Design and Verification of Speed-Independent Multiphase Buck Controller

Danil Sokolov[†], Victor Khomenko[†], Andrey Mokhov[†], Alex Yakovlev[†], David Lloyd[‡]
{danil.sokolov, victor.khomenko, andrey.mokhov, alex.yakovlev}@ncl.ac.uk; david.lloyd@diasemi.com
[†]Newcastle University, UK; [‡]Dialog Semiconductor, UK

***Abstract*—Power regulators and converters impose high requirements on the latency and resilience of their control circuitry. In this paper we design a speed-independent multiphase buck controller based on a novel lazy token ring architecture, that allows overlapping the charging cycles of multiple phases as well as simultaneous activation of all phases to handle the sudden power demand. The advantages over traditional synchronous designs include reliable handling of asynchronous inputs from sensors, low-latency reaction to the changes in power demand (under-voltage, over-current, zero-crossing and high-load conditions), and more balanced utilisation of charging phases. The essential correctness properties of the developed controller have been formally verified, and the whole buck has been validated in the industrial settings using exhaustive simulation.**

I. Introduction and basic notions

A. Motivation

The market of consumer gadgets is dominated by digital electronics that processes discrete data. However, a small portion of components remain analogue to operate on continuous values, such as the energy flows. As energy is becoming the most valuable resource in modern electronics, the efficient implementation of such analogue components as power converters [1] is paramount for a wide range of applications, from extending the battery life of mobile gadgets to reducing the energy bill of large data centres. Responsiveness and robustness of power converters heavily depends on the implementation of their digital control circuitry – millions of control decisions need to be made each second and a single incorrect decision may cause a malfunction of the whole system or even permanently damage the circuit. For example, a 3MHz switching regulator is clocked around 473,364,000,000,000 times in 5 years of its operation [2].

The practical design problem associated with power converters [3] is partially related to the state-of-the-art synthesis methods which produce suboptimal solutions. Currently the same design methods and CAD tools are used for building both the data processing components and the power control circuits. Historically these methods and tools are optimised for synchronous circuits whose bursts of activity are driven by the frequency of a global clock signal. The clocked mode of operation is natural for the data processing; however, when applied to power control, it leads to either low responsiveness or power consumption overheads. On one hand the clocking frequency must be sufficiently high to promptly react to changes in the analogue power converter (sensor readings). On the other hand this causes waste of energy (useless switching of the global clock circuitry) when the sensors' readings change slowly. Furthermore, with the increasing number of asynchronous signals coming from power regulator sensors the probability of failure in synchronisers quickly grows and may become significant.

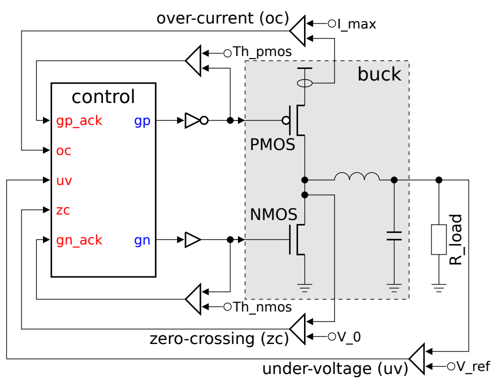
The power control could significantly benefit from the use of asynchronous logic [4] that does not rely on the global clock signal and operates at the pace determined by the current conditions. Thus such circuits are adaptable to the rate of changes in the controlled system and can react to the asynchronous signals from the sensors without the need for synchronisers. These benefits are already realised by the analogue engineers who are keen to use asynchronous circuits – at present they typically perform an *ad hoc* design of clock-less power control circuits and rely on exhaustive simulation to validate their correctness. This assemble-and-validate approach, however, is unproductive and prone to errors. To bridge this gap one needs to start from an unambiguous representation of the design intents that can be subsequently used for synthesis of the power control circuits in a correct-by-construction manner and for verifiable integration of the obtained control circuit into a power regulator system.

In this paper we explore a compositional approach to the design of the power control logic and formal verification for correctness of the obtained solutions. On this pathway we try to maximally reuse the existing synthesis tools, and apply them to a novel domain of interfacing the analogue electronics with digital asynchronous controllers. The importance of the compositional approach is paramount. The complexity of individual components is kept relatively low (up to 6-7 signals) in order to enable non-expert designers to understand the behaviour of the system in partial views as well as allow more effective interaction between the designer and the synthesis tools. The other aspect is enabling component reuse which is the key to productivity. An experienced designer can potentially come up with a more efficient solution at the level of larger components, but at cost of maintainability and clarity, which are important for designing highly reliable electronics. This design approach is applied to the development of a multiphase buck converter.

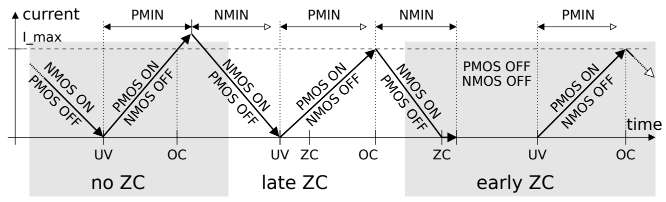
B. Basic buck converter

A basic power regulator comprises an analogue buck and a digital controller, as shown in Figure 1a. The controller operates the power regulating PMOS and NMOS transistors[1]

[1]In the textbook buck a diode is used instead of the NMOS transistor.



