Towards Asynchronous A-D Conversion *

D.J.Kinniment, B.Gao A.V.Yakovlev, F.Xia
Dept. of Electrical and Electronic Eng. Dept. of Computing Science
University of Newcastle upon Tyne
Newcastle upon Tyne NE1 7RU, UK

Abstract

Analogue to digital (A-D) converters with a fixed conversion time are subject to errors due to metastability. These errors will occur in all converter designs with a bounded time for decisions, and are potentially severe. We estimate the frequency of these errors in a successive approximation converter, and compare the results with asynchronous designs using both a fully speed-independent, and a bundled data approach. It is shown that an asynchronous converter is more reliable than its synchronous counterpart, and that the bundled data design is also faster, on average, than the synchronous design. We also demonstrate tradeoffs involved in asynchronous converter designs, such as speed, robustness to delay variations, circuit size and design scalability.

Keywords: analogue to digital conversion, arbitration, asynchronous circuits, metastability, signal transition graphs, synchronisers.

1 Introduction

N-bit analogue to digital (A-D) converters are usually specified to have a fixed conversion time, but conversion cannot be done in bounded time with complete reliability, since it requires the resolution of a continuous variable into 2^n discrete states. Because the analogue input can be infinitely close to the required boundary between two states, metastability can always be caused in the state determining hardware, and such behaviour has been reported [1]. The problem is closely related to that of arbitration between two or more asynchronous requests since in that case the time between requests [2, 3] is infinitely variable, and must be resolved into one of two or more discrete outcomes. At its root, the problem arises from the fact that there are infinitely more real numbers than there are integers [4], and that the physical mechanism used to determine the state has a limited gain-bandwidth product. We can therefore have a decision with a finite probability of error in a fixed time, or a decision with a guaranteed accuracy which may require infinite time, and proofs of this fact have been published [5, 6].

Analysis of the behaviour of metastability [3, 7] has shown that the probability of the output remaining in a metastable region decreases exponentially with time, and that therefore the probability of error resulting from a bounded decision time also decreases exponentially. If these errors were confined to the least significant bit, the consequent problems would not be serious, but it can be shown that metastability in the most significant bit can result in an error of a quarter of full scale or

*This work is supported by EPSRC, GR/K70175 (Project HADES)

more, and it is important that such problems are understood. An asynchronous comparator based on previously reported arbiters [3, 7], can be built which allows sufficient time for a decision to be made, and asynchronous converters based on this component have an exponentially decreasing probability of conversion times longer than a minimum time. Such a converter can be both more reliable, and, on average, faster than the conventional, bounded time converter. The latter is clocked on a bitwise basis and hence is prone to errors due to metastability in high-significance bits. The asynchronous one, even if working within bounds on the overall conversion time is likely to be affected only in its low-significance bits.

In this paper we use the SPICE simulator to model a simple successive approximation A-D converter, then show that quite significant errors can arise. The probability of error as a function of clock period is then given and compared with theory. Two types of self-timed converters are then modelled and their performance measured, showing that data errors due to metastability can be eliminated, and that the self-timed systems can be both more reliable and have a better performance. The first type of design is based on a bundled data (BD) approach. Although it employs an asynchronous metastability resolver, it uses fairly tight assumptions about delays in the parts of the logic circuit which are meant to work in parallel. The second type is based on speed-independent (SI) logic, using explicit completion detection logic and individual gate acknowledgement. It is designed from a formal behavioural specification, a Signal Transition Graph, using appropriate synthesis techniques and software tool petrify [8]. The SI design (three options have been studied, different in their logic complexity and design scalability) loses to the BD one in speed but gains in robustness of the implementation if the physical delay parameters of logic gates are variable.

2 Synchronous A-D Converter

2.1 Basic algorithm and converter model

We have designed a simple CMOS 4 bit *successive approximation* A-D converter based on the diagram of Figure 1. The comparator is designed to respond to a voltage difference of half the least significant bit within 4 nS, and the remaining loop delays are less than 4 nS. This has been simulated using the MIETEC 2μ CMOS SPICE model, and the analogue input carefully adjusted until metastability occurred in the most significant bit of the output.

The internal state and output registers are based on a master-slave in which a positive edge on the clock causes the master to latch, and the slave to become transparent. If there is a conflict, or indeterminate levels at the master set and reset inputs when the clock goes high, this may cause metastability, which could result in a change at the slave output at a time much later than the following positive clock edge. The 4-bit converter follows the steps of the algorithm shown in Table 1; it goes through 5 states in order to complete and output a digital sample.

2.2 Synchronous converter results

The converter is clocked every 10 nS as shown in the HSPICE waveforms of Figure 2. The converter enters state 1 at 6 nS following the positive going clock edge at 5.5 nS. The output, CB3 – CB0 then goes to 1000 and the D-A converter responds at 7.5 nS producing an output close to the analogue

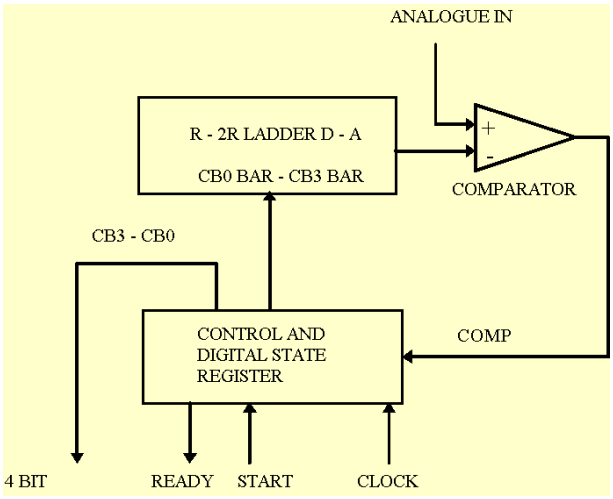


Figure 1: Synchronous Analogue to Digital Converter

Ready	State	MSB (CB3)	CB2	CB1	LSB (CB0)
1	0	X	X	X	X
0	1	1	0	0	0
0	2	Comp	1	0	0
0	3	X	Comp	1	0
0	4	X	X	Comp	1
1	0	X	X	X	Comp

Table 1: Control of Converter ('X' stands for any value, 'Comp' denotes the value produced by comparator)

input. At about 13 nS the comparator output goes high, and the master CB3 output (waveform XX3.Q) moves to low at 14 nS, because the clock is low. At 15.5 nS the clock goes high again, and CB3 is trapped in metastability. If the output of CB3 were to go to either a high or a low state before 20 nS, the subsequent conversion would end at 1000 or 0111, but the half level output from the master, allows CB3 slave output to stay high, and when the master finally goes low at 21 nS CB3 output changes state too late to prevent CB2, which has been responding to the high output of CB3, being reset low. The result is a value of 0011. If the metastability were to last for the full conversion time before going to zero, which is less likely but still possible, the result could be 0000. These outputs represent potentially disastrous errors of a quarter and a half full scale respectively.

