Can Metastable States be Trapped?

Ghaith Tarawneh (ghaith.tarawneh@ncl.ac.uk)

Institute of Neurosience, Newcastle University, UK

Abstract. Synchronizers (commonly implemented as chains of two flip-flops) are used to reduce the probability that metastable states, emanating from asynchronous input transitions, propagate to state/data flip-flops and cause unwieldy catastrophes. Another way to think of this is that synchronizers "trap" metastability, preventing it from being observed by others circuits until a failure rate criteria is satisfied. Flip-flop chains are certainly one way of doing it but are there other ways in which metastability can be trapped? More importantly, can a circuit perform useful computations while trapping metastability in the same way that a synchronizer does? This paper will explore the theoretical prospects (and, potentially foolishly, break the long- held taboos) of allowing metastability inside generic Moore machines and finding ways to trap it and manage its effects.

1 Introduction

This paper will explore the theoretical possibility of creating synchronous state machines that meet a given functional specification despite being occasionally metastable. The motivation behind this perusal is to determine whether it is possible (at least in principle) to create multi-clock systems where signals can be transferred across clock domain boundaries without latency. As the reader may well be aware, there is a formal proof (using dynamical system theory) that a fundamental trade-off between latency and reliability is inherent in any Newtonian system that attempts to make a binary decision based on an analogue quantity [1]. It is therefore important to note outright that this paper will not attempt the impossible by trying to remove synchronization altogether. Instead, what will be considered is the more realistic (but perhaps equally questionable) possibility of turning a state machine into a large synchronizer housing many flip-flops and logic gates. To develop the intuition behind this idea, the paper will start by considering a synchronization conundrum in a hypothetical system.

2 The Island

Legend has it that in a time, so long ago, the best philosophers in all parts of the world, tired of wars and petty human affairs, decided to leave the main lands and live in a distant island, known to the rest of the world only as *Synchrona*.[1]

[1] Incidentally, philosophers in that era, known for their modest attire, were colloquially called by common people *flip-flops* in reference to their (often worn out) sandals.

The inhabitants of Synchrona sought to resolve all disorder by establishing strict rules to govern their interactions and day-to-day affairs. One such rule mandated that all philosophers meet each day, precisely at noon, to share any new philosophical insights they have had overnight. This daily routine worked well for the philosophers but was occasionally disturbed by the untimely arrival of small boats from the main lands, carrying messages from common people who sought the philosophers' wisdom. When a boat arrived during a meeting, a very peculiar thing often happened: the philosophers developed conflicting views about what was said during the interrupted meeting. These problematic disagreements were often resolved during the next one or two meetings (although they often led philosophers to reach nonsensical conclusions about certain matters).[2]

Even though disagreements were settled peacefully and had no long-term ill effects on Synchrona, any messenger boats dispatched carrying conflicting replies from the philosophers stirred doubt and greater trouble when received by common people on the main lands. The philosophers were aware of the dangers of propagating disagreements and, realizing that boat arrivals at Synchrona, the interruptions of their meetings and consequently their disagreements could not be prevented, decided to add one more rule to their routine. The new rule said that all discussions preceding the drafting of a reply to common people's questions may not depend on prior discussions on which the philosophers are still in disagreement. Knowing that deciding which discussions involved disagreements was itself a discussion prone to disagreement, the philosophers decided on a most unusual way to enact their new rule. After unloading their cargo, all inbound boats are to be forced into a two-day sail around Synchrona before docking to collect drafted replies and returning to people on the main lands. This way, the philosophers reasoned, any disagreements arising from the arrival of boats would have had enough time to be settled by the time a reply is to be shipped.

Ever since the new rule came into force, the common people stopped receiving the occasional news about the philosophers' disagreements. Rumor circulated that the philosophers devised a most ingenious way to abolish disagreements of all sorts (although the more educated remained skeptical of this). Unusual notes that did not make much sense such as "$1 + 2 = 7$" were still received from the island but all carried the signature of each philosopher known to be living there.

3 The Island as a State Machine

As the philosophers living on Synchrona have argued, metastable states can be "trapped" in a synchronous component if we ensure that value changes on its asynchronous input ports cannot be observed on the outputs within n cycles (where n corresponds to a given MTBF criterion). Synchronizers are the simplest circuits that satisfy this property; it takes n cycles for an input change to be observed at the output of an n-stage synchronizer. Gray counters satisfy

[2] Scientists inhabiting the island noted that the probability such disagreements persisted for t more seconds was reduced by a factor of $e^{t/\tau}$ (where τ depended on how well-fed were the involved philosophers).

