From Digital Timing Diagrams to Natural Language and Back

Josep Carmona

Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. Digital Timing Diagrams have been and are and effective visualization aid for the understanding of a digital circuit. However, in case of complex circuits, the interplay between signals and the corresponding hidden dependencies may be missed. In this paper we consider the textual representation of digital timing diagrams, as an alternative way of describing a digital circuit. We provide ideas on how to transform a digital timing diagram into a textual description, and the (more challenging) opposite problem: obtaining a digital timing diagram from a textual description.

1 Introduction

I will always remember the first time I met Alex Yakovlev: Newcastle, (very cold) winter of 1998, my (at that time, future) PhD. advisor Jordi Cortadella took me to the ACID workshop to convince me to do a PhD. with him on asynchronous circuits synthesis. The first night I discovered two important things: first, the warm character of Alex and his family, who hosted me in a great dinner, which remarkably included dancing at the end. Since then I have met Alex in several conferences, and have collaborated with his group in different topics (asynchronous circuits, theory of regions, process mining). And the second thing I discovered ... Jordi is a great dancer!!!

In this paper I sketch some recent ideas I had about the use of Natural Language Processing (NLP) techniques to support the analysis and elicitation of timing diagrams. I shall tell a secret: my very first paper was on NLP techniques, but at some point I got into the dark side of formal methods and never went back to NLP. It is funny that I recently got again attracted for working on the NLP area.

A digital timing diagram is a representation of a set of signals in the time domain. A timing diagram can contain many rows, usually one of them being the clock (but we do not deal always with clocks, as Alex knows very well). It is a tool that is ubiquitous in digital electronics, hardware debugging, and digital communications. Besides providing an overall description of the timing relationships, the digital timing diagram can help find and diagnose digital logic hazards.

2 Motivating Example

Let us consider the following textual description of the timing diagram of Figure 1.

Example 1 (Asynchronous Circuit). The circuit contains signals A, B and C. First, signal A goes high, which causes signal B to go high. Then, the rising of signals B and C causes signal C to go low. Afterwards, the rising of signal C causes signals A and B to go high. Then the falling of signal A causes signal B to go low.

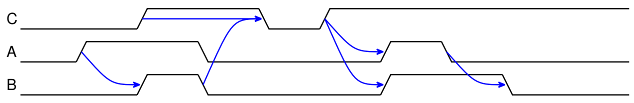

Fig. 1: Timing Diagram of an Asynchronous Circuit

Also, an alternative would be to describe the same behavior in a clocked way, as depicted in Figure 2:

Example 2 (Clocked Circuit). The synchronous circuit contains signals A, B and C. First, signal A goes high, which causes signal B to go high in the next clock edge. Then, the rising of signals B and C causes signal C to go low two clock cycles afterwards. In the next clock cycle, the rising of signal C causes signals A and B to go high one cycle afterwards. One clock cycle later, the falling of signal A causes signal B to go low.

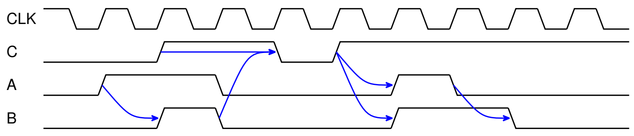

Fig. 2: Synchronous version of the circuit of Fig. 1

Next sections would illustrate how to go from the textual to the graphical description and back.

3 From Timing Diagrams to Natural Language

There are several formal descriptions of digital timing diagrams. We take a simple one, which is used by the tool `TimingDrawer` [1]. For instance, the timing diagram shown in Fig. 2 is simply specified with the following instructions:

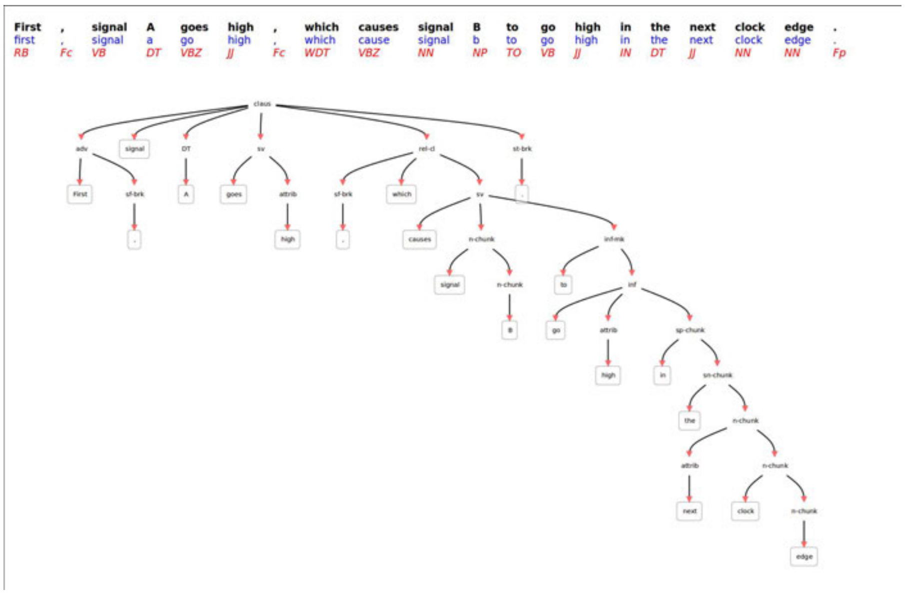
First , signal A goes high , which causes signal B to go high in the next clock edge .
first , signal a go high , which cause signal b to go high in the next clock edge .
RB Fc VB DT VBZ JJ Fc WDT VBZ NN NP TO VB JJ IN DT JJ NN NN Fp

Fig. 3: Parsing for a sentence of the timing diagram textual description.

```
CLK=clock;C=0;A=0;B=0.
# Dependency from previous A edge to new value of B
A=1.
A=>B=1;
# Dependency from multiple signals
C=1.A=0;B=0.
C,B=>C=0.
# Dependency to multiple signals
C=1.
C=>A=1,B=1.
# Vertical dependency
A=0.
A=>B=0.
```

Given a text file in the previous format, one can generate a textual explanation by: i) parsing the file in order to get a tree-like structure of the timing diagram, and ii) traversing the tree to generate the corresponding explanation in natural language, using template sentences that may be instantiated with the real names found in the tree.

