A single-rail re-implementation of a DCC error detector using a generic standard-cell library

7 Author(s)

K. van Berkel ; R. Burgess ; J. Kessels ; A. Peeters ; M. Roncken ; F. Schalij ; R. van de Wiel

View All Authors

Abstract:
We present a fully asynchronous implementation of a DCC Error Detector. The circuit uses 4-phase handshake signaling and single-rail data encoding, and has been realized using standard cells from a generic cell library. The circuit is obtained by fully automatic translation from a high-level (Tangram) description, using handshake circuits as intermediate architecture. In comparison with a previous double-rail implementation the fabricated IC is 40% smaller (core area), three times faster, and consumes only a quarter of the power. Switching between two power supplies is described as a technique to reduce power dissipation even further. A comparative evaluation also includes an improved double-rail implementation and two synchronous circuits.

Published in: Asynchronous Design Methodologies, 1995. Proceedings., Second Working Conference on

Date of Conference: 30-31 May 1995

INSPEC Accession Number: 5044974

Date Added to IEEE *Xplore*: 06 August 2002

DOI: 10.1109/WCADM.1995.514644

ISBN Information:

Publisher: IEEE

Conference Location: London, UK, UK

Keywords

IEEE Keywords
Detectors, Libraries, CMOS logic circuits, Asynchronous circuits, Delay estimation, Encoding, Power supplies, Power dissipation, Silicon, Circuit testing

INSPEC: Controlled Indexing
high level synthesis, digital audio tape, error detection codes, asynchronous circuits, cellular arrays, integrated logic circuits

INSPEC: Non-Controlled Indexing
power dissipation, DCC error detector, generic standard-cell library, single-rail re-implementation, fully asynchronous implementation, handshake signaling, single-rail data encoding, generic cell library, high-level Tangram description, handshake circuits, intermediate architecture

Authors

A Single-Rail Re-implementation of a DCC Error Detector Using a Generic Standard-Cell Library

Kees van Berkel* Ronan Burgess* Joep Kessels* Ad Peeters† Marly Roncken*
Frits Schalij* Rik van de Wiel†

Abstract

We present a fully asynchronous implementation of a DCC Error Detector. The circuit uses 4-phase handshake signaling and single-rail data encoding, and has been realized using standard cells from a generic cell library. The circuit is obtained by fully automatic translation from a high-level (Tangram) description, using handshake circuits as intermediate architecture. In comparison with a previous double-rail implementation the fabricated IC is 40% smaller (core area), three times faster, and consumes only a quarter of the power. Switching between two power supplies is described as a technique to reduce power dissipation even further. A comparative evaluation also includes an improved double-rail implementation and two synchronous circuits.

1 Introduction

In [14] we have presented a fully asynchronous implementation of a DCC Error Detector. This error detector was designed in the VLSI-programming language Tangram [9]. This program was then compiled fully automatically into a quasi delay-insensitive (QDI, [2]) circuit. The processed silicon comprises about 44 k transistors, is well testable, and was demonstrated successfully in an experimental DCC player. Most importantly, its power consumption is only a fifth of its synchronous counterpart. With a rapid growth of the markets for battery-powered electronic equipment and a continuous growth in circuit complexity we consider low-power digital circuits as a primary application area for asynchronous circuits.

However, the reported DCC error detector suffers two major drawbacks. Firstly, the IC is realized using a standard-cell library comprising many dedicated asynchronous cells, such as various C-elements and cells for double-rail logic, whereas the trend in industry is towards *generic cell libraries*. A generic CMOS standard-cell library comprises logic gates (NAND, NOR, AND, XOR, etc.), many complex gates (such as AND-OR-INVERT combinations), a set of inverters and drivers with a wide range of drive capabilities, and a few special functions, such as multiplexers, full adders, and D-type flip-flops. Generic libraries allow easy (cheap) retargetting of netlists to different technologies, possibly of different manufacturers. Also, many CAD tools, gate-array libraries, and FPGA tools support these generic libraries.

A second drawback is the considerable area overhead compared to existing synchronous designs. We estimate this overhead to be about 100%, and attribute this mostly to the quasi delay-insensitive (double-rail) implementation of the data paths. Double-rail circuits imply a doubling of the number of wires, where each wire requires a gate to drive it. The prime advantages of quasi delay-insensitive circuits are their robustness against variations in operating conditions and the simple observability of stuck-at-output faults.

Both the non-standard cell library and the substantial area overhead are considered to be major road blocks towards practical applications of asynchronous circuits for low power. This motivated us to investigate the single-rail [10] implementation (also known as bundled-data [11]) of data paths. The prospects of single-rail data encoding are:

+ a reduction in area and power combined with an increase in speed;
+ a better fit with generic cell libraries;
− a need for post-layout timing verification of the data-bundling constraints;

The compilation of Tangram into asynchronous circuits uses handshake circuits as intermediate architecture [13] (see also Figure 1). A handshake circuit is a network of handshake components, connected by point-to-point channels. The only interaction among handshake components is by means of handshake signaling along these channels. There are no global signals. Our library of handshake components consists of about 30 components, including a few hybrid components to interface to a non-handshake environment. Most components correspond directly to Tangram primitives, as a result of the syntax-directed translation. Almost all our tools produce, optimize, analyze, or simulate handshake circuits [15]. Hence, we decided that the new single-rail implementation had to be fully compatible with existing handshake circuits and the basis of handshake components. This turned out to be little of a constraint.