(a) Schematic.



(b) Informal specification.

Figure 1: Basic buck converter.

of the buck (using gp and gn outputs) as a reaction to *under-voltage* (UV), *over-current* (OC) and *zero-crossing* (ZC) conditions (uv, oc and zc inputs respectively). These conditions are detected and signalled by a set of specialised sensors implemented as comparators of measured current and voltage levels against some reference values (V_ref, I_max, V_0). Note that the gp and gn signals are buffered to drive the very large power regulating transistors (occupy more than 50% of the buck area) and their effect on the buck can be significantly delayed. Therefore, the controller is explicitly notified (by the gp_ack and gn_ack signals) when the power transistor threshold levels (Th_pmos and Th_nmos) are crossed.

The operation of a power regulator is usually specified in an intuitive, but rather informal way, e.g. by enumerating the possible sequences of detected conditions and describing the intended reaction to these events, as shown in Figure 1b. The diagram shows that UV should be handled by switching the NMOS transistor OFF and PMOS transistor ON, while OC should revert their state – PMOS OFF and NMOS ON (no ZC scenario). Detection of the ZC after UV does not change this behaviour (late ZC scenario). However, if ZC is detected before UV then both the PMOS and NMOS transistors remain OFF until the UV condition (early ZC scenario). Furthermore, for efficiency reasons the PMOS and NMOS transistors should be ON for at least PMIN and NMIN time respectively.

Note that ZC and UV are independent conditions that indicate separate physical effects and therefore the corresponding signals can happen in any order. UV indicates that the voltage supplied to the load has decreased below the reference value. ZC occurs when the coil current reduces to 0 and causes the NMOS transistor to switch OFF, so that the NMOS acts like a diode and only conducts in one direction.

C. *Multiphase buck converter*

A multiphase buck converter combines several pairs of PMOS and NMOS transistors (called *phases*) to power the same load, see Figure 2. The main advantages of this distributed design compared to a basic buck are faster reaction to the power demand, heat dissipation from a larger area, and decreased ripple of the output voltage [1].

The control circuit of a multiphase buck with N phases monitors the OC and ZC conditions of individual phases (inputs oc1,...,ocN and zc1,...,zcN respectively) and the voltage level at the load (uv and hl inputs). When UV is detected (the load voltage drops below V_ref value) the controller performs a charging cycle (switching the PMOS and NMOS transistors the same way as in a basic buck) at the currently active phase. The active phase is traditionally selected in a round-robin pattern by a generator of non-overlapping pulses that are derived from a relatively slow (in MHz range) master clock. If by the time the next phase is activated the UV condition still persists, a charging cycle is exercised on that phase too, thus helping the previous phase(s). This process is repeated until the power demand is met and the UV condition is reset, and is resumed upon detection of the next UV. Note that the UV condition is sampled on the activation of a phase only and thus a response to power demand may be delayed by a whole clock cycle of the master clock.

A special mode of operation is used to handle the high-load (HL) condition that indicates a sudden increase in power demand (the voltage drops below V_min value). In this mode the controller activates all the phases simultaneously and, as HL implies UV ($V\_min < V\_ref$), a charging cycle starts in all the phases. As a prompt reaction to a sudden power demand is crucial, sampling of the HL condition has to be made at a higher frequency than the master clock, which significantly complicates the design in the synchronous case.

D. *Main contribution*

The clock signals in a traditional design of multiphase buck are somewhat artificial – they are introduced for the sake of standard synchronous EDA tools. However, as explained above, the clocks limit the reaction time to power demand, introduce synchronisation overheads when sampling the sensors, and burn power when sensor readings change slowly. In this paper we present a design of a speed-independent controller for a multiphase buck. The clock-based phase generator is replaced with a token ring architecture, and the inputs from the voltage and current sensors are handled asynchronously eliminating thus the possibility of a synchroniser failure. The main contributions of this paper are:

- Compositional approach to the design of asynchronous power management circuits.
- Novel lazy token ring architecture for activation of charging phases.
- Formal specification of speed-independent multiphase buck control.
- Design of a component for sanitising input signals from the sensors (WAIT element).

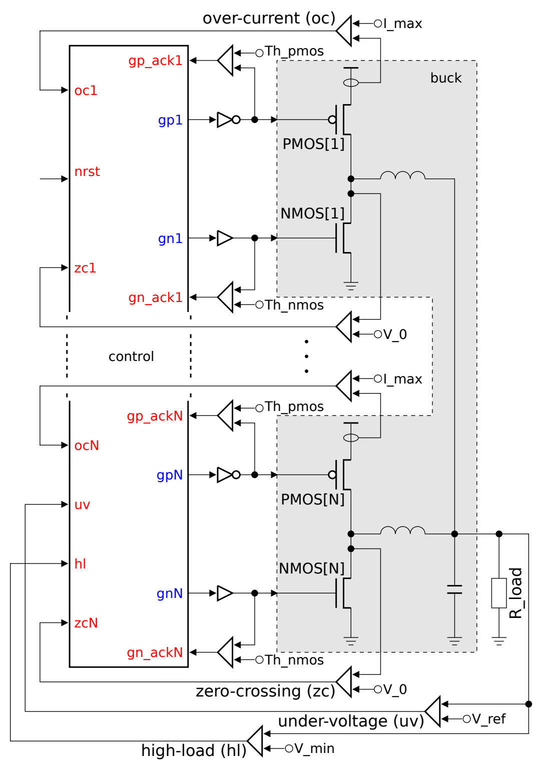


Figure 2: Schematic of a multiphase buck converter.