2.3 Metastability time

The time required by the output of a bistable to come out of the metastable state is dependent on the voltage difference between the true and inverse outputs at time $t = 0$. The time involved can be calculated from a small signal model of its constituent gates provided that the trajectories of the outputs remain in the region where the small signal model applies for most of the time. The time

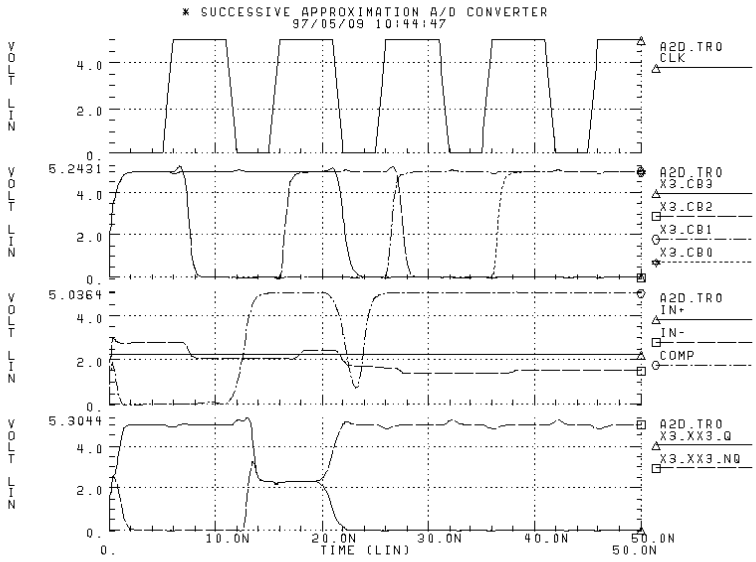
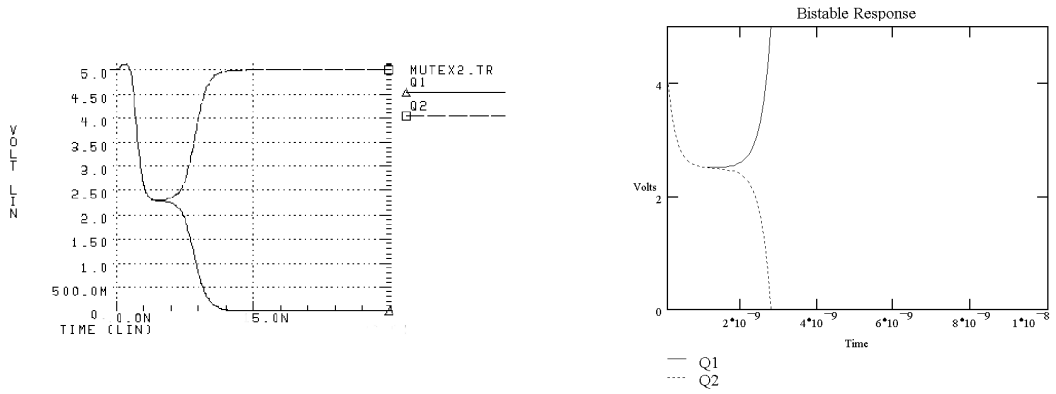


Figure 2: Synchronous converter simulation



(a) (b)

Figure 3: Metastable response: (a) SPICE simulation and (b) small signal model

domain response is given [9, 10] by:

$$Q1 \quad = \quad K_1 e^{\frac{t}{\tau}} + K_2 e^{\frac{-t}{\tau}} \quad \text{and} \quad Q2 \quad = \quad -K_1 e^{\frac{t}{\tau}} + K_2 e^{\frac{-t}{\tau}},$$

where $Q1$ and $Q2$ are the two outputs, and $1/\tau$ is the gain bandwidth product of the small signal model of the gates used. A comparison between a SPICE simulation and this model is shown in Figure 3.

Typical values for τ in the MIETEC 2μ CMOS SPICE model we are using are in the range 0.25 – 0.4 nS. If the initial difference between the two outputs V_m is known, the metastability time t_m can be calculated from:

$$t_m = \tau \ln \left[\frac{V_{high} - V_{low}}{V_m} \right].$$

Thus if both bistable outputs are initially within a band of 1μV either side of metastability, and the swing $V_{high} - V_{low}$ is 5V, the bistable will require approximately 5 nS to reach either high or low. The effect of increasing the value of V_m by a factor A, is to reduce the metastability time t_m by $\tau \ln[A]$.

In an A-D converter the comparison between the input analogue voltage and the output of the D-A converter is normally done by an amplifying comparator, before being fed through a number of gates to the output register. If we use a comparator made from the same CMOS technology as the gates forming the bistable, they will normally have similar gain-bandwidth characteristics. Hence, we can expect that a comparator will delay the input signal by at least $\tau \ln[A]$ for every factor A increase in the input signal. At the same time, the comparator amplifies the value of V_m by the factor of A, which helps reduce the bistable metastability time t_m by $\tau \ln[A]$. The net effect of a comparator made from the same technology as the bistable on time taken to make the decision is therefore negligible.

We can compute the probability of failure due to metastability from the probability of a particular analogue input being within V_m of any reference line. The reference lines are separated from each other by the value of LSB, which is the change in analog voltage required to give a change of digital output of the least significant bit. If the difference in voltage is V_m, and the effect of comparator can be ignored, then the probability of an analogue input being within V_m of any reference line is given by $P = \frac{2V_m}{LSB}$. The probability, P, of metastability occurring for longer than t_m is then:

$$P = \frac{2(V_{high} - V_{low})}{LSB} e^{\frac{-t_m}{\tau}}.$$

We have measured the metastability times in our model for different values of analogue input voltage. We then compute the probability of each input being within a given margin of the decision between two discrete outcomes, and hence the probability of metastability lasting longer than a particular time. Figure 4 compares the results obtained by this method with the function given above where $\tau = 0.36$ nS, and shows that as the clock cycle time (i.e., the value of t_m) increases, the probability of error reduces exponentially. With a cycle time of 10 nS, there is approximately one in every 10^{11} samples.

3 Asynchronous converters

The comparator used for our simple model has been replaced by a bistable circuit, Figure 5, based on previously reported arbiters [3, 7], in order to produce two asynchronous designs. In this comparator OUT0 and OUT 1 are both low when the CLAMP waveform is high, and only one goes high when there is sufficient difference between the bistable nodes NET0 and NET1. A timing waveform can therefore be derived from OUT0 and OUT1 which indicates when the comparison is complete, and the possibility of error is eliminated and replaced by a variable decision time.