The DCC error detector was chosen as demonstrator, and allowed us to make a comparative evaluation with the earlier double-rail IC. Meanwhile, double-rail data paths experience an interesting revival [16]. By relaxing the requirement of quasi delay-insensitivity (QDI), using so-called extended isochronic forks, generic cell libraries become more practical, and considerable area reductions can be obtained. In the design of low-power synchronous

*Philips Research Laboratories, Eindhoven, The Netherlands
†Eindhoven University of Technology, Eindhoven, The Netherlands

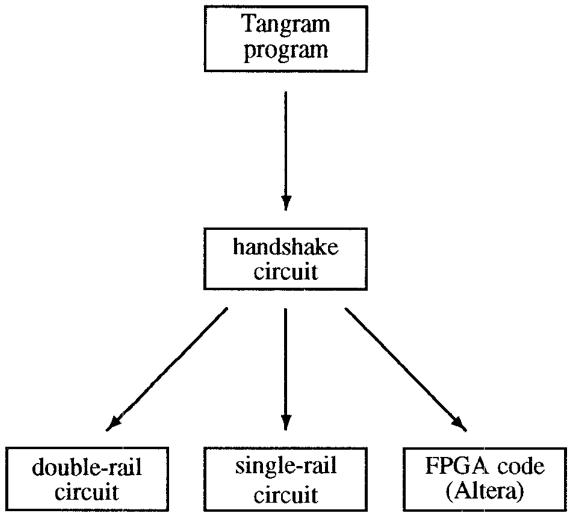

Figure 1: Alternative backends for the Tangram compiler.

circuits considerable progress is being made as well. Architectural modifications, including a reduction of clock frequencies, as well as clock gating resulted in a considerable reduction in power.

The evaluation presented in this paper is based on the single-rail and the double-rail chips and also includes recent double-rail and synchronous implementations.

The DCC error detector features a large spread in work load: correct code words require only 30% of the time and energy of those of (worst case) incorrect code words. The IC could operate on a much lower supply voltage if only correct code words would be offered, resulting in considerable further power savings. By switching between two power supplies most of this power savings can be realized, while still accommodating for incorrect code words. The implementation and impact are discussed below.

2 DCC Error Detector

In the DCC player parity symbols are recorded on tape to protect the audio information against tape errors. During play mode, a Reed-Solomon error detector accepts code words of 24 or 32 symbols (8 bits each), including 4 or 6 parity symbols. Each symbol has an associated 1-bit erasure flag, indicating the known error status of that symbol. Let e, r, and p denote the number of errors, erasures, and parity symbols respectively. The detector is required to output the positions and values of the e errors and the r erasures, provided that $2e + r \leq p$. See Figure 2 for the communication interface. For a formal specification of the detector see [18]. The DCC application specifies a performance of 3000 code words of length 24 and 2300 of length 32 per second. This implies an *average* input rate of 145,600 symbols per second.

The Euclid algorithm was adopted for the implementation of the detector. This algorithm comprises four main phases (cf. Figure 3): Syndrome Computation, Euclid, Chien Search, and Output. The syndrome is computed "on the fly" during input of the code word. After Syndrome Computation, the correctness of the input code word can be checked simply. For correct words the remaining two phases can be skipped. During the time allowed for Euclid and Chien Search an asynchronous CMOS circuit can then remain quiescent, consuming effectively zero energy (see also Figure 15 in [14]). The Tangram program has been designed for the lowest power for correct code words, given their prevalence ($\geq$ 95%). Compared with the earlier reported detector IC the Tangram program was improved, mainly by reducing the power consumption for syndrome computation with little effect on area and performance.

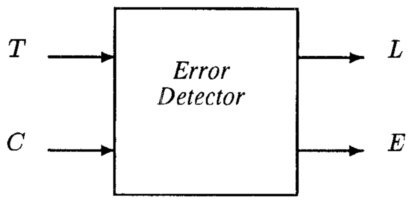

Figure 2: Error detector with communication ports: ports T and C for the input of code-word type and code-word symbols, ports L and E for the output of error locations and error values.

The Tangram program for the error detector including a discussion on the relevant VLSI-programming issues are described in [3]. The Tangram text comprises 430 lines, including 60 lines for the introduction of Galois arithmetic. The simulated timing of the four main phases is depicted in Figure 3. The compiled handshake circuit consists of some 2200 handshake components.