- Novel technique for eliminating redundant cycles of charging based on opportunistic merging.
- Rigorous verification of the synthesised circuit against the specification.

Previous work tackled the subject of deriving a formal specification from phase diagrams [5] and the reuse of SYNOPSYS tools to automate place-and-route and off-line testing of asynchronous power management controllers [6]. In conjunction with the work in this paper, the entire flow – from specification to layout and production testing – can be automated.

II. Design of multiphase buck controller

In this section we explain the general structure of our asynchronous multiphase buck controller and justify the high-level architectural decisions. Note that some details and features have been omitted in this paper for the sake of presentation and due to commercial sensitivity. In order to simplify the process of specification and reduce the synthesis and verification effort we partitioned the design into reusable simple modules. The modules communicate with each other in traditional asynchronous style by means of request-acknowledgement handshakes.[2] The naming convention for the handshake signals is as follows. A request signal starts with 'r' and an acknowledgement with 'a'. If a module has more than one channel then the second letter refines their semantics – 'i' for input channels, 'o' for output channels, 'd' for delay/timer

[2]The controller implementation can be optimised by combining several modules using parallel composition of their specifications and hiding the intermediate handshake signals. This *resynthesis* technique is well known [7].

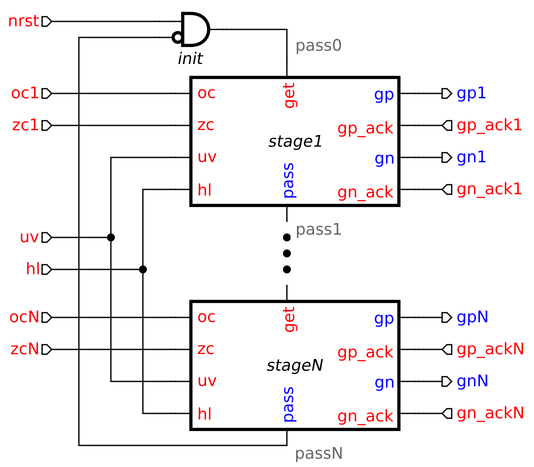


Figure 3: Ring architecture for the multiphase buck control.

interfaces, and 'p'/'n' for switching PMOS and NMOS power transistors respectively. Channels with the same semantics are distinguished by a numerical suffix. For visualisation purposes, the names of the signals are colour-coded: red for inputs and blue for outputs.

A. *Lazy token ring architecture*

A natural asynchronous replacement for the clock-driven round-robin activation of phases in an N-phase buck is a token ring with N identical stages, as shown in Figure 3. Each stage delays the token for at least a predefined duration of time (corresponds to the period of the master clock in synchronous design) before propagating it to the next stage. As the token enters the stage, it becomes active and may perform a cycle of charging at the corresponding buck phase.

The stages are composed in a ring with an inversion before the first stage (init gate that also handles the initialisation). Abstractly, a stage can be viewed as a delay, and thus an oscillator is formed: the useful work is done on the set phase of the oscillator that may be quite long, and the reset phase of the ring is fast.

We distinguish two kinds of token ring architectures: *busy* and *lazy*. A busy ring always propagates the token to the next stage after a predefined delay, even if the charging cycle has not been activated (similar to phase activation by non-overlapping pulses in synchronous design). This design, however, requires extra arbitrations (between the end of delay and UV), produces unnecessary switching activity (the token keeps moving around even if sensor readings do not change), and may result in uneven utilisation of buck phases. These drawbacks are addressed in a lazy token ring where the token is passed to the next stage only after both a cycle of charging has started and a predefined delay has elapsed. If there is no power demand then the lazy token stays in the currently active stage and there is no switching activity in the controller. Hence the arbitration between the end of delay and UV is not required, and the buck phases are utilised evenly. We focus on the lazy ring architecture in the rest of the paper.