In a successive approximation converter only one of the n comparisons involves a long metastability time, the other $n-1$ are comparisons between voltages greater than the least significant bit (LSB), and are therefore relatively fast. For the worst case comparison, the average metastability time is given by taking the average of all times resulting from a voltage difference of between 0 and $LSB/2$ [10]:

$$\begin{aligned} t_{ave} &= \frac{2\tau}{LSB} \int_0^{\frac{LSB}{2}} \ln\left(\frac{V_{high}-V_{low}}{V}\right) dV, \\ t_{ave} &= \tau\left[1 + \ln\frac{2(V_{high}-V_{low})}{LSB}\right]. \end{aligned}$$

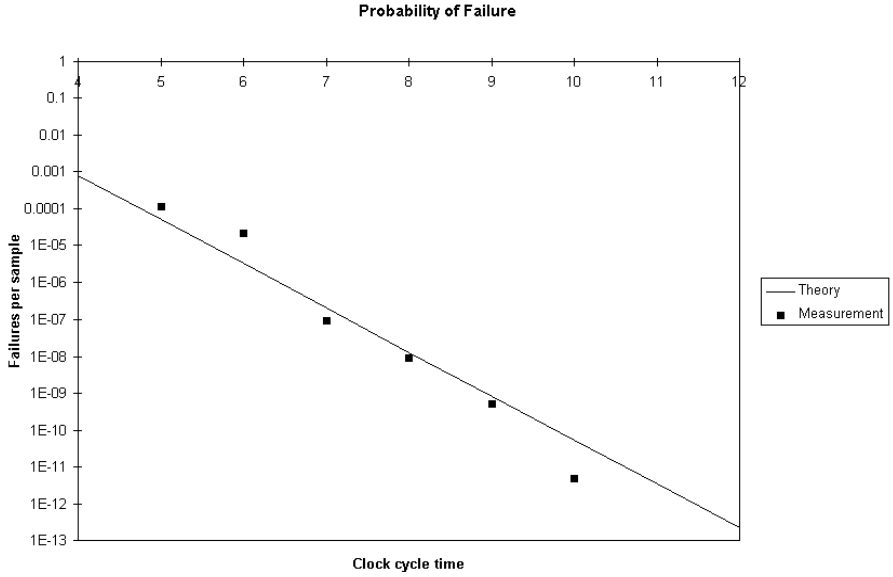


Figure 4: Converter failure rate

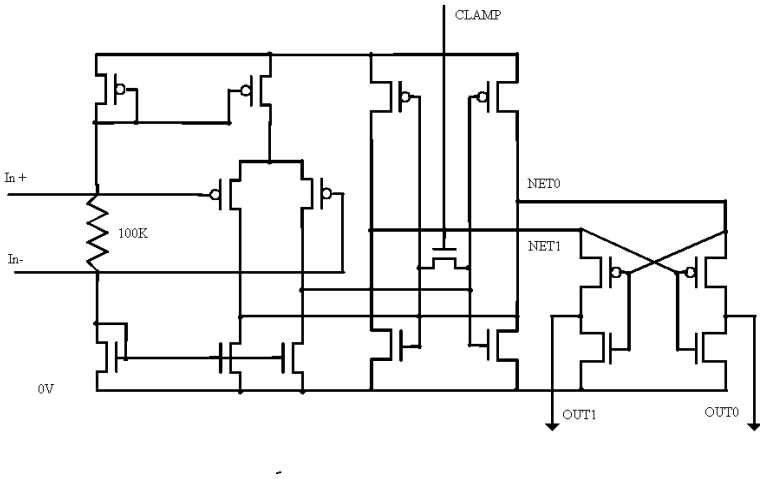


Figure 5: Asynchronous comparator

Using a value of $\tau = 0.3$ nS obtained from measurements of the simulated comparator we can predict 1.34 nS additional time would be needed on average to accommodate the metastability of the worst case decision.

The successive approximation algorithm leads to one comparison with a comparator input (i.e., difference between the analogue and reference voltages) between 0 and $LSB/2$, one between $LSB/2$ and 3 $LSB/2$, one between 3 $LSB/2$ and 7 $LSB/2$ etc. With other delays in the loop timing equal to 7.35 nS, Table 2 gives the predicted time for the 4 bit converter to complete each bit comparison

3.1 Bundled data design

We have constructed an HSPICE model of such an asynchronous converter in which the clock edge of the synchronous design is replaced by a timing signal (C) derived from the comparator output. It is

Time, nS	0 - $LSB/2$	$LSB/2$ - 3 $LSB/2$	3 $LSB/2$ and 7 $LSB/2$	7 $LSB/2$ and 15 $LSB/2$
Average case	8.69	8.22	7.93	7.69
Worst case		8.39	8.06	7.81
Best case		8.06	7.81	7.58

Table 2: Asynchronous converter: predicted bit times

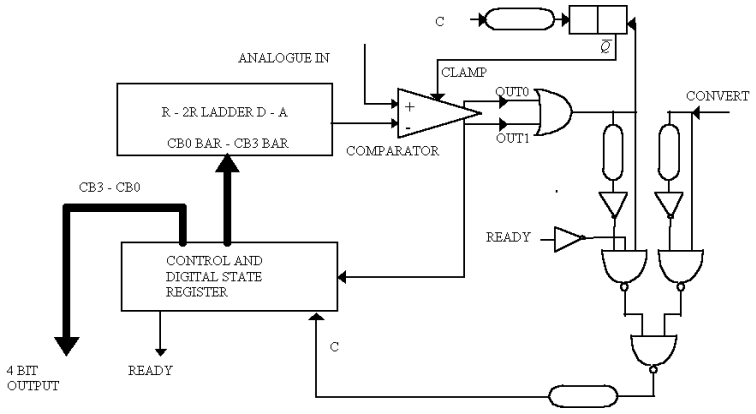


Figure 6: Asynchronous 'bundled data' converter

shown in Figure 5. The CLAMP is driven from the delayed timing signal edge. This delay is fixed at 3.5 nS, allowing for the D-A converter and the first stage of the comparator to complete. The overall 'bundled data' overheads were 7.35 nS, where the additional delay is needed to compensate for the time from comparator output to the resetting of the register bit.

The system operates essentially in a 'bundled data' mode with all state changes linked to the timing signal. Typical waveforms are shown in Figure 7, which clearly shows the increased time of about 15 nS between the first and second rising edge of C, resulting from an analog input very close to the D-A converter's output given by 1000.

