brand and P. Zafiropulo, "On communicating finite-state machines," *J. ACM*, vol. 30, no. 2, pp. 323–342, Apr. 1983.

[10] R. Alur, L. Fix, and T. A. Henzinger, "Event-clock automata: a determinizable class of timed automata," *Theoretical Computer Science*, vol. 211, no. 1, pp. 253–273, 1999.

[11] D. Lorenz, P. A. Hartmann, K. Grüttner, and W. Nebel, "Non-invasive power simulation at system-level with SystemC," in *Power And Timing Modeling, Optimization and Simulation - 22nd International Workshop (PATMOS'2012)*, ser. Lecture Notes in Computer Science. Springer, Sep. 2012, pp. 21–31.

[12] D. Lorenz, K. Grüttner, N. Bombieri, V. Guarnieri, and S. Bocchio, "From RTL IP to functional system-level models with extra-functional properties," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, ser. CODES+ISSS '12. New York, NY, USA: ACM, 2012, pp. 547–556. [Online]. Available: http://doi.acm.org/10.1145/2380445.2380529