Loadable Kessels Counter

Oyinkuro Benafa, Danil Sokolov, Alex Yakovlev
School of Engineering, Newcastle University, UK
Email: {o.benafa, danil.sokolov, alex.yakovlev}@newcastle.ac.uk

Abstract—We present the decomposition and implementation of a loadable self-timed counter that can perform seamless modulo loading and counting operation. The challenges in designing a loadable self-timed counter stem from the need to dynamically reconfigure operations between the counter components to arrive at the desired count modulo. The counter was decomposed into a combination of parallel and interacting computing cells as presented by Kessels. The binary equivalent for the count modulo \( n \) determined the operation of each cell in relation to its significance. Specification and verification of the counter are by formal asynchronous design methods employing Petri Nets. A 5-bits loadable counter is implemented and fabricated in 350nm CMOS process. Average power consumed at 3.3V for count 31 is in the range 89\( \mu \)W to 157\( \mu \)W. The response time of the counter after a load request is received ranges from 28.80ns to 32.71ns. Such a counter is robust and presents a practical application in timing systems like the Digital Pulse Width Modulator (DPWM) used in a DC-DC converter with fine tune control. For example, the DPWM design can sustain a variation of Vdd in the range of 3.3V to 1.8V maintaining its duty-cycle with a margin of error in the range of 1\% to 7\%.

I. INTRODUCTION

Digital counters are widely employed in digital and mixed-signal systems like the phase locked loop (PLL), timers and in circuitry that requires frequency and pulse control. In these applications, the counter functions as a programmable modulo\( n \) counter.

In a typical synchronous modulo\( n \) counter, both the counting and the modulo detection units are triggered by a clock signal. The coarse nature of the clock can introduce quantisation in the modulo\( n \) counter operation, which may affect its application in the overall systems. For example, in a synchronous modulo\( n \) counter based Digital Pulse-Width Modulator (DPWM) employed in DC-DC converters, quantisation from the clock can cause the converter to go into a state called Limit Cycle Oscillation (LCO) [1], [2]. Increasing the resolution of the DPWM is one method used to control or eliminate LCO. However, this may involve among others, tuning the clock frequency. While this approach may be sufficient, two factors to be considered are the time margins of the gates [3] and part of power consumption that is affected by the operating frequency [4].

In asynchronous systems, the absence of a global clock means the system will operate in a fine-grained mode, thus minimising quantisation while maintaining robustness [5] and low dynamic power consumption. Therefore, it is vital that a practical approach to realising a self-timed loadable modulo\( n \) counter is explored. This requires efficient decomposition and specification of the loadable modulo\( n \) counter.

Decompositions of asynchronous modulo\( n \) counters have been presented in [6]–[11]. In [6]–[8], a delay-insensitive modulo\( n \) counter was realised by decomposing it to a combination of basic asynchronous elements like the toggle, merge and join circuits. In [7], [8], a self-timed counter is described that consists of toggles cascaded in series and a completion detection circuit in a closed system. The counter described in [6] is triggered by an event from the environment which is acknowledged after the counter has changed its state in response to the trigger by an output event in one of two output channels. The other output channel is used to indicate end of count operation. The counters presented in [6]–[8] all have fixed structures and therefore can only perform a statically predefined count modulo sequence.

In [9], [10], [12], the counter was decomposed into a combination of interacting counter cells. In [9] a counter cell is classified as either even or odd. The classification depends on the value and position of its corresponding binary digit for \( n \). In [10], the counter cells were further decomposed and shown to consist of both even and odd operations which can be reconfigured to change the count modulo \( n \).

The modulo\( n \) counter decompositions presented in [9], [10], [12] applied Horner’s method. Counter decomposition using Horner’s method was first described by Kessesls in [13]. This paper adopts the work presented in [10]. The main contributions of this paper are:

- Definition of the different conditions and range of possible even and odd operations in a counter cell. This approach led to a reconfigurable counter operation.
- Specification of the counter cells operations using formal models: Labeled Petri Nets (LPN) [14] at the high level and Signal Transition Graphs (STG) [15] at the low level.
- Design of a control block which consists of interacting control cells, each in a one-to-one configuration relationship with the counter cells.
- Specifications of load channel encodings between interacting control cell parts and configuration channel encodings between each related control and counter cell parts.
- Implementation of the specified counter in 350nm CMOS technology. This involved technology mapping of synthesised gates to AMS standard library.