Measurements on the time required by the comparator as the difference between the analogue input voltage and the D-A converter output is varied show that the critical bit time follows fairly closely the function:

$$t_m = \tau \ln\left[\frac{V_{high} - V_{low}}{V_m}\right] + t_f,$$

where τ= 0.3 nS, t_f = 7.35 nS (the 'bundled data' overheads), and $V_{high} - V_{low}$ = 5 V.

This is shown in Figure 8, where the probability of error is assumed to be determined, as before, by:

$$P = \frac{2(V_{high} - V_{low})}{LSB} e^{\frac{-t_m}{\tau}}.$$

The HSPICE modelling of this system gives an average of 32.5 nS for the four bits of a conversion, excluding initialisation of the control loop and outputting the result. The only inaccuracies are now in the quantisation of the input signal and are confined to the least significant bit. The conversion time

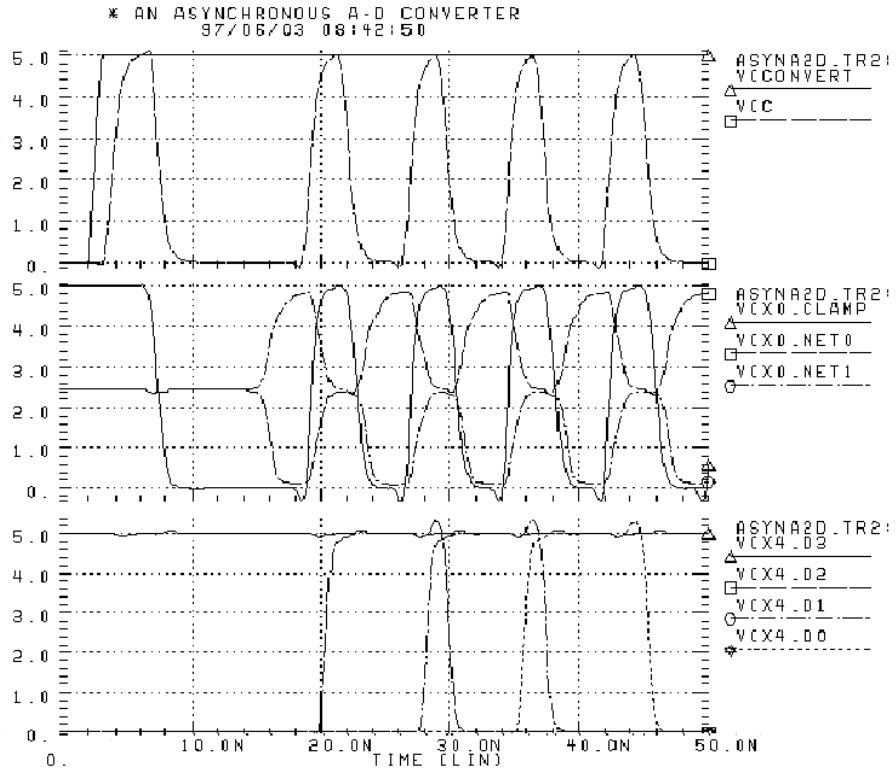


Figure 7: Bundled data design waveforms

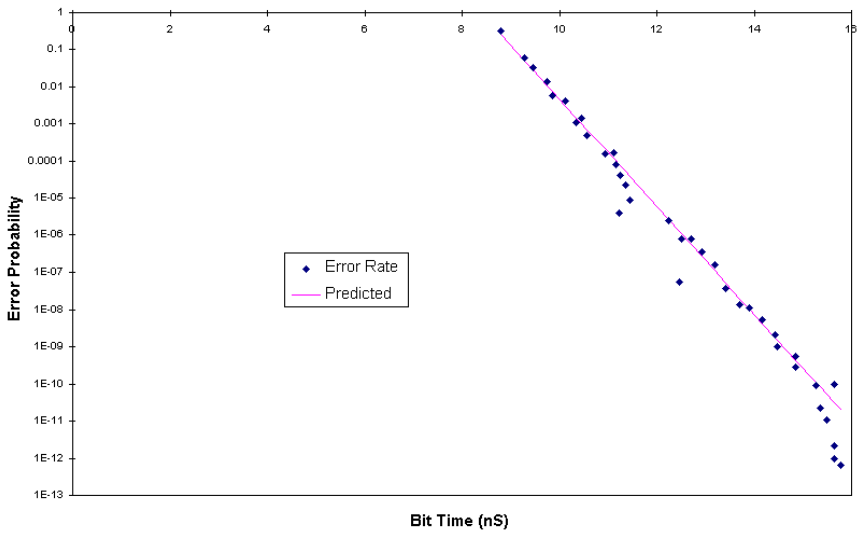


Figure 8: Asynchronous comparator response (rate of failure to compare within a given bit time)

compares favourably with the clocked design, where an error rate of 10^{-12}, which can occur in *any* bit, can only be achieved with bit times of 12 nS, or 48 nS total for a four-bit conversion.

3.2 Speed-independent design

An alternative design has been obtained using formal techniques and software [8] available for synthesis of speed-independent circuits. This design uses a more conservative approach to gate delays than the 'bundled data' design. Every logic gate in the circuit that implements the successive approximation algorithm is assumed to have an arbitrary finite delay. Delay independence guarantees greater reliability of the circuit as it: (a) becomes more robust to the parameter variations in the CMOS technology, and (b) allows a design to be implemented without a change on a range of different CMOS technologies.

The overall structure for our speed-independent design is shown in Figure 9. It consists of an asynchronous comparator and D-A converter, identical to those used in the 'bundled data' design, and four parts of the speed-independent control logic. The latter include:

- A buffer for intermediate storage of the (one-bit) result of comparison $D, \hat{D}$; both the inputs $D, \hat{D}$ and outputs e_0, e_1 of the buffer are represented in a dual-rail code with spacer 00. The completion logic of the buffer produces a CLAMP signal and a signal e_a used by the scheduler in determining when to "shift" to the next step in successive approximation.

- A register for accumulating the four-bit digital code of the conversion result. The input of the register consists of the two-rail encoded and buffered (e_0, e_1) value of the comparison result as well as five control signals $L_i, i = 4, ..., 0$, which, respectively, encode (one-hot) the following five commands: "Store the initial code 1000", "Store Comparator result in Bit 3, and set Bit 2 to high", "Store Comparator result in Bit 2, and set Bit 1 to high", "Store Comparator result in Bit 1, and set Bit 0 to high", "Store Comparator result in Bit 0". Thus, the register's data input is multiplexed. The output of the register consists of the four-bit (dual-rail) digital result $(y_i, \check{y}_i)$ and a set of four bitwise pre-completion signals (b_i).