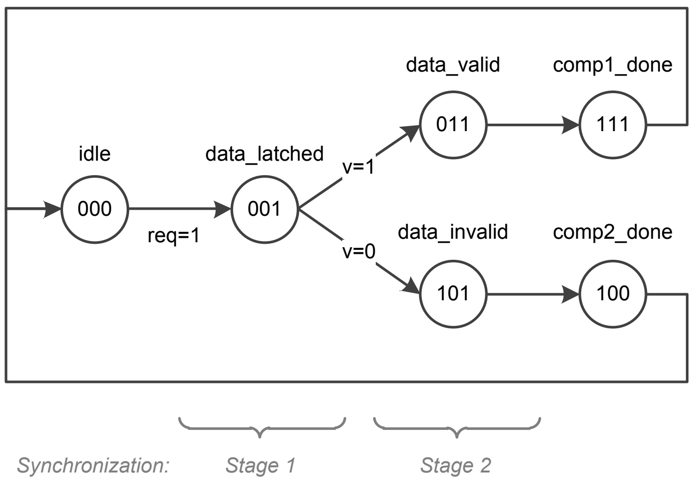

Fig. 1. A state graph of Synchrona

this property too but, like synchronizers, are not typically used for anything beyond counting. In this section we aim to present a toy example of a datapath controller that traps metastability in the same way that a synchronizer does. The controller processes a data item received asynchronously and synchronizes its associated request signal *simultaneously*. Our aim here is not to provide a rigorous methodology for designing such components but to present and argue for the correctness of at least one non-trivial example.

To continue our analogy from the previous section, our datapath controller is assumed to be Synchrona itself. Figure 1 shows a state graph of the island in which three "binary philosophers" (A, B and C) are used to encode its state (with A being the most significant philosopher). The graph has two branches that correspond to a sequence of computations to be performed based on whether the received data satisfies a certain condition v, where v is computed by the datapath after data has been latched. When `req` is asserted, the island is kicked into a series of transitions along one of the two main branches. State codes are chosen such that any two consecutive states within 2 transitions from the idle state differ by only 1 bit. This Gray-based encoding scheme ensures that any metastable state bit in the system will have a *single* sensitized combinational path to a destination state bit. When the island is metastable, therefore, only a single state bit is open to misinterpretation and the island will either transition to the next state or safely roll back to its existing one.

But what about the signal v? If v is generated by the datapath based on a data item that is received asynchronously then isn't v itself vulnerable to becoming metastable? No, the data bundling constraint implies that data bits arrive well before their corresponding `req` transition and so the controller state bits are the single point of failure in the system [2].

So far we have presented arguments that the controller will function correctly (more strictly, the arrival of `req` will trigger a chain of state transitions as per Figure 1, although occasionally the controller will stay in one of the states 000 or 001 for an additional cycle). Can the controller be safely integrated into a larger system without causing metastability failures? Yes if we make it impossible for external observers that monitor the controller's output ports to know when the controller is metastable (just like the philosophers did to hide news of their disagreements from common people). We do this by making the controller's output ports have the same values in the states 000 and 001. In our example we assume the controller has a single output `valid` that is logic high when the controller is in either state 011 or 101 and logic low otherwise (i.e. `valid` $= (A \oplus B) \wedge C$). `valid` will go high once synchronization is complete and is therefore a safe signal for external circuits to sample.

In effect, our controller has two "built-in" synchronizer chains: CB (when $v = 1$) and CA (when $v = 0$). The value of `valid` is asserted once the transition of `req` appears at the second synchronizer flip-flop (either B or A) and so the controller satisfies our criterion for functioning as a synchronizer. In the state graph, entering the state 001 corresponds to the completion of the first stage of synchronization and entering either state 011 or 101 corresponds to the completion of the second.

4 Conclusion

The main point of the informal discussion presented here is to show that there exists more than meets the eye in the design space of metastable synchronous components. Although it is impossible to prevent the occurrence or propagation of metastable states in synchronous modules, our toy datapath controller demonstrates few key things that we still *can* do. First, we can create more generic forms of synchronizer circuits that are neither flip-flop chains nor Gray counters. In these designs, the propagation of metastable states can be restricted to specific chains of flip-flops. Second, these generic synchronizers can be state machines that complete a number of state transitions in sequence. There remains a non-deterministic uncertainty of 1 cycle in how long these transitions take but they are nonetheless guaranteed to be completed *eventually.*

References

1. Michael Mendler and Terry Stroup. Newtonian arbiters cannot be proven correct. *Formal Methods in System Design*, 3(3):233–257, 1993.
2. Ghaith Tarawneh, Alex Yakovlev, and Terrence Mak. Eliminating synchronization latency using sequenced latching. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(2):408–419, 2014.