II. MODULO\( n \) COUNTER OVERVIEW

Fig. 1 shows the block diagram of the loadable modulo\( n \) counter denoted by \( C_n \). A load request is sent to the counter from the environment on input port \( W_i \), and this causes the
counter to load $n$ after which a load acknowledgement is sent to the environment from the counter on output port $Wia$. Loading $n$ configures the internal operation of the counter to produce $n$ pulses on $ar$ after which an end of count pulse is produced on $br$.

The relationship between the loaded count modulo $n$ and the operation on output oports $ar$ and $br$ of counter $C_n$ is described by the regular expression I, where $n \geq 1$. The term $!$ denotes an output port.

$$C_n = (ar^{n+} br^!)^*$$ (1)

Expression 1 does not provide sufficient information as to the order of internal events on each output channel of $C_n$.

The aim here is to decompose the counter by expressing the count modulo $n$ such that the counting operation on $ar$ channel is distributed in a network of adjacent interacting cells. This approach, helps us to arrive at a counter with bounded response time on $ar$, and $br$ outputs irrespective of the count modulo $n$. Response time is viewed from the perspective of causality, with events on $ar$ caused by loading $n$ while an event on $br$ caused by $n$ events on $ar$.

III. DECOMPOSITION OF $n$ USING HORNER’S METHOD

Horner’s method rewrites $n$, arriving at a suitable decomposition as shown in equation 2 while preserving its value. This decomposition approach was first presented by Kessels [13].

$$n = ((..((0 \times 2 + d_{N-1})2 + d_{N-2})2 + ..)2 + d_1)2 + d_0)$$ (2)

Horner’s method also expresses $n$ as an unsigned binary number in the range $d_{N-1}$ to $d_0$. Here $d_{N-1}$ is the Most Significant Bit (MSB) and $d_0$ is the Least Significant Bit (LSB).

A. Cells and Cell Parts of decomposed $n$

In (2), $n$ is shown to consist of interacting blocks which are in the form $(\times 2 + d_i)$. Each block is referred to as a counter cell and is denoted by $ci$.

Each cell can further be decomposed into two parts, a left part $CL$ which does the $(\times 2)$ operation and a right part $CR$ which does the $(+d_i)$ operation.

The number of bits $N$ representing $n$ is equal to the number of counter cells in the decomposed $n$ and it is given by $[\log_2 n] + 1$.

The notations $ci$, its subparts and $d_i$ denote the $i$th counter cell and $i$th bit respectively. This method will be used throughout this paper to name cells, cell parts and port names.

TABLE I

<table>
<thead>
<tr>
<th>$N$</th>
<th>$c_2$</th>
<th>$CR_2$</th>
<th>$CL_1$</th>
<th>$CR_1$</th>
<th>$CL_0$</th>
<th>$CR_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>-</td>
<td>-</td>
<td>0 +1</td>
<td>x2</td>
<td>+1</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>-</td>
<td>0 +1</td>
<td>x2</td>
<td>P</td>
<td>x2 +1</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0 +1</td>
<td>x2</td>
<td>P</td>
<td>x2 +1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>0 +1</td>
<td>x2 +1</td>
<td>P</td>
<td>x2 +1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>0 +1</td>
<td>x2 +1</td>
<td>P</td>
<td>x2 +1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>0 +1</td>
<td>x2 +1</td>
<td>P</td>
<td>x2 +1</td>
<td></td>
</tr>
</tbody>
</table>

B. Operations of Cell Parts

Consider a counter implemented in silicon with $N_T$ total number of counter cells, the range of values of $n$ it can accept is given by $n \in \{0, 2^{NT} - 1\}$.

Table I illustrates a counter with $N_T = 3$ cells, with the range of numbers for $n$ decomposed into cells with left and right parts. The set of counter cells directly mapped to a valid bit of $n$ are the active cells involved in the computation of $n$. The left and right part operations of an inactive cell are shown as spacers "-" since they do not add to the count sequence.

The operation of $CL$ for the most significant active cell is always a zero operation $(0 \times 2)$. For all other active cells, the operation of $CL$ is always $(\times 2)$. To realise a loadable counter, $CL$ operations of each cell must include a spacer (-) operation, a $(\times 2)$ and zero operations. The choice of operation depends on the binary sequence of $n$. The only exception to this is $CL_{N_T-1}$ which is the left part of the Most Significant Cell (MSC) $c_{N_T-1}$ in the implemented counter. From Table I, this cell part operates as either a spacer (-) when its corresponding bit is a '0' or a Zero operation $(0 \times 2)$ when its corresponding bit is a '1'. The range of possible operations of $CL$ is referred to as EVEN Operations.