- Completion logic, to produce the global completion signal b from the pre-completion ones and the command signals.

- A scheduler, which controls the execution of the successive approximation algorithm. In doing so, it produces the appropriate controls to the register and signals to the environment with the READY signal going low (meaning that the converter is initialised with the code 1000) and high (for the completion of the conversion). Those signals are produced at appropriate times from the events on signals START (from the environment) and e_a (from the buffer).

The speed-independent circuit synthesis method we use here is based on the initial specification of the control logic behaviour in terms of a Signal Transition Graph (STG) [1]. The latter is a specialised form of a Petri net [11] whose event nodes are labelled with edges of signals in the specified behaviour. The STG is therefore essentially a formal capture of a timing digram that is used traditionally in digital design. Figure 10 shows an STG that is more condensed (abstract) than the actual model used for synthesis of the above-mentioned elements of the converter. It is possible to see in this model the idea of pipelined action. For example, the Compare action (D:=Comp) is done in parallel with the resetting of the register (only, $b+$, the completion action is shown); the storing in the register (e.g., Reg:=D3.D2.1.0) is in parallel with the actual clamping of the comparator into spacer (D:=00).

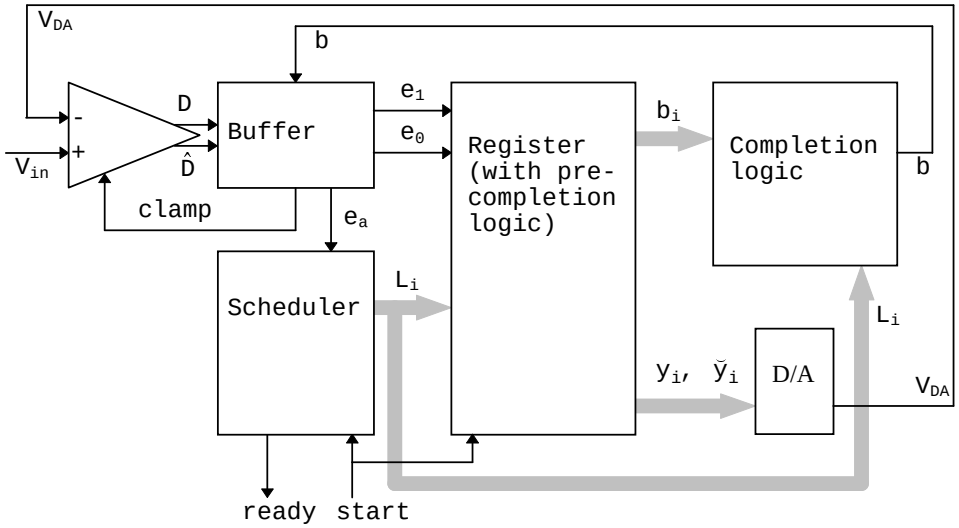


Figure 9: Speed-independent design (overall system structure)

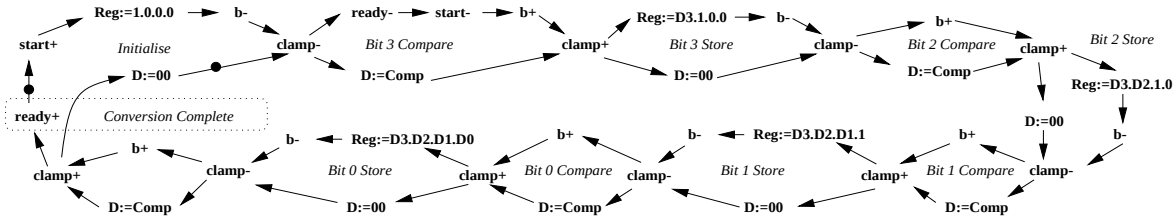
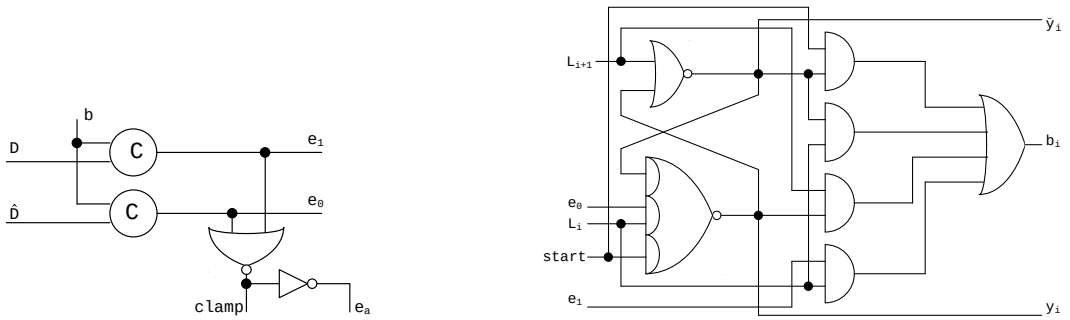


Figure 10: A condensed STG model for speed-independent design

The STG model of the complete circuit would be too large to show in this paper. Furthermore, to obtain logic using petrify, we built separate STGs for the register and the rest of the structure. In those STGs we hid the events of the missing part, and thus avoid excessive complexity. One of such STGs and the output from petrify are presented in the appendix. The various elements of the circuit are shown in Figures 11 (a) and (b), 12 (a) and (b), and and 13.

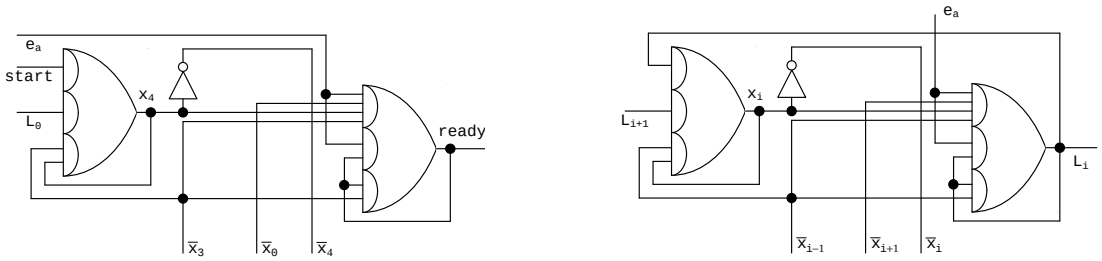
This design does not completely avoid delay dependence since both the D-A converter and the input layer of the comparator are not speed-independent. We believe (after extensive HSPICE experiments) that the delay in the completion logic block and in the buffer is sufficiently large (at least 10 nS) to allow those delay-dependent parts to complete in coherence with the corresponding change of CLAMP.