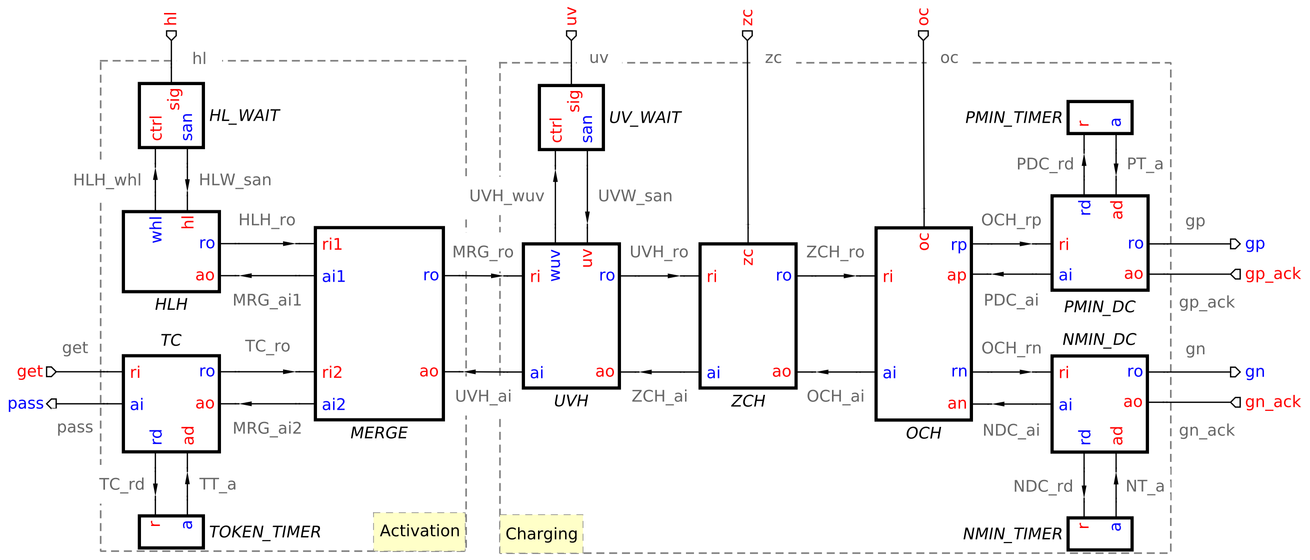


Figure 4: Single stage of the multiphase buck control.

B. *Activation and charging functions*

The architecture of a single token ring stage is shown in Figure 4. There are two distinctive functions for a token ring stage to perform: handling its activation and charging the buck. The stage may become active either when it receives a token from the previous stage or when the HL condition is detected. Upon activation, the stage listens for UV, ZC and OC conditions to perform a charging cycle on the buck. Note that the charging must not block the token propagation to the next stage, and thus the phase charging has to be decoupled from the stage activation.

The stage activation is partitioned into a high-load handler (HLH) and a token control (TC). The HLH module waits for the HL condition (whl/hl handshake interfacing a WAIT element that waits for the hl signal, see Section III-B for details) and activates the stage when it is detected. (Recall that all stages are activated by the HL condition.) The TC module waits for the token on its get input to issue the stage activation request. When the token is received, the token timer is started (rd/ra handshake) to delay passing the token to the next stage via pass output. The activation requests from HLH and TC modules are processed by the MERGE element [8] that combines two request-acknowledgement channels into one.

A naïve implementation of the MERGE element would perform arbitration between the two input requests and generate a separate output request for each of them. There are two drawbacks of this basic design:

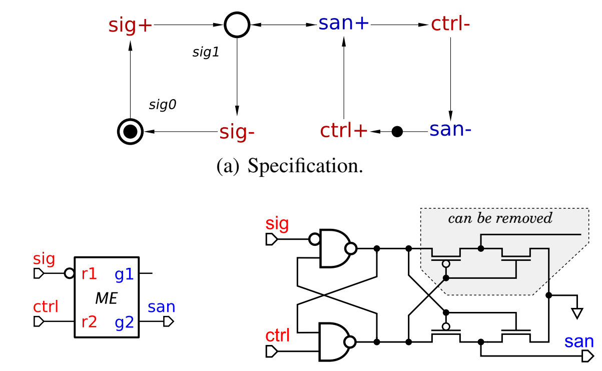
- The arbitration in the critical path that may slow down reaction to a critical HL condition. This can be addressed by moving arbitration, with its potentially slow metastability resolution, from the critical requests path to the non-critical acknowledgements path, and propagating the rising phases of the requests on the input channels to the output channel in the OR-causal way [9].
- Two cycles of charging, one for each of the activation requests, may be executed even if these requests arrive simultaneously. A better implementation would opportunistically bundle requests from different channels whenever they arrive sufficiently close to each other. The details of such a design can be found in [10].

Charging of the buck phase follows the same pattern as in the basic buck – the UV, ZC and OC conditions are monitored to make decisions about the state of PMOS and NMOS transistors in the buck. We implemented the handler of each condition in its own module – UVH, ZCH and OCH respectively. The UVH module also decouples token propagation from charging by giving an early acknowledgement to the MERGE element immediately after the UV condition is detected (the uv signal sanitised using a WAIT element). Note that the HL condition implies the UV condition, as they both are the results of comparisons of the same voltage with different thresholds. We exploit this in our design: when a stage is activated by the HL condition, the charging is still initiated by UV as in the regular case. However, in some anomalous situations when both HL and UV are asserted and immediately withdrawn, there may be insufficient time to latch uv in UV_WAIT, in which case a charging cycle is not initiated. This is unproblematic, as charging is not necessary in this situation, and the stage will not stall as the next UV condition will trigger a charging cycle.

In order to enforce the requirement for the minimum On time for PMOS and NMOS transistors of the buck, two instances (PMIN_DC and NMIN_DC respectively) of a special *Delay Control* (DC) module are used. DC module interfaces a timer (PMIN_TIMER or NMIN_TIMER) to delay the acknowledgement that a transistor has crossed its threshold.

Implementing delays may be expensive, and it would be advantageous to reduce their number. Our design allows the following optimisations:

- Since the PMOS and NMOS transistors are never On at the same time, the PMIN_TIMER and NMIN_TIMER can be merged (provided their delays are the same).



(a) Specification.

(b) ME-based solution.

(c) Gate-level implementation.

Figure 5: WAIT element.

- The TOKEN_TIMER can be shared between the stages of the token ring as only one of them holds the token (note that the HL mode does not affect token propagation).