The operation of $CR$ for an active cell $c_i$ adds the corresponding bit value $d_i$ once to the count sequence. When $d_i = 0$, nothing is added to the count sequence. In this case, $CR_i$ operates as a channel for passing counts received from $CL_i$ to $CL_{i-1}$. This operation is referred to as a Pass (P) operation as shown in Table I. To realise a loadable counter, $CR$ operations of each cell must include a spacer (-), a +1

Fig. 2. Diagram illustrates concurrent operation in counter cells.
and a \( P \). The only exception to this is \( CR_{N_{-1}} \) of the Most Significant Cell (MSC) in the counter which can operate as either a spacer (-) when its corresponding bit is a '0' or a +1 when its corresponding bit is a '1'. The range of possible operations of \( CR \) is referred to as **ODD Operations**.

### C. Concurrency in Decomposed \( C_n \)

The decomposition of \( n \) in (2) shows independent interactions between \( CR_i \) and \( CL_{i-1} \). This interaction occurs concurrently across all cells. The result of each interaction is communicated to an adjacent cell part \( CR_{i-1} \).

Consider the decomposition example for \( n = 5 \), decomposed as \( \frac{(0 \times 2 + 1)2 + 02 + 1}{2} \). Concurrent operation between cells and cell parts in the counter is illustrated in Fig. 2.

In Fig. 2, dashed arrows between cell parts indicate data flow direction. The new state of each counter cell part after passing to a cell part that has completed its computation for \( n \) communicated within or without cell parts. It can only be passed to a cell part that has completed its computation for \( n \) on its \( ar \) channel. Hence, it appears on the \( br \) output of the counter after \( n \) counts on its \( ar \) output.

The numbers shown in the count column are the counts outputted from the interaction of the counter cells. The blank spaces between outputted counts do not model the delay in the system.

### IV. LOADABLE MODULO—\( n \) COUNTER

In Section III-B, we described the modulo—\( n \) counter and the different operations each cell part can perform depending on the binary sequence of \( n \). Two main functions can be intuitively identified in each counter cell. They are counting, and configuration functions. The counting function of a cell part is the even and odd operations earlier identified. The configuration function can be described as the part of the counter that determines from the binary sequence of \( n \) the active cell parts and the correct even or odd operation for each active cell part. We call the combination of cells dedicated to performing the counting function the counter block and those that perform configuration function we call the control block.

Fig. 3 shows the block diagram of the loadable modulo—\( n \) counter. It contains a control and a counter block in a cell-to-cell interaction. The control and counter blocks are each made up of an equal number of cells. The wrapper shown in the control block provides an interface between the environment and the control cells on input \( Wi \) through which it receives a load request and on output \( Wia \) through which it sends a load acknowledgement.

In Fig. 3, a cell \( C_i \) is shown to consist of a control cell \( c'_i \) and a counter cell \( c_i \). The cells \( c'_i \) and \( c_i \) each consists of a left part \( CL'_i \) and \( CL_i \) and a right part \( CR'_i \) and \( CR_i \) respectively.

Fig. 3 shows a high-level interaction between the cell parts, using signal names in each cell part and an arrow to indicate origin and destination of an action between each cell part. This form is suitable for high-level specification of the cell parts using LPN, in which signal names are used to represent events. In asynchronous systems, a valid communication involves a handshake between computing units. To specify \( CL', CR', CL \) and \( CR \) using STG, the signal names are first refined to request and acknowledge signals pairs, as shown in Fig. 4, and then the operation of each part is specified using a 4-phase handshake protocol [16].
TABLE II
RELATIONSHIP BETWEEN REFINED SIGNAL NAMES OF FIG. 4.

<table>
<thead>
<tr>
<th>Cell Parts</th>
<th>Req (Out → in)</th>
<th>Ack (in → out)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL_i : CR_i</td>
<td>Lo1, Lo1' → Lo0, Lo1</td>
<td>Lo0' ← Lia</td>
</tr>
<tr>
<td>CR_i : CR_{i-1}</td>
<td>Lo0, Lo1 → Lo0, Lo1</td>
<td>Lo0' ← Lia</td>
</tr>
<tr>
<td>CL_i : CR_i</td>
<td>ar, br → cr, dr</td>
<td>aa, ba ← ca, da</td>
</tr>
<tr>
<td>CR_i : CR_{i-1}</td>
<td>ar, br → cr, dr</td>
<td>aa, ba ← ca, da</td>
</tr>
</tbody>
</table>

V. SPECIFICATION OF COUNTER PARTS

In this section, the left and right parts of the counter cells are specified. Two levels of specifications are used here. These are high level using LPNs and low level using STGs.