3 Generic standard-cell layouts

A generic standard-cell library contains only a few sequential gates, typically several master-slave flip-flops (D-types) often with scan-test facilities. Other sequential gates, such as (asymmetric C-elements) are not available as cells. Sequential gates of the form

$$\begin{array}{ccc} F & \mapsto & z\uparrow \\ \neg G & \mapsto & z\downarrow \end{array}$$

where F and G are boolean expressions over the inputs (without negation) can be realized with two combinational gates: $y = \neg(F \vee (z \wedge G))$ and the inverter $z = \neg y$. For all sequential gates with two inputs the corresponding complex CMOS gate can be found in a generic cell library. For instance, the asymmetric C-element as specified by the production rules

$$\begin{array}{ccc} a \wedge b & \mapsto & z\uparrow \\ \neg b & \mapsto & z\downarrow \end{array}$$

can be realized using two generic standard cells: the complex gate $y = \neg(b \wedge (a \vee z))$ and the inverter $z = \neg y$. However, for sequential gates with three inputs this is rarely

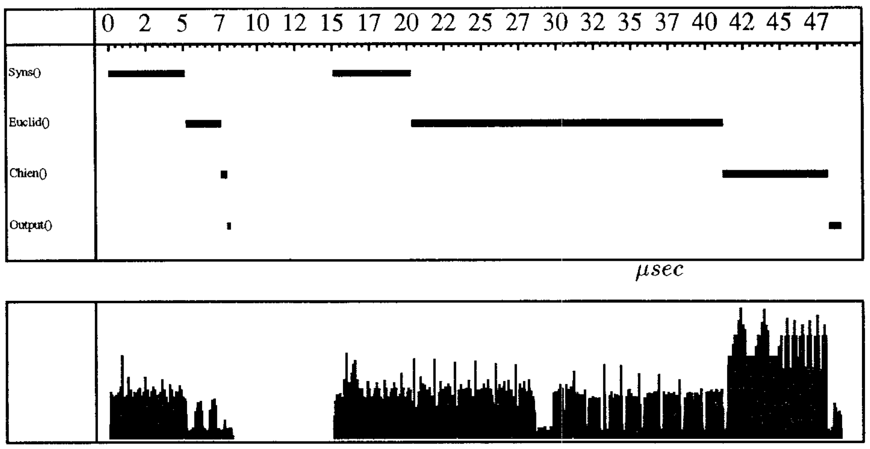

Figure 3: Simulated timing of the four main phases of the error detector (top) and energy dissipation (bottom) for both a correct code word (time interval [0..7]μsec.) and an incorrect code word (time interval [15..48]μsec.).

the case. For instance, a symmetric 3-input C-element cannot be realized this way.

The fixed output drive of standard cells may result in poor output transitions for gates with a large fanout. This problem is amplified by the large variation and unpredictability of net capacitances in standard-cell layouts. These poor transitions tend to degrade performance, waste power, may cause hazards in isochronic forks, and complicate delay matching. Fortunately, generic cell libraries have a set of inverter cells with a wide range of drive strengths. As a result, rise (or fall times) for highly capacitive loads can be kept within a few inverter delays. For instance, in a 0.8 μ CMOS process, a fanout to 32 gates plus 10 mm interconnect results in a capacitive load of about 3pF. Rise and fall times can be kept as low as about 1ns (see Figure 4) or about twice the delay of an unloaded inverter.

We adopted the following pre-layout "driver strategy":

1. sum the cell-input capacitances for each net;
2. add a proportional fraction to account for the (unknown) wiring capacitance (typically 50-100%);
3. bound transition times by strengthening of inverters and by insertion of drivers for high-capacitance nets.

This simple pre-layout strategy requires post-layout verification to check the validity of the assumption on the wiring capacitance. The strategy proved effective for the detector. For larger circuits the cell placement can be taken into account to improve the prediction of the wiring capacitances.

The implementation of isochronic forks did not pose any problems [12]. Our driver strategy bounds transition times to a few nanoseconds. Furthermore, the variation

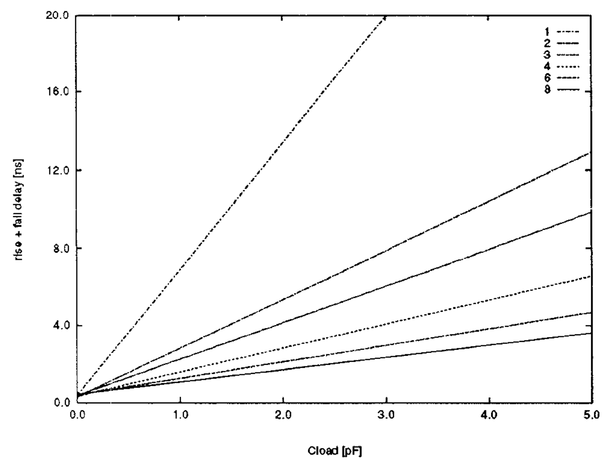

Figure 4: Delay times (rise + fall) for a range of inverters. Number denotes relative transistor widths.

in logic threshold voltages in the generic cell library is modest: at most +/− 8% of the supply voltage. This variation is that low because all gates are static and long chains of n-MOS and p-MOS transistors are avoided.

