Asynchronous Clocks

Simon Moore
University of Cambridge

Abstract. Asynchronous circuits typically operate in a clock-free manner. That said, low-level timing characteristics like equipotential regions and matched delays are often employed in self- timed circuits, a class of asynchronous circuits. This paper takes this a step further and reviews approaches to generating clocks inspired by asynchronous circuits, from frequency distribution using Muller C-element chains through to pausible clocks and asynchronously oscillating grids.

1 Introduction

After many years of discussion with Professor Alex Yakovlev and Professor David Kinniment in Newcastle, and other members of the asynchronous circuits community, I am fortunate to have gained a deeper understanding timing in circuits. With that understanding brings enlightenment but not always back-and-white clarity. The question of what is a clock and what is not a clock is a grey area when one looks closely. Proponents of asynchronous (or *self-timed*) circuits believe that clocks are an evil and that clock-less circuits have many virtues. This paper reviews the heretical approach of using asynchronous (clock-less) circuits to generate clocks, and how the boundary between asynchrony and synchrony can be blurred.

2 Clocking basics

In its simplest form, a clock for a digital circuit comes from a precision timing source like a quarts crystal. The precision timing source is then distributed across a chip, to the clock inputs of components like the D flip-flop (DFF). The DFFs provide storage of state and also control the rate of data propagation by delaying data output until the next clock edge. Thus, data is advanced on the clock edge. To provide the illusion that all state updates happen simultaneously (so called *synchronous digital circuits* or *clocked digital circuits*), the clocks to each DFF are expected to arrive simultaneously (synchronously). This provides the illusion of discrete (digital) time to go with the discrete (digital) signal levels. In practise, a truly synchronous clock does not exist. Instead we must be satisfied with a close approximation that, within tolerances (e.g. setup and hold times of the DFFs), provides an accurate enough implementation of the desired synchronous abstraction, which arguably makes the circuit designer's life easier.

3 Asynchronous clock source

Many clocked circuit designers like to provide an external clock locked to a highly stable quartz crystal. It is ironic that the performance of their circuits will vary with temperature, so the clock frequency has to be set against the worst case path delay between synchronising elements at the worst case temperature. Much performance is, therefore, thrown on the floor when the circuits are operating at more typically temperatures. But this approach does preserve the digital time abstraction.

External crystals typically operate at a much lower rate than the desired clock frequency. A phase locked loop is often used to multiply this lower frequency stable clock up to a higher frequency on-chip clock.

An alternative clock generation strategy is to use a delay-line. Some low-cost microcontrollers simply use an inverter ring to provide a clock frequency. Typically the resulting clock frequency varies significantly between devices and with device temperature. We investigated the possibility of constructing a tuneable delay-line that can be self-calibrated from a low-frequency and power-efficient watch crystal [7]. An overview is presented in Figure 1 with details of the delay-line cell in Figure 2.

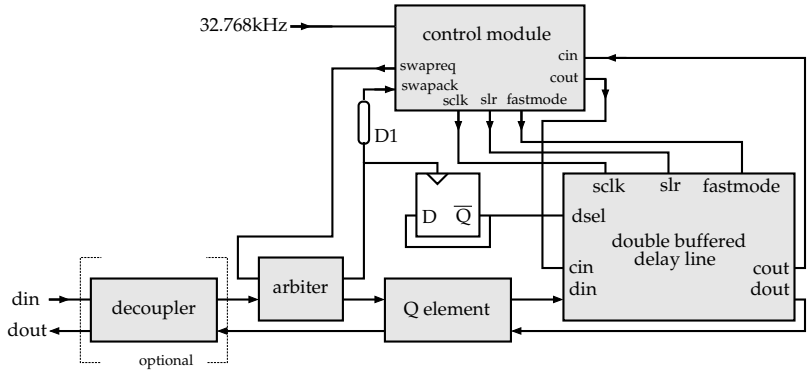


Fig. 1. Overview of an asynchronous self-calibrating delay-line (from [7])

Critical to the functional behaviour is the Q-element [5] used to send both rising and falling events through the delay line before acknowledging dout. The Q-element ensures that the arbiter is not released until the delay line has been through both rising and falling edge phases. Analysis of the behaviour of our Q-element implementation (Figure 4) was undertaken using signal transition graphs (STGs) [8, 9], a form of Petri net, with assistance from the Petrify tool [2]. Professor Yakovlev was pivotal in establishing STGs and the creation of the Petrify tool.