III. Formal specification and implementation

In this section we focus on the formal specification and logic synthesis of a few most interesting components of the multiphase buck controller.

A. Signal transition graphs

An important subclass of asynchronous circuits, called *Speed-Independent* (SI) circuits, follows the classical Muller's approach [11] and regards each gate as an atomic evaluator of a Boolean function, with a delay element associated with its output. In the SI framework this delay is unbounded, i.e. the circuit must work correctly regardless of its gates' delays, and the wires are assumed to have negligible delays. Alternatively, one can regard wire forks as isochronic and add wire delays to the corresponding gate delays (*Quasi-Delay Insensitive* (QDI) circuit class [12]).

Signal Transition Graphs (STGs) [13] are a formalism for specifying such circuits. They are Petri nets [14] in which transitions are labelled with the rising and falling edges of circuit signals. The details of logic synthesis from STGs based on state graphs can be found in [15].

Graphically, the places are represented as circles, transitions as textual labels, consuming and producing arcs are shown by arrows, and tokens are depicted by dots. For simplicity, the places with one incoming and one outgoing arc are often hidden, allowing arcs (with implicit places) between pairs of transitions. We use thick arcs to denote *concurrency reduction* – such arcs are not necessary, but simplify the implementation.

B. Sanitising sensor readings

UV and HL conditions are handled by all stages of the token ring and thus signals uv and hl are not persistent – they may be reset by a competing stage. Therefore these inputs should be *sanitised* before passing them to the HLH and UVH modules. This is done by the *WAIT element* whose specification and gate-level implementation are shown in Figure 5. The WAIT element's interface comprises:

- Input sig, which may be non-persistent. As can be seen from the STG in Figure 5a, sig may go high and low without any restrictions.
- Input ctrl, whose rising transition brings the WAIT element into the *waiting mode* and falling transition returns it back to the *dormant mode*.
- Output san, which is insensitive to sig in the dormant mode, and goes high as soon as sig+ is detected in the waiting mode, and is latched: unlike input sig, output san is persistent – it is not reset until ctrl- indicates the receipt of san+. Pair (ctrl, san) thus forms a 'clean' asynchronous handshake driven by the 'dirty' sig input.

Note that there is a read-consume conflict between transitions sig- and san+. We resolve this conflict using a *mutual exclusion* (ME) *element* [16] with one inverted input, see Figure 5b. The intuition is that when ctrl+ arrives, it cannot propagate to the output san until the ME element is released by sig+. Once san+ is asserted, the ME element becomes insensitive to changes of sig until ctrl is released. There is a hazard on output g1 as sig- may disable it before the metastability inside the ME element is resolved. However, this output is unused and can be removed together with the corresponding part of the *metastability filter* [16], see Figure 5c.

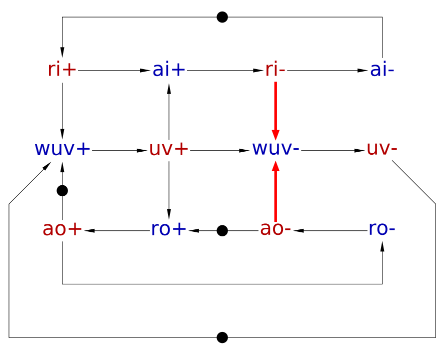
C. Charging the buck

The control of the charging is done by three modules: UVH, ZCH and OCH. Their STGs are shown in Figures 6a, 7a, and 8a.

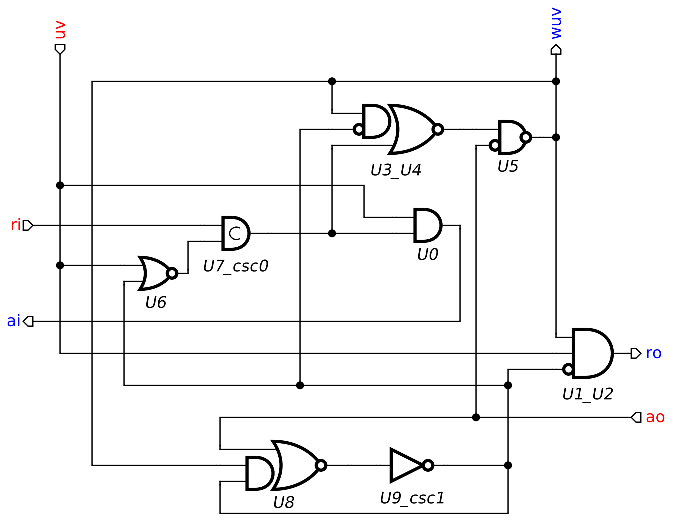
UVH module waits for an activation request ri+ from the MERGE element and then, by means of wuv+, triggers UV_WAIT into monitoring the UV condition. When uv+ arrives, an early acknowledgement ai+ to the MERGE element and a request ro+ to the ZCH module are sent concurrently, thus decoupling token propagation from buck charging.

ZCH module handles three scenarios, which were informally specified by the phase diagram in Figure 1b: (i) ZC does not happen during a cycle of charging, (ii) ZC arrives after UV and is ignored, and (iii) ZC is detected before UV and initiates the charging. The former two scenarios are similar and are combined into one branch of a choice. In the later scenario, upon arrival of zc+ a request ro+ is sent to OCH to switch the NMOS transistor Off; however, switching the PMOS transistor On is delayed until arrival of a request ri+ from UVH. Note that *dummy* transitions no_zc and join are used for convenience. A dummy does not model any signal of the circuit – the states before and after its firing correspond to the same circuit state.