4 Control structures

The handshake channels in the control part of the handshake circuit, with few exceptions, do not convey data. The handshake then reduces to a simple two-wire 4-phase handshake. The conversion to the generic library can be carried out by simple substitution at the gate-level. The dominance of simple gates (mostly OR gates) and (asymmetric) C-elements made the conversion to a generic cell library straightforward. The resulting control circuitry is still QDI and requires about 15% more transistors then the control circuit based on the dedicated cell library.

Two types of optimizations have been applied to the control circuits. Firstly, a number of handshake components have been generalized to multi-channel versions. For instance, a tree of $N-1$ binary mixers (CALL-components) can be replaced by a single N-port mixer. This substitution alone reduced the control circuitry by 12%, and resulted in an interesting performance gain, because of the reduced logic depth. Another 3% reduction in costs was obtained by adopting the multi-channel sequencer of [1].

Secondly, a number of peep-hole optimizations at the gate level have been applied, resulting in another reduction of about 7%. As a consequence of the relatively small basis of handshake components and the syntax-directed translation from Tangram, inefficiencies may occur across the boundaries of handshake components. Quite a few peep-hole optimizations are possible, by which a simple subcircuit comprising a few gates is replaced by an equivalent but cheaper subcircuit. A typical optimization is shown in Figure 5. All these gate-level peep-hole optimizations remain in the realm of QDI.

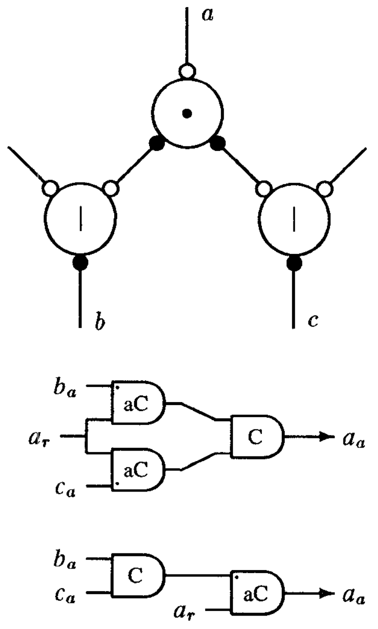

Figure 5: Fork-mixer optimization. (top) handshake circuit, (middle) direct implementation of a subcircuit, (bottom) optimized subcircuit. The remaining C-elements can be further combined into one asymmetric three-input C-element.

5 Single-rail data paths

In a single-rail implementation of asynchronous data paths a *data valid* wire is used to indicate the validity and stability of the N wires of an N-bit data word. Both a premature indication and an expired indication of validity may be fatal. To assure correct timing of the validity signal, matching of delays in data (-valid) paths cannot be avoided.

In combination with 4-phase handshake signaling in the control circuits, the selection of an appropriate data-valid scheme (defining the data validity relative to the handshake phases) is critical, for it determines circuit costs and performance. Note that in this respect handshake circuits are more general than micropipelines [11]: in some handshake channels data (validity) may be encoded in the request phase, in others in the acknowledge phase. The "reduced broad scheme" as introduced in [7] has been applied to the Error Detector. It can be applied to all compiled Tangram handshake circuits and has the attractive property that the return-to-zero phases of both the control circuitry and data-path circuits have become productive.

In CMOS circuits delays depend to a large extent on the lengths of the wires involved (i.e. their capacitances). Their actual values become available only after the cells are laid out. Since *differences* in delays also depend on processing technology (transistor gains and thresholds, parasitic capacitances) and on operating conditions, a suitable margin is required to guarantee correct operation. One may, for instance, implement the delay-matching such that the data-valid path has twice the delay of the slowest path in the data circuit [6]. The correct implementation of the data-valid path generally requires post-layout verification.

Apart from the delay matching, the resulting data path building blocks are very similar to traditional synchronous building blocks. Hence, they can generally be realized efficiently by means of cells of a generic library. A major difference, however, is the use of latches, rather than master-slave flip-flops.

It is interesting to look at the energy consumption during the execution of the assignment $z := x \oplus y$. The energy consumption in the combinational circuit for $\oplus$ does not occur during this assignment, but when the value of x or y is being updated. Similarly, each expression in which z occurs is re-evaluated with the above assignment. This side effect makes variables with many read-ports suspect (powerwise), especially if these read ports are connected to deep combinational circuits such as ALU's or multipliers. The VLSI programmer for low-power should be aware of this, although his task is complicated by this decrease in transparency.

The single-rail re-implementation of the detector had to be pin compatible with the existing double-rail version. In order to enable experiments in a single-rail environment as well, switchable conversion elements were included (cf. Figure 6). Handshake channels t, c, e, and l are single-rail encoded. Channels T, C, E, and L are either single-rail encoded (when *i/o-mode* is low) or double-rail encoded (when *i/o-mode* is high). The conversion elements are akin to those of [10]. Basic versions of these conversion elements have been generated automatically by means of the ASSASSIN compiler [4].