HSPICE modelling results for the speed-independent design are shown in Figures 14 and 15. The first figure illustrates two examples of comparator response to the falling edge of CLAMP, one with metastability (lasting nearly 4 nS), when the input analogue voltage is near to the nearest LSB line, and the other without. The second waveform shows how the most critical groups of signals behave during conversion. Similar to the bundled data design, we used the same example – converting the value whose code is close to the mid-range point (between 0111 and 1000). A one-bit conversion cycle is about 45 nS, which is about five times that of the bundled data design. The critical cycle is obviously via the following components: buffer, scheduler, register and completion logic. Note that the circuit operates with a four-phase self-timed protocol. There is a slight difference between the phases, where the phases during which the value is stored in the register (when both the register latches and the



(a) (b)

Figure 11: Speed-independent design: (a) buffer circuit, and (b) register and pre-completion logic (i-th bit, i=1,2)



(a) (b)

Figure 12: Speed-independent design: scheduler; (a) initialisation control element (producing L_4 = READY), and (b) i-th bit control element (for $L_i, i = 0, 1, 2, 3$)

completion logic are working) are 3-5 nS longer than the phases when only the completion logic is in action. Thus, the time 20 nS (for one of those phase pairs) is approximately divided between the elements of the circuit in the following way: buffer 3 nS, scheduler 6 nS, pre-completion logic 5 nS and completion logic 6 nS.

3.3 Exploring other speed-independent design options

In addition to the above-mentioned design (SI-1) we also produced two other speed-independent designs with slightly different approach to synthesis of control logic. One of them (SI-2) used a more canonical type of register consisting of a row of four self-timed master-slave flip-flops that was designed using the idea from [12], Fig.4.11 (b)). The SI-2 design also included a completion circuit, a command scheduler (similar to the one above) and an additional synchronisation logic. The use of a master-slave register allowed us to avoid putting an additional buffer between the comparator and register, but at the cost of extra size (the full row of masters). The synchronisation logic we used also provided a more conventional handshaking with the environment – the READY signal would only go high after the entire conversion cycle (with the initialisation phase being part thereof). As a result, the SI-2 design

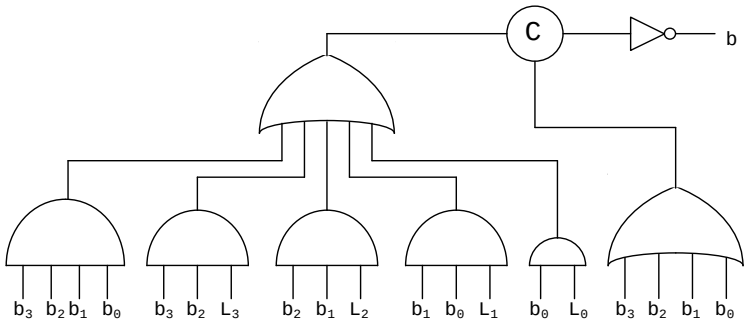


Figure 13: Speed-independent design: completion logic

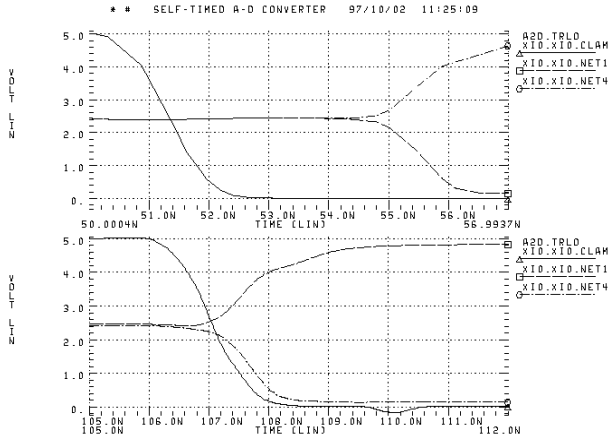


Figure 14: Comparator response with and without metastability

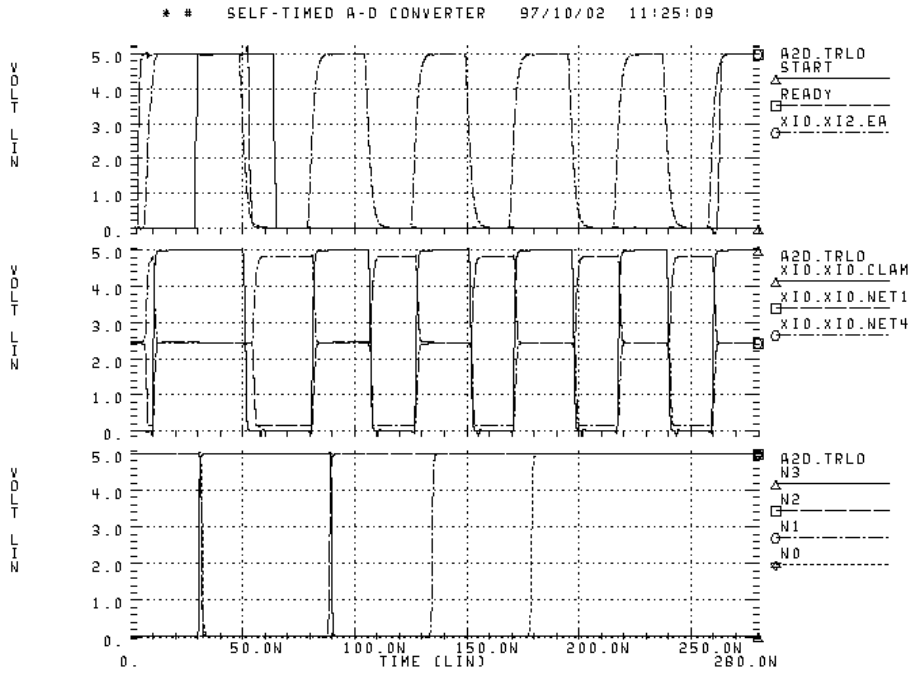


Figure 15: Speed-independent design: conversion waveforms

was more modular, and the parts of logic that had to be synthesised with petrify were simpler. The payment for these benefits was overall slower operation, with the one-bit conversion cycle being about 60 nS.

The third design (SI-3) was on the other hand a more specialised than SI-1. For example, bit-selection was part of the functionality of the register, thus eliminating the need for a separate scheduler. The advantages of this approach, where the register acted as a multiplexing state machine, were less circuit size and overall faster operation. The one-bit cycle could be reduced down to 30 nS (reducing it further to the level of that of the "bundled data" design would require eliminating completion logic and putting more stringent conditions on relative delays). The main shortcomings of such a less structured design were: (i) that synthesis with petrify was at its complexity limit (total number of signals in the STGs up to 40); (ii) that the design of the register logic was no longer scalable (from four bit to n-bit converter).