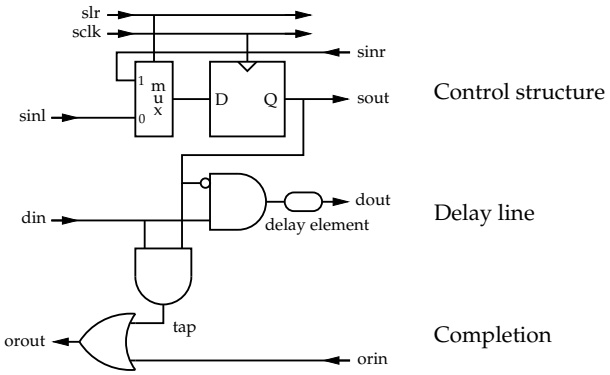


Fig. 2. Asynchronous delay-line cell (from [7])

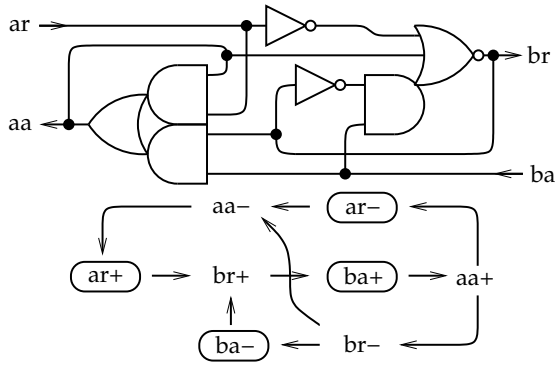


Fig. 3. Asynchronous decoupler with behaviour as an STG (from [7])

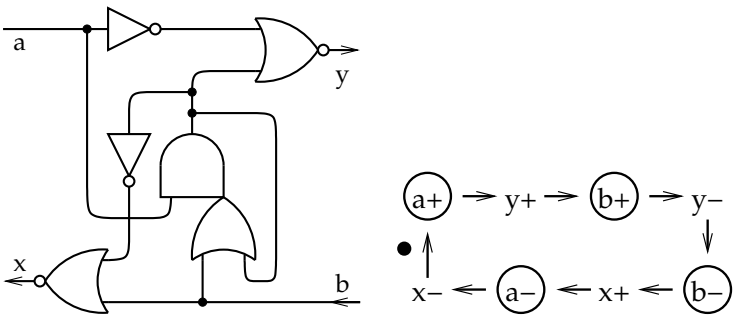


Fig. 4. Asynchronous Q-element with behaviour as an STG (from [7])

4 Asynchronous clock distribution

Clock distribution is the art of broadcasting a clock across a chip so that its frequency and phase appear identical at every clocked element (e.g. DFF). Frequency distribution, in contrast is rather easier. One could, for example, construct a long chain of inverters (Figure 5a) and arrange them in a serpentine manner over the surface of the chip. This would (almost) manage to broadcast the frequency. I say "almost" because a pulse proceeding down an inverter chain will undergo pulse shrinkage, so it is unlikely to reach the end of a long chain. On the other hand, an asynchronous micropipeline made of Muller-C elements (Figure 5b) will successfully distribute the frequency and will guarantee that pulse shrinkage will never obliterate a pulse as it carefully copies the pulse (or *token*) to the next Muller C-element before destroying the source.

A conventional clock distribution approach uses a H-tree fractal over the surface of the chip. This works quite well, though still presents potential discontinuities in clock phase (e.g. see nodes A and B in Figure 6 which are clocked from different branches but are physically adjacent). Self-calibration in the tree can help. Also, sometimes a grid is used at the lowest level to crowbar the H-tree leaves together.

Rather than drive a grid from a H-tree, Dr Scott Fairbanks and I investigated the use of a micropipeline structure laid out as a grid to form a self-oscillating clock distribution system (both frequency and phase) [3]. This originated from earlier work on the one-dimensional asP micropipline structure [1] (see Figure 7a) and was evolved into a two-dimensional structure (Figure 7c). The grid inputs are mixed using the circuit in Figure 8. Pull-up and pull-down nodes are alternated across a grid. Pull-up nodes use the mixer to identify when the majority of inputs are low and then pulls high. Pull-down nodes do the inverse. Thus the grid oscillates in unison and measurements indicate very low skew even in the presence of device variability.