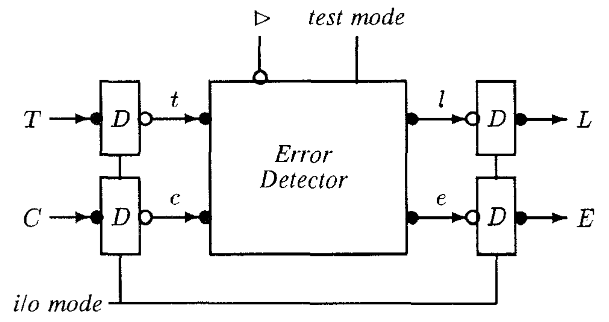

Figure 6: Error detector encapsulated with switchable conversion elements D from/to double-rail data-encoding.

By going from double-rail to single-rail data paths the problem of testing for fabrication faults changes significantly. Observation of stuck-at output faults by means of deadlock is no longer sufficient. Modeling stuck-at faults at the handshake circuit level allowed for a high-performance fault simulator [17]. Together with automated feedback at the Tangram level interactive *design* of test patterns has become a feasible task. A form of partial scantest [8] has been applied to the error detector to simplify the design of test patterns, and to reduce their number [17]. The scan facilities cover 80% of the latches and result in an area overhead of only 4%. For scan-in and scan-out we re-use channels C and E.

6 Switching power supplies

From Figure 3 we can see that the processing of a correct takes only one quarter or the time required for a (worst-case) incorrect word. Hence, we could safely process correct code words at about one third of the regular supply voltage, and thus reduce the power consumption by an order of magnitude! This is especially interesting, given the correctness of more than 95% of the code words. However, the decoder must at all times be ready to process an incorrect code word.

Nielsen *et al* [5] describe a technique of adaptive scaling of the supply voltage which takes advantage of such variations in workload. The technique is based on a DC/DC converter controlled by a feed-back loop that minimizes the phase difference between the observed and the required output (or input) stream. Kessels [3] proposes a simpler scheme based on two fixed supply voltages, which can be applied if an increase (and decrease) in the workload can be predicted. This scheme is explained below for the error detector.

The applied error-correction algorithm consists of four major phases (cf. Figure 3): Syndrome Computation, Euclidian Division, Chien Search, and output. The latter three require about 75% of the computation time (worst case) and can (almost) be skipped for correct code words. The idea is to slow down the Syndrome Computation by supplying a low supply voltage, and to speed up the remaining two phases by supplying a high supply voltage. By trading-off these two separate supply voltages it is still possible to meet worst-case timing requirements while simultaneously reducing the power consumption.

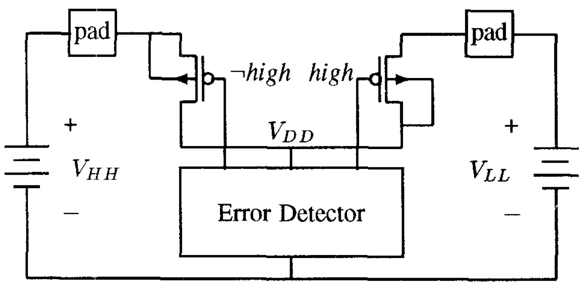

Figure 7: Switching between power supplies V_{LL} and V_{HH}, assuming $V_{LL} \leq V_{HH}$.

A circuit implementation of this scheme is shown in Figure 7. The power switches are implemented by very wide p-MOS transistors. Their layouts are based on the layout of a pad output-driver combination. In order to ensure a reverse-biased condition of the source-well diode the circuits and layouts for the two switches have been implemented slightly differently. For the V_{HH} switch the n-well must be connected to the bonding pad, and for the V_{LL} switch the n-well must be connected to V_{DD}.[1] The measured voltage drop across the p-MOS transistor is only 82 mV.

In order to avoid a short-circuit current from supply V_{HH} to V_{LL} the intermediate state in which node V_{DD} is connected to both supplies ($\langle\neg\text{high}, \text{high}\rangle = \langle 0,0\rangle$) must be prevented from occurring. This implies that V_{DD} is left floating for a short period, which is generally not a problem because of the relatively large capacitance of the on-chip power rail. In the error detector the state of the switches is changed only in states with no other activities taking place and the floating period is very short ($<$ 1 nanosecond).

In double-rail mode, all chip inputs and outputs are low between handshakes, because the communication with the environment is implemented via active handshake ports. The power switch is used in such a way that communication with the off-chip environment always takes place at the low supply voltage V_{LL}. This guarantees that all chip outputs and inputs are low when the on-chip V_{DD} is connected to V_{HH}. Therefore, the power switches required no level shifters.

The single-rail chip has three pads for power, V_{DD}, which connects directly to the on-chip power rail, and V_{SS} and V_{HH}, which connect to the power switch. During experiments with the power switch the V_{DD} terminal can be left dangling and can be used to monitor the on-chip power supply. The power switch can be overruled by connecting the power supply directly to the V_{DD} terminal and leaving the other two dangling. In a product,

[1]Here we assume an n-well based CMOS technology. Otherwise a similar asymmetry applies to the substrate connections.

a solution based on on-chip level-shifters may be more attractive then having multiple power-supplies.