OCH module controls the PMOS and NMOS transistors (via PMIN_DC and NMIN_DC modules) and handles the OC condition. Both phases of the input handshake from ZCH module do useful work: ri+ initiates switching NMOS transistor Off, while ri- triggers PMOS transistor On. The opposite switching of transistors (PMOS Off and NMOS On) is triggered by the OC condition. Acknowledgements an and ap are used to monitor the state of buck transistors and interleave their On



(a) Specification.



(b) Implementation.

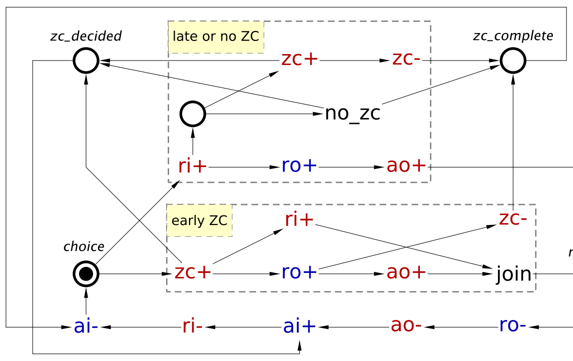
Figure 6: UVH module.

states. Note that initially both NMOS and PMOS are OFF, therefore the first instance of ri+ is immediately acknowledged.

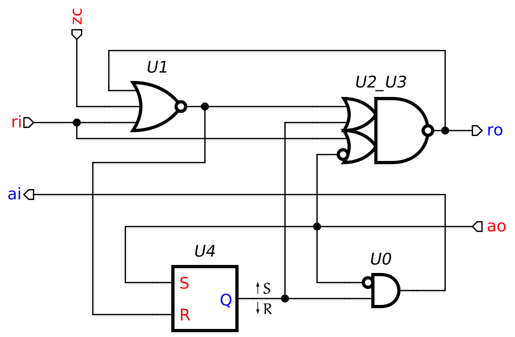
Results of logic synthesis and technology mapping by PETRIFY [15] (into a library of available gates based on petrify.lib) are shown in Figures 6b, 7b, and 8b.

IV. VERIFICATION

STG specifications of all controller modules were developed and verified using WORKCRAFT framework [17], [18]. We verified that all STGs are consistent, deadlock-free, and output-persistent. For checking specific buck converter properties, such as the absence of a short circuit in PMOS/NMOS transistors and possibility for sharing a timer for both PMIN and NMIN delays (provided they are the same), an STG specification of the whole stage was needed. This STG was obtained by parallel composition [7] of the STGs for individual modules. Note that signals shared between the modules (connected output and input pins) had to share a common name. We used the drivers' names – name of the controller ports or output pins in MODULE_signal format. These names are shown as grey labels next to the wires in Figure 4. The obtained STG is quite large for illustration or logic synthesis, but still serves the purposes of verification. Using this STG we successfully verified that:



(a) Specification.



(b) Implementation.

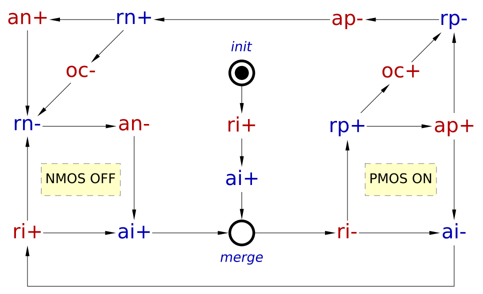
Figure 7: ZCH module.

- There is no short circuit in the buck, as no state is reachable where PMOS and NMOS transistors are both ON, i.e. the following state predicate is never satisfied: (gp=1 ∨ gp_ack=1) ∧ (gn=1 ∨ gn_ack=1).
- The same timer can be used for PMIN and NMIN delays, as there is no state where PMIN_TIMER and NMIN_TIMER are both in use, i.e. the following state predicate is never satisfied: (PDC_rd=1 ∨ PT_a=1) ∧ (NDC_rd=1 ∨ NT_a=1).

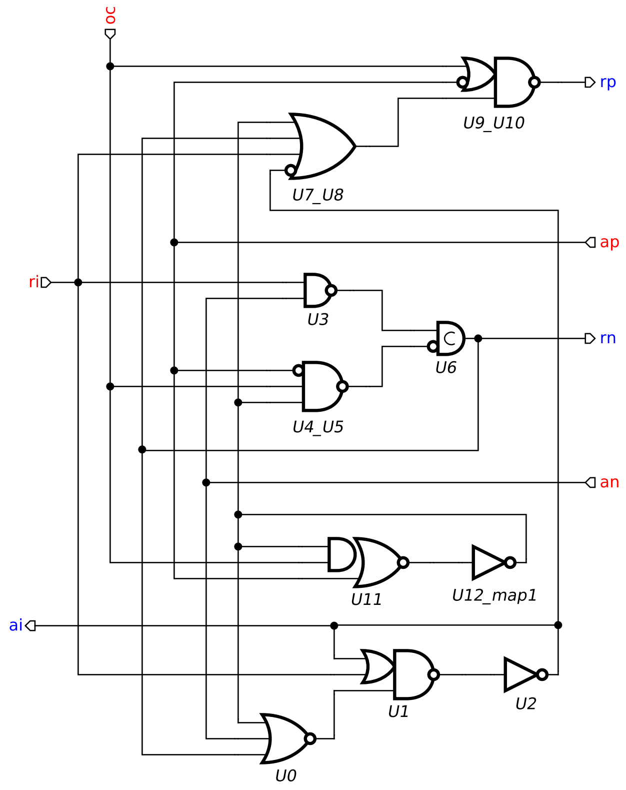
We also checked the possibility of sharing the same instance of timer for token delay in all the stages of a lazy token ring. For this we built an STG model of a buck controller with 2 stages. The internal signals of Stage 1 and Stage 2 were prefixed with S1 and S2 respectively while the interface signals were named according to the wire labels in Figure 3. The obtained STG was used to verify the following property:

- The same timer can be shared between the stages as no state is reachable where token timers of both stages are in use, i.e. the following state predicate cannot be satisfied: (S1_TC_rd =1 ∨ S1_TT_a=1) ∧ (S2_TC_rd=1 ∨ S2_TT_a=1).

Similar checks were made for 3-phase and 4-phase controllers, showing that no more that one token timer is in use. Furthermore, the following argument can be used to generalise this result to arbitrary number of stages, as a special case of parametrised verification approach [19], [20]. Consider Figure 3: ignoring all the interface signals except get and pass, each stage of the ring can be abstractly represented by a buffer. The composition of two sequential buffers is just



(a) Specification.



(b) Implementation.

Figure 8: OCH module.

a buffer, and so two stages behave like a single stage from the point of view of the ring. Consider the STG in Figure 9 obtained as follows:

- Two stages of the ring are composed.
- The result is composed with their environment: the init module (an inverter, as the reset can be ignored) and the other ring stages represented by single buffer, as explained above; this buffer can be further merged into the inverter corresponding to the init module.
- In the resulting STG all the signals except the handshakes with the token timers are hidden.
- The STG is resynthesised with PETRIFY.

The final STG is simply two sequential timer handshakes. Each of these handshakes can be compressed into a single "timer in use" transition. If two stages share a timer, their two "timer in use" transitions can be further compressed into a single such transition in a conservative way. Hence, the usage of a shared timer by two stages can be conservatively approximated by that of a single stage, and the argument holds for N stages by induction.

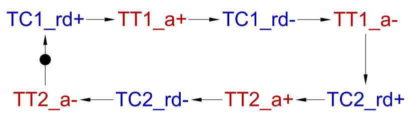


Figure 9: Timer interface of two ring stages.

All the gate-level implementations were also verified to be deadlock-free, hazard-free and conformant to their STG specification. We used WORKCRAFT [17], [18] and cross-checked the verification results with VERSIFY tool [21]. One inconsistency was discovered when VERSIFY claimed there is a circuit-environment synchronisation error in the ZCH module. However, this was due to incorrect interpretation of dummy transitions (and non-determinism in general) by VERSIFY. (This was confirmed by a simple testcase of an inverter and its STG specification where a dummy was inserted before one of the output transitions.)

V. SIMULATION AND ANALYSIS

A. Simulation setup

The focus of this work is on replacing the manual *ad hoc* design of asynchronous power management controllers with logic synthesis from formal specifications. While this approach enables formal verification of the obtained circuits, one still needs to ensure that all the power management scenarios were captured and correctly interpreted by the formal specification.

A traditional approach to checking the functional correctness of a power regulator is by exhaustive simulation that is split in two stages. In the first stage a coverage-driven constrained-random digital regression is used to ensure that the circuit has been stimulated for all possible scenarios and all its responses are as expected. A gate-level VERILOG netlist is used for the digital controller while the analogue part of the system is modelled behaviourally with SYSTEM VERILOG (to speed-up the simulation and simplify the exploration of corner cases). A typical digital regression would perform tens of millions of charging cycles for each of its runs targeting a 100% functional and 100% code coverage. In order to complete overnight, such a regression needs to run on a compute farm; the regression would be run every night for many weeks before tapeout. During the second stage the performance and functionality of the whole system are validated. This is based on SPICE and mixed-signal simulation techniques that require many man-months of effort and extensive use of a compute farm to complete.

We reused an existing industry-standard test environment of Dialog Semiconductor, a leading power electronics company. The correct operation of the presented multiphase buck controller was checked by a digital regression that was previously employed in one of the company's commercial products. System integration and its validation is a work in progress.

Modules	Circuit size (µm², ME)	Latency (gates)	
		Worst	Critical
HLH	***80***	2	1
TC	***240***	6	6
basic MERGE	352	3+ME	3+ME
opportunistic MERGE	***256***	2+ME	2+ME(no metastability)
UVH	***272***	3	1
ZCH	***184***	3	2
OCH	352	5	2
PMIN_DC, NMIN_DC	144	3	1
OCH_and_DCs	***584***	5	2
HL_WAIT, UV_WAIT	***72***	1+ME	1+ME(no metastability)
Stage size	***1760***		

Table I: Implementation statistics.