We thus believe that among those three speed-independent options the SI-1 design would be optimal in modularity, speed and efficiency. It should also be relatively easy to test due to its modularity and self-checking properties of speed-independent circuits.

4 Comparison of synchronous and asynchronous A-D converters

Figure 4 shows that the synchronous converter can be clocked with a bit time as low as 5 nS if error rates of 10^{-4} are acceptable. The design of the bundled data converter is similar, with the exception of the comparator circuit, and the timing signal generation, yet it requires at least 8 nS to make a comparison. This is because the bistable circuit used is not active until the clamp signal goes low, and this delay is fixed at 3.5 nS after the register change. Additionally, a time of 3.85 nS is required from the time metastability is resolved in the bistable to generate the next clock. In the synchronous system, the only timing signal is the clock, and therefore register changes are communicated to the D-A converter, and hence the comparator, immediately. Similarly, the next clock edge is not determined by the comparator output.

If the converter is used in a real-time application, where a control loop or some other system is critically dependent on the receipt of accurate data within a given time, the unbounded conversion time of the asynchronous system may cause errors even though the data is accurate. We have simply exchanged uncertainty in data for uncertainty in time.

An advantageous compromise can be achieved by terminating the conversion process at a fixed time, and accepting the current value of the output register whether metastability exists or not. Thus if 42 nS is allowed for the four-bit comparisons, a worst case time for the fastest three bits is 24.26 nS, leaving 17.74 nS for the worst bit. This would give a probability of error of less than 10^{-12}, which can only be matched by a synchronous converter with a time of 48 nS for the four comparisons. Over the range $42 - 47$ nS, the bounded asynchronous design is both faster and more reliable than a synchronous converter because the clock rate does not need to be fixed at the time for the slowest comparison. In converters with 8 or 12 bits, the improvements will be greater.

Another advantage of bounded asynchronous timing is that errors are less likely to be severe. The probability of error in a particular bit, and no higher bit, in a successive approximation synchronous converter, is the same as the probability of the comparison at that time being longer than t_b and not

earlier. This depends on the probability of the analog input being closer than V_m to the D-A output, where $V_m = (V_{high} - V_{low})e^{\frac{-t_b}{\tau}}$ as before. For the least significant bit, that probability is $P = \frac{V_m}{LSB}$, and for bit n it is $P = \frac{2^{-n}V_m}{LSB}$ [1].

The probability of the n-th bit comparison taking longer than t_b is therefore:

$$P = \frac{2^{-n}(V_{high} - V_{low})}{LSB} e^{\frac{-t_b}{\tau}}.$$

If we assume that only one bit comparison in an asynchronous converter can cause metastability, and that the others take a time t_b, the most significant bit can take the full conversion time, say, $4t_b$, before being terminated, the second $3t_b$, so that the probability of error in the high order bits where n is large is much less than later bits. In general the probability of error in a particular bit will now be:

$$P = \frac{2^{-n}(V_{high} - V_{low})}{LSB} e^{\frac{-(n+1)t_b}{\tau}}.$$

this gives a probability of error in the most significant bit of less than 10^{-47} for the four bit bounded time asynchronous converter compared with a figure of about 10^{-20} for the asynchronous converter.

Obviously, comparison of a speed-independent converter, which is significantly slower than the bundled data one, would be more difficult to perform in numerical terms. On the other hand, speed-independent designs should not be considered only out of theoretical interest if one had to take into account other risk factors besides metastability, such as delay variations of CMOS submicron technology and the need to maintain portability of the design. This however takes us beyond the scope of the paper.

5 Conclusions

We have shown that analogue to digital converters with a fixed conversion time are subject to errors due to metastability. These errors will occur in all converter designs with a bounded time for decisions, and are potentially severe. It is also clear that an asynchronous design can be more reliable, and faster, on average, than a synchronous design but its completion time may be infinite. Two types of asynchronous converters have been described, the design of the first follows a 'bundled data' approach in which there is a single timing path whose delays are matched with the data path delays, and the second is a totally self-timed (only the D-A converter part is assumed to have bounded delay, quite sufficient in practice) system, which is insensitive to variations in individual component delays. It is clear from the first design that a reliable asynchronous conversion could be completed in under the 10 nS per bit required by the synchronous design with an error rate of 10^{-11}. The fully self-timed design can be more portable and robust because it does not assume that gate delays on the same chip will have similar delays, but has at least twice as many transistors and typically takes three-four times longer period for a conversion than the bundled data one. Such a significant slow-down is caused mainly by the requirement of explicit acknowledgement of signal tranistions for speed-independent operation, and as a result, use of additional hardware (completion logic). It is interesting to note that while the asynchronous designs will not produce an error of more than one bit in the result, the conversion time

[1] The integral value of this probability through all bits can be estimated as the sum of the series of $P = \frac{2^{-i}V_m}{LSB}, i = 0, ..., n$, which tends to $P = \frac{2V_m}{LSB}$ as n goes to infinity. This last formula was used in Section 2.3.

is not predictable, and this may cause difficulties in a real time system. This problem is linked to the process of conversion between reals and integers, and the limited gain - bandwidth of all physical devices used as comparators.

References

[1] T.-A. CHU and L.A. GLASSER, "Synthesis of Self-Timed Control Circuits from Graphs: An Example", *IEEE International Conference on Computer Design (ICCD86)*, 1986, pp.565 – 571.

[2] T.J.CHANEY and C.E.MOLNAR, "Anomalous behavior of synchronizer and arbiter circuits", *IEEE Transactions on Computers*, C-22(4), pp. 412–422, April 1973.

[3] D.J.KINNIMENT and J.V.WOODS, "Synchronisation and arbitration circuits in digital systems", *Proceedings IEE*, vol 123, No. 10, pp. 961–966, October 1976.

[4] B. RUSSELL, *Introduction to Mathematical Philosophy*, London, G.Allen and Unwin, 1920 (pp. 84 – 86).

[5] M. MENDLER and T. STROUP, "Newtonian arbiters cannot be proven correct", *Formal Methods in System Design*, 3(3), pp. 233-257, December 1993.

[6] I. M. MITCHELL and M.R. GREENSTREET, "Proving Newtonian Arbiters Correct, almost surely", *Proc. of Conf. on Designing Correct Circuits (DCC'96)*, 1996 (see also: IAN M. MITCHELL, "Proving Newtonian Arbiters Correct, Almost Surely", MSc Thesis, University of British Columbia, October 1996).

[7] C. L. SEITZ, "System Timing"; In *C. Mead and L. Conway, Introduction to VLSI Systems*, Chapter 7, pp.218–262, Addison Wesley 1979.