7 Results and comparative evaluation

In June '94 we froze the verified netlist, and a layout was realized in a 0.8 μm CMOS process. The resulting silicon (November '94) passed all tests, and proved audibly correct in an experimental DCC player. In this section we present the measured area, speed, and power of the IC and compare these to a double-rail IC and a synchronous IC. This comparison is somewhat complicated by other differences than timing, including differences in Tangram text, control optimizations, driver strategy, cell library, and CMOS feature size. Where appropriate, we take these differences into account and attempt to isolate the differences between single-rail and double-rail.

Area

The core area (IC, excluding periphery) measures 4.5 mm^2. A second layout with a 3.9 mm^2 core was designed after further control optimizations and after stripping the switchable power supplies and switchable *i/o*-conversion elements. This compares favorably to the 7.0 mm^2 of the double-rail implementation (11 mm^2 in 1.0 μm CMOS).

In comparison with the double-rail version the control circuitry reduced little in size: the peep-hole optimizations cancel more or less the absence of dedicated standard cells of C-elements. Almost all savings were obtained in the data paths, due to the absence of completion-detection, the trivial read ports of handshake registers, and the much simpler gates to implement the combinational logic. As a result the fraction of the control circuitry rose from 18% in the double-rail layout to 35% in the single-rail layout.

The latest synchronous implementation of the function occupies about 3.3 mm^2, corresponding to an area overhead of about 20% for the stripped single-rail version. Given the many similarities between the respective data paths we think that difference stems from the distributed nature of the asynchronous control circuit and its quasi delay-insensitive implementation. Further optimizations for the control circuits are clearly needed. Also, adding a few asynchronous cells to the generic library may help to reduce the remaining overhead.

The circuit is fully testable against stuck-at output faults at the cost of only 4% of the core area. This is due to the design-for-test approach in which structures already present in the datapath are reused for scan [8]. A straightforward full-scan solution (as in the synchronous circuit) would require the latches to be replaced by scan flip-flops and would lead to considerable overhead.

Meanwhile, recent insights allowed us to redesign the double-rail circuit using generic standard cells [16]. Substantial reduction in circuit costs could be obtained by means of peep-hole optimizations across the boundaries of handshake components (similar to those explained in the section on control structures). In addition, the introduction of so-called *extended isochronic forks* made it possible to optimize the double-rail data paths considerably. Together this resulted in a reduction from 44 to 32 thousand transistors. Due to the less dense standard cells, the core area only reduced from 7.0 to 6.5 mm^2.

Power

The decoder IC dissipates about 0.5mW at 5V, assuming the DCC-specified mix of the two types of code words and 95% correct code words. The stripped version was laid out better, and hence had considerably lower parasitic capacitances. By simulating both layouts after backannotation of extracted capacitances we found a reduction in power consumption by 30%.

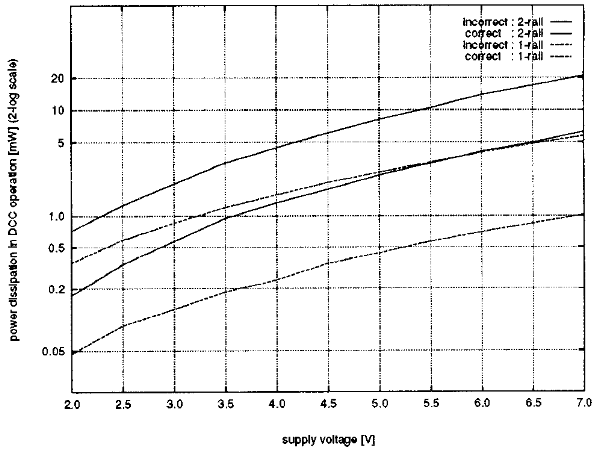

Figure 8: Measured power consumption versus supply voltage for correct and incorrect code words.

The measured power dissipation of the single- and double-rail ICs is shown in Figure 8. (Note the logarithmic scale.) The ratio between the power for incorrect and correct words at 5V is close to 5 for the single-rail case. For the double-rail circuit this ratio is only 3. The increase is due to the improvement of the Tangram program.

The best-case double-rail curve and the worst-case single-rail curve nearly match around 5V. The steeper rise for the double-rail IC is due to the excessive short-circuit power of the double-rail IC[2]. By computing

$$1 - \left(\frac{5}{2}\right)^2 \times \frac{\text{power @ 2V}}{\text{power @ 5V}}$$

we estimate the short-circuit dissipation about 15% at 5V for the single-rail circuit, compared to 40% for the double-rail circuit.

By comparing the worst-case power at 2 V we see a power advantage of a factor 2 for the single-rail IC. The remaining differences (control optimizations, transistor sizes, and technology) cancel each other more or less, suggesting that this factor represents the single-rail advantage.