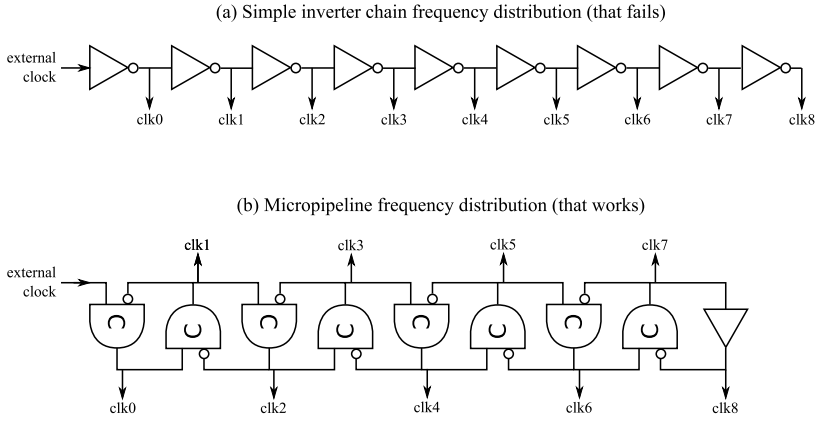


Fig. 5. (a) inverter and (b) micropipeline clock frequency distribution

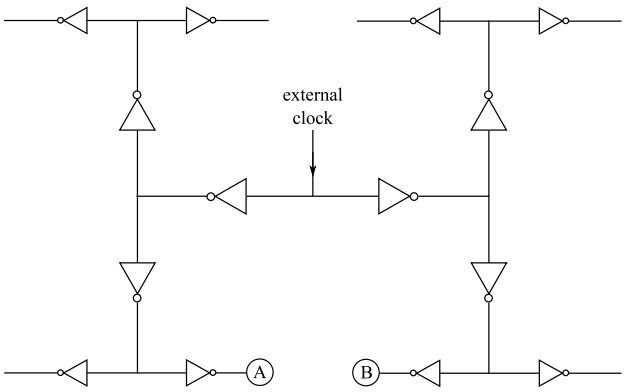


Fig. 6. Simplified H-tree clock distribution

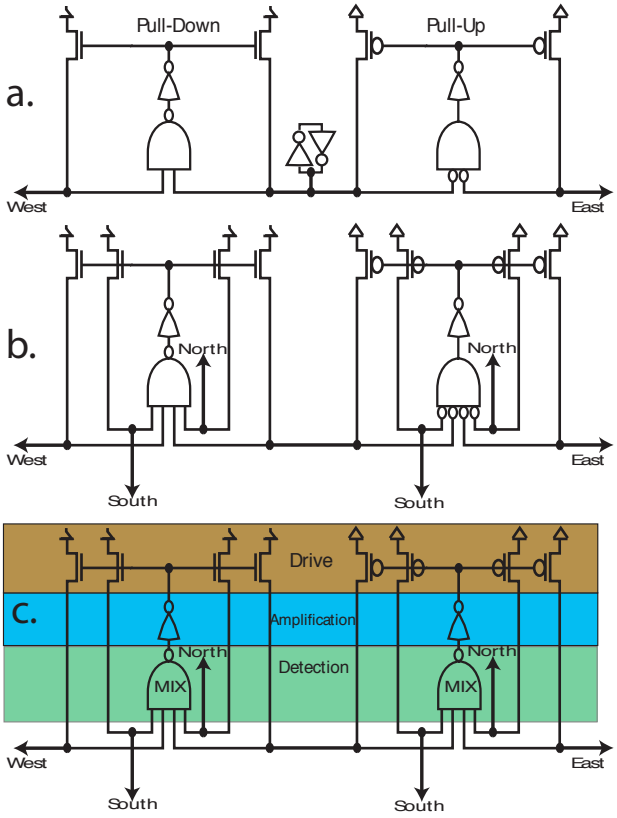


Fig. 7. Evolution from dynamic asP to a distributed clock generator (from [3])

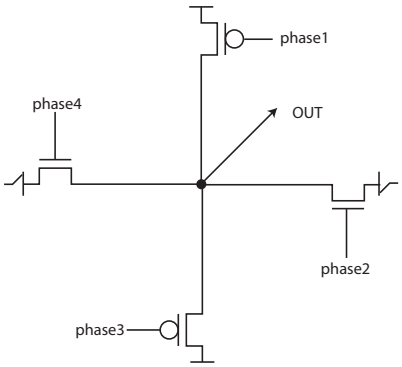


Fig. 8. Clock mixer for the distributed clock generator (from [3])