Specifying each cell part operation using LPN, allows easy modelling and verification of the counter operation by unfolding actions using signal names to describe interactions between cell parts in a flow diagram of cause and effects.

The following terms or indicators used previously and in the rest of the paper are described below:

- **Channel**: Input and output transitions cr', cr and ar'. ar occur on ar channel in which the count modulo n is computed, while input and output transitions dr', dr and br', br occur on br channel in which the zero operation is computed.

- **Choice**: Refers to two input signals with the same pre-set Place (P). If P contains a token, then an input transition on one signal disables the other.

- **Names and colouring**: In all the diagrams the output, input and internal signals are coloured blue, red and green, respectively. Signals of a left cell part (control or counter) are indicated by an apostrophe.

- **MSC and LSC**: MSC refers to the most significant cell in an implemented counter, denoted as c'_N−1 and c_{N−1} while LSC refers to c'_0 and c_0 when used in the context of control and counter cells respectively.

A. COUNTER CELL PARTS

1) High Level Specification (LPN): In Fig. 2, each cell part performed two non-mixing operations. They are the computation for n operation and the zero operation. In this section, our specification of each cell part accounted for both operations on ar and br channels respectively. Each channel in a counter cell part has an input port by which it receives events from an adjacent cell part and an output port by which effects of received events are sent to an adjacent cell part. The ar channel has inputs cr', cr and outputs ar', ar for parts CL and CR respectively. The br channel has inputs dr', dr and outputs br', br for parts CL and CR respectively.

On the left side of Fig. 5, two LPNs for CL are shown. The first is a zero operation specified as an enabled br' output signal which can fire independently. The second is a ×2 operation, with a choice for an input event on cr' and dr'. An input event on cr' results in two output events on ar' while an input event on dr' activates a pass channel for zero operation resulting to an output event on br'.

On the right side of Fig. 5, two LPNs are shown for CR. The first LPN is an enabled choice of pass operations. An input event on cr or dr will result in an output event on ar or br respectively. The second LPN is a +1 operation in which an event on ar output is enabled and can fire independently. This action enables a choice of pass operation on cr and dr. An input event on cr results in an output event on ar and re-enables the choice of pass operations, while an input event on dr results in an output event on br after which +1 operation is re-enabled.

2) Modeling of Decomposed Count Modulo 5 by Unfolding: Fig. 6 is a decomposed modulo 5 counter showing its constituent cell parts, their operations and LPNs. The interaction between signal names of cell parts indicates the destination and origin of an output event and input event in the LPNs respectively. In the unfolding of events presented in Fig. 7, this relationship is employed.
in the following steps:

Transition events are grouped and shown in steps indicated by correctness of the decomposition and specification approach. Transitions will fire in minimal step bundle. This assumption on Fig. 7 could not be shown to occur until a valid input transition enabling when the pre-conditions of both signals are satisfied. For input connection (interaction between signals of two cell parts is given the output in Fig. 7 means an output event on ar

The rest of the unfolding followed this pattern of token flow as a result of interaction between enabled output and input signals of two interacting cell parts. After five ar transitions, a br transition occurred in line 13.

After the first ar transition event in step 1, the next ar transition event occurred after cr received an input event from ar'. From the graph of Fig. 7, ar' transition did not occur until step 4. This is because CR operates as a Pass, requiring an input event on cr to produce an output event on ar. A chain of causes and effects beginning from ar transition in step 1 unfolded until step 3 before ar transition occurred in step 4, thus the next transition on ar is shown in step 5.

The br transition in step 2 is a zero operation which is gradually passed through cell parts from the MSC to the LSC on the br channel. The zero operations can only occur in the br channel of a counter cell part after it as completed its computation for n on the ar channel. This occurred in steps 2, 5, 8, 11, 12 and on the counter output (br) in step 13.

In steps 2, 5, 8, 11, 12 and 13 each highlighted Place indicates that cell part is in the initial state (token in initial place and the cell part on its right has completed its computation for n). This return to initial state action began from the MSB cell left part, and gradually moved towards the LSB cell after a cell part performs a zero operation.
After completion of the first count sequence (output on $b_{h0}$), a token is placed in $P0$ as shown in line 13. This action enabled $ar_0$ because the right cell operation of the LSB is a (+1) thus, the second count sequence is started and shown in the output on line 14. For an even count modulo, the first $ar_0$ transition would not appear in step 1, because the LSC right part does a pass operation and this requires an input transition on $cr_0$. After the first count sequence, a step is skipped before the start of the next count sequence for the same reason.