The latest synchronous decoder dissipates about 2.6 mW (at 5V) for correct code words and 5.0 mW for code words with six erasures. Under typical DCC conditions this gives an advantage of a factor 5. The 3 MHz clock, even when gated part of the time, enables the flip-flops still far too often, given the input symbol-rate of about 150

[2]Due to a poor driver strategy.

kHz. Also, the ROM-based centralized controller dissipates considerably at 3 MHz, compared to the highly distributed handshake circuit consisting mostly of sequencers and mixers.

Speed

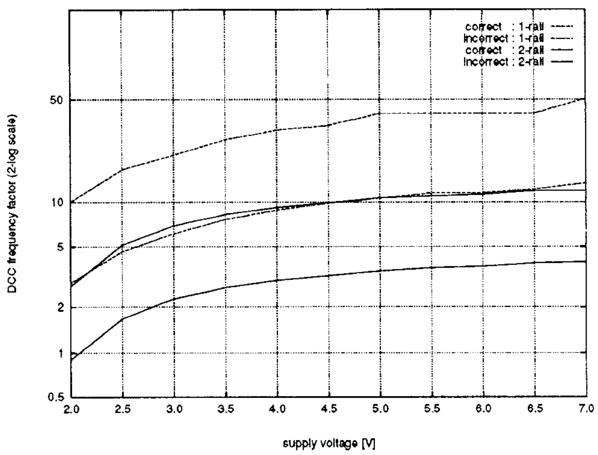

Figure 9: Measured execution frequencies versus supply voltage for correct and incorrect code words. The specified frequency for DCC application is used for normalization.

Figure 9 shows the decode frequencies as function of the supply voltage. (Note, again, the logarithmic scale.) The frequency of the DCC specification is taken as unit. At 5V, the decoder has an excess performance of a factor 10, whereas a safety factor of 2 suffices in practice. This excess performance could be used to reduce the costs of the circuit by eliminating some parallelism from the Tangram text. Alternatively, the power consumption can be reduced dramatically (9×) by lowering the supply voltage to 2.0V, or even lower by switching the power supply voltage between 1.6 and 2.5 volts.

Interestingly, the worst-case single-rail performance nicely matches the best-case double-rail performance. The relative speed improvement is a factor three. Factoring out the speed gain by technology, layout, and control optimizations, we estimate the single-rail advantage to be about a factor 1.5. To some extend the safety factor 2 applied in delay matching in the data path corresponds to the overhead of the return-to-zero phase. The factor 1.5 is then due to the absence of completion detection and to simpler CMOS cells in the data paths (fewer transistors in series).

Correct behavior of the IC can still be observed at 1.2V. This wide supply-voltage range with correct behavior is another indication of the robustness of the applied delay matching.

Summary

In Table 7 six circuits of the DCC error detector are compared: two single-rail circuits, two double-rail circuits, and two synchronous circuits. The four asynchronous circuit have been obtained by compilation from a Tangram program.

Circuit *1-rail*$_1$ is the single-rail IC presented in this article. For the sake of a better comparison, we derived *1-rail*$_2$ by removing the double-rail to single-rail converters (-400 transistors), and the power switches (-150 transistors). Also, a number of additional peep-hole optimizations not available for *1-rail*$_1$ have been applied (-1000 transistors).

Circuit *2-rail*$_1$ is the IC reported in [14]. The circuit also uses 4-phase handshake signaling, but double-rail data encoding. The circuit is QDI and is implemented using a dedicated asynchronous cell library. In particular, layout-cells for double-rail arithmetic were required to keep the layout costs reasonable. Layout *2-rail*$_2$ is based on a double-rail circuit based on generic standard cells and extended isochronic forks [16]. The same controller optimizations have been applied as for *1-rail*$_2$.

Circuit *sync*$_1$ is part of a synchronous IC in the current DCC players. Its successor *sync*$_2$ is optimized towards low power consumption. Architectural modifications allowed the halving of the clock frequency. Furthermore, by means of clock gating, power could be reduced further. The latter also decreased the power ratio for best over worst case. Both synchronous circuits are fully scan testable, accounting for about 3-5% of the circuit costs. Interestingly, the factor 5 in power for the best case between *2-rail*$_1$ and *sync*$_1$ recurs between *1-rail*$_1$ and *sync*$_2$.

8 Conclusion

The single-rail DCC error detector proves that single-rail circuits can be realized efficiently in a standard-cell layout style using a generic standard-cell library. This library provides a good basis for the single-rail datapath, and even QDI realizations of the control path can be obtained with little penalty in area, speed, and power.

In the single-rail datapath, the choice of relating data validity to the four phases of a handshake proved very critical. By adopting the "reduced broad scheme" all four phases could be made productive. The required delay matching is well manageable, despite the fixed transistor sizes in the generic cells and the unpredictability and variation in wiring capacitances in standard-cell layout. An effective driver strategy proved essential.

The single-rail error detector is only 20% larger than the equivalent synchronous version, and its power consumption is only a fifth of the latest synchronous version.