B. Analysis of the results

Some statistics related to multiphase buck controller modules are summarised in Table I. The size of gate-level implementation is reported for each module. The circuit worst case latency is measured in the total number of gates and ME elements in a longest path from an input stimuli to an output reaction. As these delays are often outside the critical path, the latency in circuit reaction to critical signals is reported separately. The circuit size of the modules that are selected for implementation of a single token ring stage (from several alternatives) are emphasised. Note that there are ME elements in the critical path, but metastability never arises there in the chosen implementation (it can occur in non-critical paths though). The computation time for synthesis and verification was negligible (<1sec in all cases).

Note that if a combined latency of several modules in the critical path is too high, then it can sometimes be reduced by joint resynthesis of several modules. For example, a reaction to `oc+` is 3 gates (critical path of OCH and PMIN_DC modules). If OCH is jointly resynthesised with the delay control modules PMIN_DC and NMIN_DC, the latency is reduced to 2 gates (OCH_and_DSs line in the table). We will explore this in our future research.

VI. Conclusion

This work addresses a challenging task of interfacing asynchronous digital circuits with analogue electronics. We designed, implemented and verified a speed-independent multiphase buck controller. It uses a lazy token ring architecture to overlap the charging cycles of multiple phases and balance utilisation of the phases. The controller also handles high-load mode by activating all phases simultaneous. The characteristic features of the controller are reliable handling of sensor readings and low-latency reaction to the changes in power demand. We formally verified the essential correctness properties of the controller, and validated its operation in conventional industry environment. The use of STG specifications, synthesis and verification has been beneficial both in providing a formal framework and guidance during the development process as well as increasing the confidence in the final design. The role of compositional approach in this design exercise is important for many reasons, such as maintainability and clarity.

Future work includes extending the proposed controller with design-for-test features, its post-layout mixed-signal simulation in combination with Spice models of the analogue part, and fabrication of a first power regulator with a speed-independent controller.

Acknowledgements

The authors are grateful to the anonymous reviewers for all the constructive feedback and inspiring comments. This research was supported by EPSRC grants EP/L025507/1 *A4A: Asynchronous design for Analogue electronics* and EP/K001698/1 *UNCOVER: UNderstanding COmplex system eVolution through structurEd behaviouRs*.

References

[1] A. Pressman, K. Billings, T. Morey: *“Switching power supply design”*, 3rd edition, McGraw-Hill, 2009.

[2] J. Audy: *“Navigating the path to a successful IC switching regulator design”*, Tutorial at IEEE International Solid-State Circuits Conference (ISSCC), 2008.

[3] T. Towers: *“Practical design problems in transistor DC/DC converters and DC/AC inverters”*, Proc. IEE, vol. 106(18), pp. 1373–1383 1959.

[4] S. Unger: *“Asynchronous Sequential Switching Circuit”*, Wiley-Interscience, 1969.

[5] D.Sokolov, A.Mokhov, A.Yakovlev, D.Lloyd: *“Towards asynchronous power management”*, Proc. Faible Tension Faible Consommation (FTFC), 2014.

[6] D. Lloyd, R. Illman: *“Scan insertion and ATPG for C-gate based asynchronous designs”*, Synopsys User Group (SNUG), 2014.

[7] W. Vogler, R. Wollowski; *“Decomposition in asynchronous circuit design”*, In J. Cortadella et al. (editors), *“Concurrency and Hardware Design”*, Lecture Notes in Computer Science 2549, pp. 152–190, 2002.

[8] M. Greenstreet: "Real-time merging", Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), pp. 186–198, 1999.

[9] R. Janicki, M. Koutny: *"On causality semantics of nets with priorities"*, Fundamenta Informaticae, v. 38(3), pp. 223–255, 1999

[10] A. Mokhov, V. Khomenko, D. Sokolov, A. Yakovlev: *“Opportunistic merge element”*, Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2015.

[11] D. Muller, W. Bartky: *“A theory of asynchronous circuits”*, Proc. International Symposium of the Theory of Switching, pp. 204–243, 1959.

[12] A. Martin: *“Compiling communicating processes into delay-insensitive VLSI circuits”*, Distributed Computing, vol. 1(4), pp. 226–234, 1986.

[13] T.-A. Chu: *“Synthesis of self-timed VLSI circuits from graph-theoretic specifications”*, PhD thesis, Massachusetts Institute of Technology, 1987.

[14] C. Petri: *“Kommunikation mit automaten (Communicating with automata)”*, PhD Thesis, University of Bonn, 1962.

[15] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev: *“Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers”*, IEICE Transactions on Information and Systems, vol. E80-D(3), pp. 315–325, 1997.

[16] D. Kinniment: *“Synchronization and Arbitration in Digital Systems”*, Wiley Publishing, 2008.

[17] I. Poliakov, A. Mokhov, A. Rafiev, D. Sokolov, A. Yakovlev: *“Automated verification of asynchronous circuits using circuit Petri nets”*, Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), pp. 161–170, 2008.

[18] Workcraft homepage: http://workcraft.org/.

[19] K. McMillan: *“Verification of infinite state systems by compositional model checking”*, Proc. Correct Hardware Design and Verification Methods (CHARME), pp. 219–234, 1999.

[20] C.-T. Chou , P. K. Mannava , S. Park: *“A simple method for parameterized verification of cache coherence protocols”*, Proc. Formal Methods in Computer Aided Design (FMCAD), pp. 382–398, 2004.

[21] O. Roig: *"Formal verification and testing of asynchronous circuits"*, PhD Thesis, Universitat Politècnica de Catalunya, 1997.