The delay noticed after the first transition on $ar_0$ in the first count sequence is eliminated in subsequent count sequences because as cell parts return to the initial state, transitions for the next count sequence are enabled and can even fire before the end of the active count sequence. This is shown in lines 9, 10, 12 and 13 where transitions on $ar_2, br'_2, ar'_1$ and $ar_1$ occurred respectively.

3) Low Level Specification (STG): The STGs of Fig. 9a and Fig. 8a show even operations for $CL$, specified based on the cell position. Namely, two main types of cells are singled out: one is for the MSC and the other for all other cells. This also applies to Fig. 9c and Fig. 8b for odd operations of $CR$.

For example, in Fig. 8a, input transitions $cr'_i+ and n_{11}+$ activate the $\times 2$ operation on $ar$ channel of $CL$. This results in two $ar'_i+$ events. Similarly, input transitions $dr'_i+$ and $n_{10}+$ enable the zero operation on $br$ channel of $CL$ and this causes $br'_i+$ transition. A combination of input transitions on $dr'_i+$ or $cr'_i+$ and $n_{11}+$ opens a pass operation on $br$ or $ar$ channels respectively. Details of configuration command encoded from each control cell part are shown in the next section.

VI. CONTROL BLOCK SPECIFICATION

The wrapper and control block cell parts were specified in a top-down approach using high-level and low-level specifications. However, due to lack of space, this section only shows the wrapper STG and each control cell part configuration command to a counter cell part, conditions and encodings.

A. The Wrapper STG

The wrapper and environment interfaces with the environment in a four-phase communication protocol by relaying a load request from the environment on input $Wi$ to $CL_{i-1}$ on its $Lo$ output. This load request is propagated through the control cell parts along the load channel to $CR_{i}$ which interacts with the wrapper on input $Li$ to relay a load acknowledgement to the environment, see Fig. 3. The STG of the wrapper is shown in Fig. 10a.

After a four-phase handshake communication between wrapper and environment, a new/previous count modulo $n$ can be set and a new load request issued, even when the
previous count sequence is still active in the counter block. The new load request and count configuration effectively starts from the MSC left cell part and is propagated towards the LSC right part as each cell part completes its computation for the previously loaded $n$. The specification of four phase communication protocol between interacting cell parts ensures hazard free transition between successive count sequence for each new load request.

B. Control Block Left and Right Cell Configuration Command Specifications

For cell $N_T - 1$

Left Part (Refinement: Single-rail $n1 = n10$)

$$n1 = \begin{cases} - & \text{if } d_i = 0 : \ (n1 = "00") \\ 0 & \text{if } d_i = 1 : \ (n1 = "11") \end{cases}$$

Right Part (Refinement: Single-rail $n2 = n21$)

$$n2 = \begin{cases} - & \text{if } d_i = 0 : \ (n2 = "00") \\ +1 & \text{if } d_i = 1 : \ (n2 = "11") \end{cases}$$

For $0 \leq i < N_T - 1$

Left Part Control (Refinement: Dual-rail $n1 = n10, n11$)

$$n1 = \begin{cases} - & \text{if } \forall j \geq i : d_j = 0 : \ (n1 = "00") \\ 0 & \text{if } \forall j > i : d_j = 0 \land d_i = 1 : \ (n1 = "10") \times 2 & \text{if } \exists j > i : d_j = 1 : \ (n1 = "01") \end{cases}$$

Right Part (Refinement: Dual-rail $n2 = n20, n21$)

$$n2 = \begin{cases} - & \text{if } \forall j \geq i : d_j = 0 : \ (n2 = "00") \\ P & \text{if } \exists j > i : d_j = 1 \land d_i = 0 : \ (n2 = "10") \\ +1 & \text{if } d_i = 1 : \ (n2 = "01") \end{cases}$$

C. Control Block Left and Right Cell Load Request Specifications

Load request between control cell parts is encoded in single-rail. The conditions and encodings for "LoLo'Lo1" and "Lo0Lo1" are:

- "00": if Load Req = 0
- "10": if $\forall j \geq i : d_j = 0$ and Load Req = 1
- "01": if $\exists j \geq i : d_j = 1$ and Load Req = 1

The load request between a control cell part and the wrapper is encoded in dual-rail, as shown by the STG in Fig. 10a.