4 From Natural Language to Timing Diagrams

The opposite problem to the one faced in the previous section is a challenging one: given a text describing the main behavior of a digital timing diagram, derive

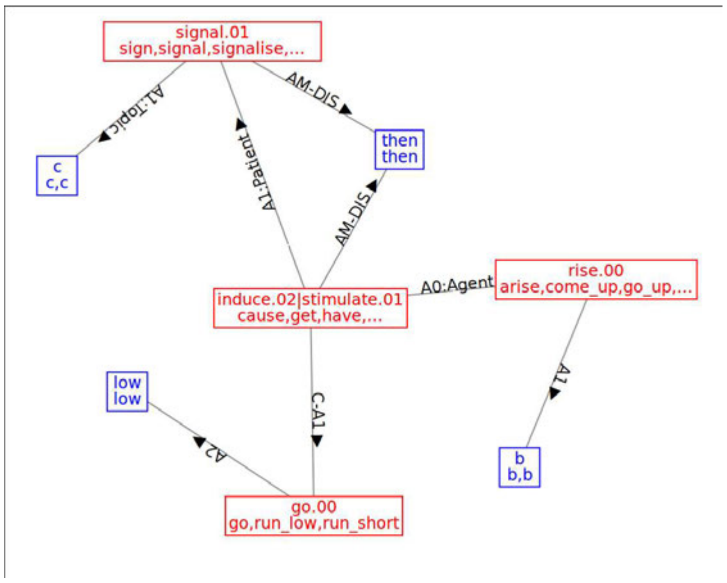

Fig. 4: Semantic Graph for a sentence of the timing diagram textual description.

a formal representation (e.g., the one used by `TimingDrawer`) of it. Inspired by the approach presented in [2] to obtain process diagrams from textual descriptions, *Natural Language Processing* (NLP) techniques can be used to tackle this problem.

Likewise it is done in [2] for the case of process diagrams, the generation of a digital timing diagram can be done in three steps. However, given the narrower focus considered in this paper, some of the steps can be significantly simplified. Below we provide an informal description of each one of the three steps considered.

Step 1: Sentence Level Analysis Using NLP techniques (e.g., tokenization, parsing, and similar), sentences in the input text can be analyzed to decompose the input into phrases with clear actors (signals, in our case) and the actions corresponding to them. Also, irrelevant or unrelated sentences are filtered. Morphosyntactic analysis is one of the prominent techniques to apply, that may derive a categorization as provided in Figure 3.

Step 2: Text Level Analysis Then the output of the previous phase is analyzed, taking into account the relationship between signals and/or signal actions across different phrases (a phenomena well known as *anaphora resolution*). For each sentence, a *semantic graph* describing these relations can be obtained. Figure 4 depicts an example of the semantic graph obtained from some of the sentences describing the digital circuit of the previous section. Nodes in this graph denote semantic meanings of the words in the sentences, and arcs represent the semantic

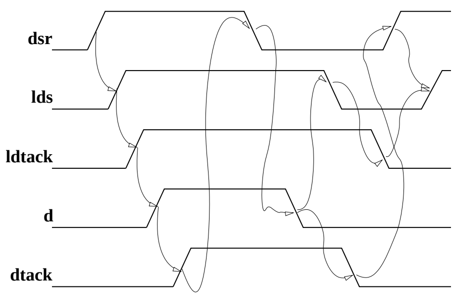

Fig. 5: VME Bus Timing Diagram.

relations between them. For instance, in Figure 4 it can be extracted that the rise of B (arc between concept "rise" and concept "B", with "B" as main actor denoted by "A1"), causes signal C (arc between concept "induce" and "signal" concept, with semantic arc "A1:Patient" denoting the signal to be the result) to go low (arc between concept "induce" and concepts "go" and "low", through arcs "C-A1" and "A2", respectively).

Step 3: Timing Diagram Generation Given the previous analyses, traversing the corresponding data structures would allow to generate the timing diagram corresponding to the input textual description. The idea would be to select the meaningful nodes/arcs from the semantic graph that can be translated into causalities in the timing diagram, generating a formal description as the one used by `TimingDrawer`.

5 Discussion

In the last decades, Alex has been a key person in the field of asynchronous circuits. One of the first works that I read from Alex was describing a VME bus controller with Signal Transition Graphs, which are labeled Petri nets representing the behavior of a digital circuit. The timing diagram of the read cycle of the controller is shown in Figure 5. In my research on asynchronous circuits (CSC encoding, synthesis), I was using this example all the time. I hope Alex can consider this paper as a way to pay back his enormous influence on the area, and in particular, on my work.

References

1. Salokin, V.: TimingDrawer Tool. https://sourceforge.net/p/timingdrawer/wiki/Home/ (2016)
2. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings. (2011) 482–496