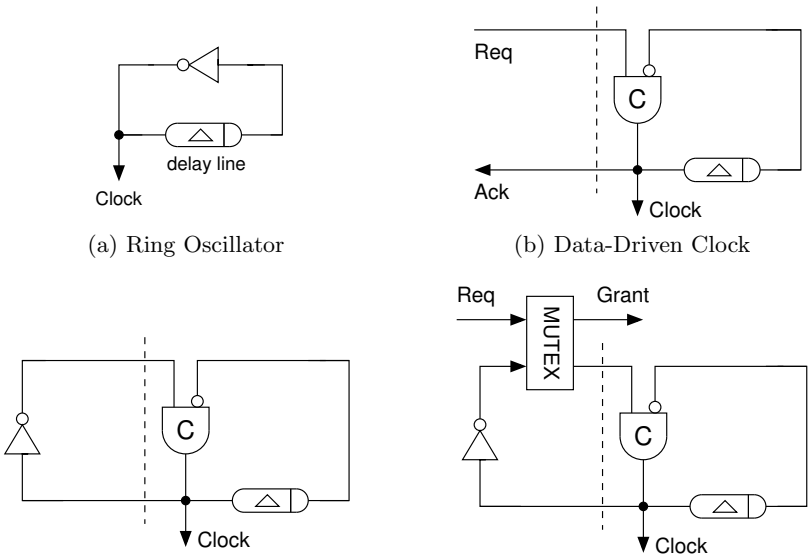
5 Globally asynchronous but locally synchronous circuits

Given that global synchronisation is difficult to achieve, one option is to build chips which are globally asynchronous but locally synchronous (GALS). Since local synchrony is easier to achieve, it allows the clock (synchronous) design method to be used in the small (e.g. a processor core) with asynchronous interconnect between these clocked islands. Global frequency distribution might still be used to control the rate of transfer of information between blocks, making it easy to use credit-based flow control.

Moving data between synchronous domains is not without its problems, however. Sampling a "data ready" bit or some other flow control information coming from another clock domain is likely to result in metastability in the sampling flip-flop. Using a two-flop synchroniser is one approach and with careful design it is possible to reduce the mean-time between failure (MTBF) to once in the lifetime of the universe [4]. However, with incorrect design, or device variability reducing the performance of the sampling flip-flop, the MTBF can easily become less than a minute.

In order to avoid metastability altogether, it is possible to use pausible clocks to ensure completely safe data transfer. Dr Robert Mullins and I undertook a great deal of work in this area with the key final paper being *Demystifying Data-Driven and Pausible Clocking Schemes* [6]. Dr Robert Mullins and I were delighted to collaborate with Professor David Kinniment and Professor Alex Yakovlev on the book *Synchronization and Arbitration in Digital Systems* [4] with several circuits from [6] being reproduced.

Pausible clocks are based around the use of a delay line clock source (Figure 9a) that can be transformed into a data driven clock (Figure 9b) where a local clock signal is produced whenever there is new input data. This is, however, rather restrictive since one typically requires that the local clock continues to oscillate regardless of whether there is new data or not. To this end, the circuit in Figure 9d (an evolution from the circuits in Figure 9a–c) can be used so that



(a) Ring Oscillator (b) Data-Driven Clock

(c) Ring Oscillator (II) (d) Pausible Clock

Fig. 9. Pausible and Data-Driven Local Clocks (from [6])

the clock is only paused to safely transfer new data. Using this basic concept, a complete GALS system can be produced (see Figure 10). For further details, see [6].

Conclusions

Just as digital circuits abstract the analog world into discrete ones and zeros, clocked synchronous circuits abstract continuous time into discrete ticks. In much the same way that it can be useful to analyse digital circuits in their true analog form, it can also be helpful to analyse the true asynchronous (or analog-time) behaviour using techniques like STGs that Professor Yakovlev has been pivotal in creating. Moreover, the ability to mix clocked and asynchronous circuits enables a broader range of design tradeoffs. As we face challenges in clock distribution and device variability for future CMOS circuits, asynchronous techniques may well become critical to meet design requirements. Finally, it should be noted that we can use asynchronous techniques to control and generate clocks, blending synchrony with asynchrony.