VII. IMPLEMENTATION AND MEASUREMENT RESULTS

The circuits of each cell part were synthesized from their STGs using the WORKCRAFT toolset (https://workcraft.org/). Figs. 9b and 10b show the synthesized circuits for $CL_{N_T-1}$ and the wrapper respectively. The circuits were modified with extra logic gates to provide controlled reset inputs.

Two five bits loadable self-timed modulo $n$ counters were implemented in 350nm AMS CMOS Technology using standard cells from AMS library. Fig. 11 shows the die photo of the fabricated counters. It contains two identical loadable self-timed modulo $n$ counters. The area consumed by a control and counter block is $40\mu$m$^2$ and $30\mu$m$^2$ respectively.

A. Measurement Results

An FPGA was used to produce an acknowledgement for count outputs on $ar$ and $br$ channels in a four-phase handshake interaction with the counter. The time to produce and withdraw an acknowledgement was controlled.

Fig. 12 shows seamless count transition from count modulo 3 to count modulo 15 in which the counter starts the next count sequence after the end of its active operation.

Table III shows the response time of the counter obtained from the post-layout simulation at 3.3V. The pattern for even and odd numbers under the $ar \rightarrow br$ column can be explained by the different traces for zero channel operation in Fig. 8b that requires a different combination of logic gates. The trace $n21+, dr+$ occurs only for odd numbers, while the trace $n20+, dr+$ occurs for even numbers.

Fig. 13 show plots of the average power consumption of the counter for all thirty-one count sequences at different supply voltage for $1\mu$s and $5\mu$s acknowledgement delays, respectively. The power consumption was measured for a single counter and excludes power consumption of the pads.
For a given ack-delay and count sequence, the counter operated at a fixed frequency for different voltages. For the two ack-delays, the longer the delay, the lower the average power consumed. At 3.3V the average power for counts 9, 18, 31 are 169µW, 117µW, 157µW and 86µW, 30µW, 89µW @ 1µs and 5µs ack-delay, respectively.

### TABLE III
**RESPONSE TIME**

<table>
<thead>
<tr>
<th>Count</th>
<th>Response Time (ns)</th>
<th>(\text{Load} \rightarrow \text{ar} )</th>
<th>(\text{ar} \rightarrow \text{br} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>28.80</td>
<td>5.17</td>
<td>5.17</td>
</tr>
<tr>
<td>8</td>
<td>31.06</td>
<td>5.17</td>
<td>5.17</td>
</tr>
<tr>
<td>15</td>
<td>32.30</td>
<td>5.07</td>
<td>5.07</td>
</tr>
<tr>
<td>16</td>
<td>30.97</td>
<td>5.17</td>
<td>5.17</td>
</tr>
<tr>
<td>31</td>
<td>32.71</td>
<td>5.07</td>
<td>5.07</td>
</tr>
</tbody>
</table>

### TABLE IV
**COMPARISON WITH PREVIOUS SELF-TIMED COUNTERS**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[6]–[8]</td>
<td>Async. Fragments</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>[9]</td>
<td>Distributed Cells</td>
<td>N/A</td>
<td>&lt;5mW @ 80MHz</td>
</tr>
<tr>
<td>[13]</td>
<td>Distributed Cells</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>This paper</td>
<td>Distributed Cells</td>
<td>Yes</td>
<td>169µW @ 3.3V</td>
</tr>
</tbody>
</table>

### VIII. CONCLUSION

We presented the decomposition and specification of a loadable modulo—\(n\) counter into a linear array of interacting control and counter cells. The decomposition into array of counter cells was originally presented by Kessels [13], the counter cells can perform even and odd operations. The control cells determine from the binary input of the count modulo \(n\), the correct even and odd operation of each counter cell. The counter cells were specified using formal Petri Net models, which allowed verification and synthesis of the decomposed counter. The loadable modulo—\(n\) counter was implemented in 350nm CMOS technology. It operates correctly over a wide range of voltages and can perform seamless count transition. We used this counter in a fine-tunable DPWM circuit that produces constant duty ratio over a range of voltage supply with a controllable marginal error of 1% to 7%.

### ACKNOWLEDGEMENT

We thank Delong Shang for his many useful suggestions on VLSI design. This work is sponsored by EPSRC research grant A4A (EP/L025507/1).

### REFERENCES