[8] J. CORTADELLA, A. KONDRATYEV, M. KISHINEVSKY, L. LAVAGNO and A. YAKOVLEV, Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers, *Proc. 11th Conference on Design of Integrated Circuits and Systems (DCIS'96)*, Barcelona, Nov. 1996, pp.205-210.

[9] H.J.M. VEENDRICK, "The behaviour of flip-flops used as synchronizers and prediction of their failure rate", *IEEE Journal of Solid-State Circuits*, vol. SC-15(2), pp. 169–176, April 1980.

[10] D.J. KINNIMENT, A.V. YAKOVLEV, and B. GAO, "Metastable Behaviour in Arbiter Circuits", Technical Report Number: 604, Dept. of Computing Science, University of Newcastle upon Tyne 1997.

[11] T. MURATA, "Petri nets: properties, analysis and applications", *Proceedings of IEEE*, vol. 77(4), pp. 541–580, April 1989.

[12] V. VARSHAVSKY, M. KISHINEVSKY, V. MARAKHOVSKY, V. PESCHANSKY, L. ROSENBLUM, A. TAUBIN, and B. TSIRLIN, *Self-Timed Control of Concurrent Processes*, Kluwer Academic Publishers, Dordrecht, 1990 (Russian Edition: Nauka, Moscow, 1986).

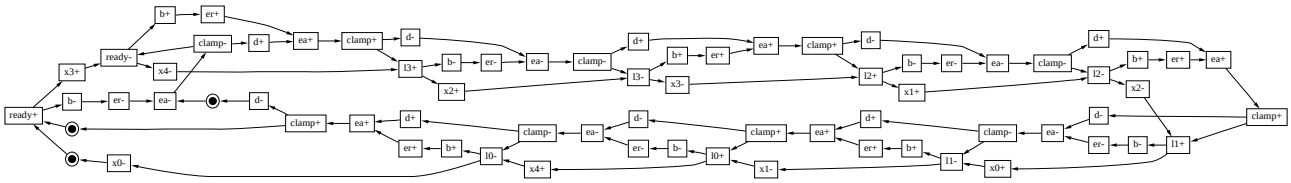


Figure 16: STG specification for scheduler

Appendix: Signal Transition Graphs and synthesis results from Petrify [2]

The STG used to synthesise the schedule logic is shown in Figure 16. It was built automatically (by tool draw.astg) from the designer's textual specification. This STG hides the details of the buffer, register and completion logic – one can observe that only the completion signals b, from the register, and e_a, from the buffer, are used. Signals x_i are used to resolve coding conflicts and act as internal memory of the scheduler. Their transitions are organised in parallel with the main events in order to minimise the critical cycle delay.

The output from petrify is shown below:

```
# EQN file for model AD_control_buffer_top1
# Generated by petrify 3.3 (compiled 6-Sep-97 at 9:33 AM)
# Outputs between brackets "[out]" indicate a feedback to input "out"
# Estimated area = 68.00

INORDER = ready l3 l2 l1 l0 b clamp er ea d x1 x2 x3 x4 x0;
OUTORDER = [ready] [l3] [l2] [l1] [l0] [b] [clamp] [er] [ea]
[d] [x1] [x2] [x3] [x4] [x0];
[ready] = x3' (clamp x4 x0' + ready) + ready clamp;
[l3] = x2' (clamp x3 x4' + l3) + l3 clamp;
[l2] = x1' (clamp x3' x2 + l2) + l2 clamp;
[l1] = clamp x1 x2' x0' + l1 (x0' + clamp);
[l0] = clamp x1' x4' x0 + l0 (x4' + clamp);
[b] = l3' ready' l0' l1' l2';
[clamp] = ea;
[er] = b;
[ea] = er (ea + d) + ea d;
[d] = clamp';
[x1] = x0' x1 + l2 + l1;
[x2] = x2 x1' + l2 + l3;
[x3] = x3 x2' + ready + l3;
[x4] = x3' x4 + l0 + ready;
[x0] = x4' x0 + l1 + l0;
```

Another example is the STG, shown in Fig 17, specifying the register together with its pre-completion logic. The details of the surrounding logic, such as scheduler, are again hidden inside the handshakes between the completion signals b, outputs of the buffer ($e0, e1$) and controls $ready, l3, ..., l0$.

The output from petrify is shown below:

[2] included to help reviewing; can be omited in the final version.

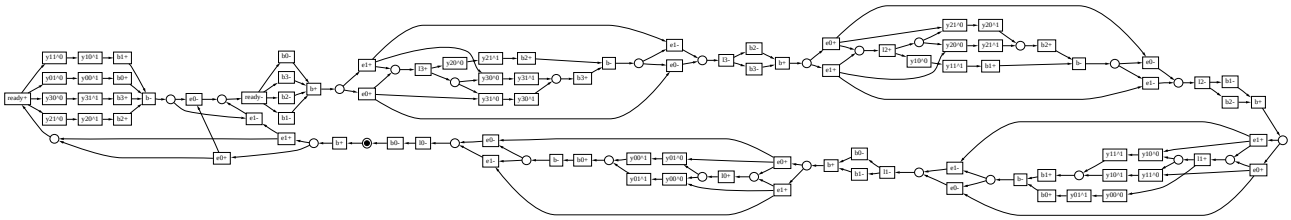


Figure 17: STG specification for register

```
# EQN file for model AD_control_buffer_latch1
# Generated by petrify 3.3 (compiled 6-Sep-97 at 9:33 AM)
# Outputs between brackets "[out]" indicate a feedback to input "out"
# Estimated area = 68.00

INORDER = e0 e1 ready l3 l2 l1 l0 b0 b1 b2 b3 b y30 y31 y20 y21 y10 y11 y00 y01;
OUTORDER = [b0] [b1] [b2] [b3] [b] [y30] [y31] [y20] [y21] [y10] [y11] [y00]
[y01];
[b0] = l0 (e1 + y00 + b0) + l1 y01 + ready y00;
[b1] = l1 (b1 + y10 + e1) + l2 y11 + ready y10;
[b2] = l2 (y20 + b2 + e1) + l3 y21 + ready y20;
[b3] = l3 (e1 + b3 + y30) + ready y31;
[b] = l0' b (l1' b2' + l3' b1') + b3' (l3 + ready) + b0' (b3' b2' b1' +
ready + l1);
[y30] = y31' ready';
[y31] = y30' (l3' + e0');
[y20] = y21' l3';
[y21] = ready' y20' (e0' + l2');
[y10] = l2' y11';
[y11] = ready' y10' (l1' + e0');
[y00] = y01' l1';
[y01] = ready' y00' (l0' + e0');
```