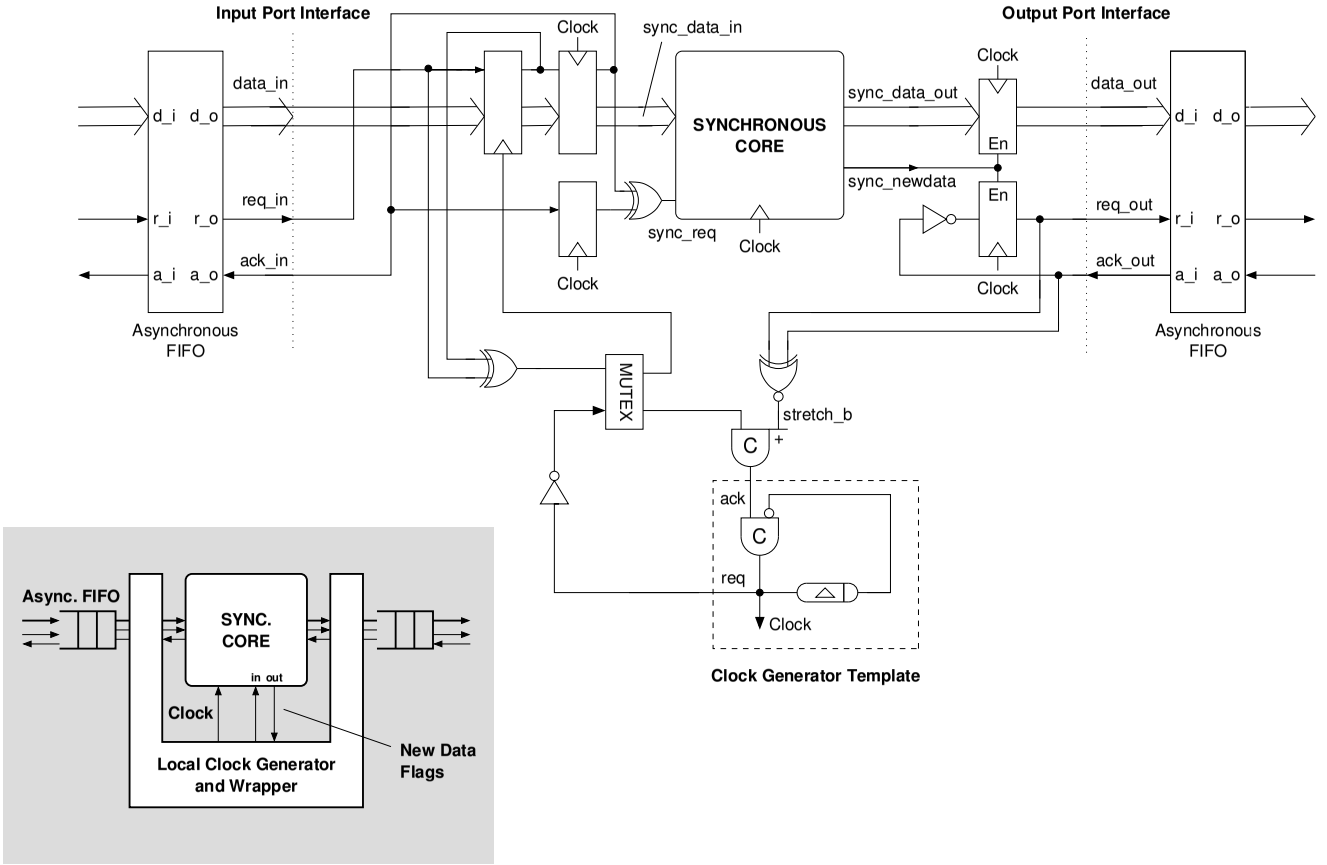


Fig. 10. A locally-clocked synchronous block with a pausible-clock input port and registered/stretchable-clock output port (from [6])

References

1. Control structure for a high- speed asynchronous pipeline (1999)
2. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. IEICE Transactions on information and Systems 80(3), 315–325 (1997)
3. Fairbanks, S., Moore, S.W.: Self-timed circuitry for global clocking. In: 11th IEEE International Symposium on Asynchronous Circuits and Systems. pp. 86–96 (March 2005)
4. Kinniment, D.J.: John Wiley & Sons, Ltd (2008)
5. Martin, A.J.: Synthesis of asynchronous VLSI circuits. In: Straunstrup, J. (ed.) Formal Methods for VLSI Design, chap. 6, pp. 237–283. North-Holland (1990)
6. Mullins, R., Moore, S.: Demystifying data-driven and pausible clocking schemes. In: 13th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'07). pp. 175–185 (March 2007)
7. Taylor, G., Moore, S., Wilcox, S., Robinson, P.: An on-chip dynamically recalibrated delay line for embedded self-timed systems. In: Advanced Research in Asynchronous Circuits and Systems, 2000. (ASYNC 2000) Proceedings. Sixth International Symposium on. pp. 45–51 (2000)
8. Yakovlev, A., Lavagno, L., Sangiovanni-Vincentelli, A.: A unified signal transition graph model for asynchronous control circuit synthesis. In: Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers., 1992 IEEE/ACM International Conference on. pp. 104–111 (Nov 1992)
9. Yakovlev, A., Lavagno, L., Sangiovanni-Vincentelli, A.: A unified signal transition graph model for asynchronous control circuit synthesis. Formal Methods in System Design 9(3), 139–188 (1996)