The re-implementation of the DCC error detector handshake circuit as single-rail circuit also demonstrates the neutrality of handshake circuits with respect to different asynchronous implementations. The closing gap in circuit costs between the single-rail and the synchronous versions also gives evidence that the use of handshake circuits as intermediary does not introduce inefficiencies.

Acknowledgements

Gert-Jan Hekelaar and Arnold Gruijthuijsen laid out the standard-cell circuit, and Jan Stuyt assisted in the layout of the power switches. Steven Vercauteren (IMEC) proposed circuits for the single-rail to double-rail converters using ASSASSIN. Peter Arts and John Sherry (Philips Consumer Electronics) have designed the synchronous versions and assisted us in the comparative evaluation.

	unit	1-rail$_1$ IC	1-rail$_2$ layout	2-rail$_1$ IC*	2-rail$_2$ layout	sync$_1$ IC-part	sync$_2$ IC-part
# transistors	×1000	21.8	20.3	44.0	32.0		
Core area	mm^2	4.5	3.9	7.0	6.5	3.4	3.3
Time (best**)	μs	8.7	7.3	14.4	10.3	N.A.	N.A.
Time (worst**)	μs	38.1	31.3	51.2	42.0	N.A.	N.A.
Energy (best**)	μJ	0.08	0.06	0.41	0.14	2.6	0.6
Energy (worst**)	μJ	0.54	0.42	1.50	0.87	3.1	1.1

Table 1: Six implementations of the error detector compared (see text).
Cycle times and energy at 5V for code words of size 32.
*Data have been normalized to a 0.8 μ CMOS process.
**Best and worst case refer to correct code words and code words with 6 erasures respectively, where typical is close to best case for timing and energy.

Ferry Huberts (University of Twente) provided the numbers for column 2-rail$_2$ in Table 7. With Cees Niessen, Martin Rem (Eindhoven University), Shiv Sikand and Craig Farnsworth (both Manchester University) we have had fruitful discussions on single-rail circuits. Part of the presented work has been funded by the European Commission under ESPRIT contract 6143 (EXACT). Also the ESPRIT Working Group ACiD proved an interesting forum for discussions on the asynchronous implementation of the error detector.

References

[1] Andrew Bailey and Mark Josephs. Sequencer circuits for VLSI programming. These Proceedings.

[2] Steven M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits.* PhD thesis, California Institute of Technology, 1991.

[3] Joep Kessels. VLSI programming of a low-power asychronous Reed-Solomon decoder for the DCC error player. These Proceedings.

[4] Bill Lin, Chantal Ykman-Couvreur, and Peter Vanbekbergen. A general state graph transformation framework for asynchronous synthesis. In *Proc. European Design Automation Conference (EURO-DAC)*, pages 448--453. IEEE Computer Society Press, September 1994.

[5] Lars Skovby Nielsen, Cees Niessen, Jens Sparso, and Kees van Berkel. Low-Power Operation Using Self-Timed Circuits and Adaptive Scaling of the Supply Voltage. *IEEE Transactions on VLSI Systems*, 2(4):391--397, 1994.

[6] N. C. Paver. *The Design and Implementation of an Asynchronous Microprocessor.* PhD thesis, Department of Computer Science, University of Manchester, June 1994.

[7] Ad Peeters and Kees van Berkel. Single-rail handshake circuits. These Proceedings.

[8] Marly Roncken. Partial scan test for asynchronous circuits illustrated on a DCC error corrector. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 247--256, November 1994.

[9] Frits Schalij. Tangram manual. Technical Report UR 008/93, Philips Research Laboratories Eindhoven, P.O. Box 80.000, 5600 JA Eindhoven, The Netherlands, 1993.

[10] Charles L. Seitz. System Timing. In C.A. Mead and L.A. Conway, editors, *Introduction to VLSI Systems*, chapter 7. Addison-Wesley, 1980.

[11] Ivan Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720--738, 1989.

[12] Kees van Berkel. Beware the isochronic fork. *Integration, the VLSI journal*, 13(2):103--128, 1992.

[13] Kees van Berkel. *Handshake Circuits. An asynchronous architecture for VLSI programming.* International Series on Parallel Computation 5. Cambridge University Press, 1993.

[14] Kees van Berkel, Ronan Burgess, Joep Kessels, Ad Peeters, Marly Roncken, and Frits Schalij. A Fully-Asynchronous Low-Power Error Corrector for the DCC Player. *IEEE Journal of Solid-State Circuits*, 29(12):1429--1439, 1994.

[15] Kees van Berkel, Ronan Burgess, Joep Kessels, Ad Peeters, Marly Roncken, and Frits Schalij. Asynchronous Circuits for Low Power: a DCC Error Corrector. *IEEE Design & Test*, 11(2):22--32, June 1994.

[16] Kees van Berkel, Ferry Huberts, and Ad Peeters. Stretching QDI in double-rail handshake circuits. These Proceedings.

[17] Rik van de Wiel. High-level test evaluation of asynchronous circuits. These Proceedings.

[18] Tom Verhoeff. Z Specification for DCC Error Decoder. Technical Report 93/?, Eindhoven University of Technology, 1993.