10 Min_Max Circuit

10.1 Introduction

This benchmark circuit has been proposed by Luc Claesen for [FMVD90]. It is the first non-trivial example which has gained some popularity and reveals some problems, arising in the area of digital signal processor verification.

10.2 Specification

The Min_Max unit has an input signal *in* which consists of a sequence of integers in the range of -256 to +255. The Min_Max unit has three boolean control signals *clear*, *reset*, and *enable*. The unit produces an output sequence *out* at the same rate as *in* in the following way:

- *out* is zero if *clear* is true, independent of the other control signals.
- if *clear* is false and *enable* is false then *out* equals the last value of *in* before *enable* became false.
- if *clear* is false and *enable* is true and *reset* is true then *out* follows *in*.
- if *reset* becomes false, then *out* equals, on each time point *t*, the mean value of the maximum and minimum value of *in* until that time point. So

$$out = \frac{max(in) + min(in)}{2} \tag{10-1}$$

A number of properties of Min_Max have (deliberately) not been specified, e.g. the latency of the system is unspecified. This example was used at the international workshop on "Applied Formal Methods For Correct VLSI Design" and several solutions can be found in the proceedings [FMVD90].

10.3 Implementation

Fig. 10-1 gives an overview of the implementation of the MinMax-Circuit. A storage element Last (Fig. 10-2) may store a new value while Enable is true, otherwise the former value is stored.

The heart of the circuit is the MeanValue device (Fig. 10-3). Here the incoming values are compared with the stored minimum and maximum. The used comparator is a modified version of the nCMPN from GateLib which is able to compare numbers in 2-complemented format. If the new value is greater than the stored maximum or less than the stored minimum, its value is stored in the corresponding branch of the circuit. Otherwise, its value is discarded. The stored minimum and maximum now are added and the result is divided by two, i.e. one shift right. MeanOut contains the mean value of the stored minimum and maximum at each time point, regardless of any control signals. If reset becomes true, the stored maximum and minimum are set to -256 and +255 as soon as Clk becomes true.

To generate the required output signals depending on the control signals, the unit Three_to_four generates 4 condition signals out of the three incoming control signals (Fig. 10-4):

- Condition_1 := Clear : DataOut = '0'
- Condition_2 := ~Clear & ~Enable : DataOut = last_out
- Condition_3 := Reset & (Enable & ~Clear) : DataOut = DataIn

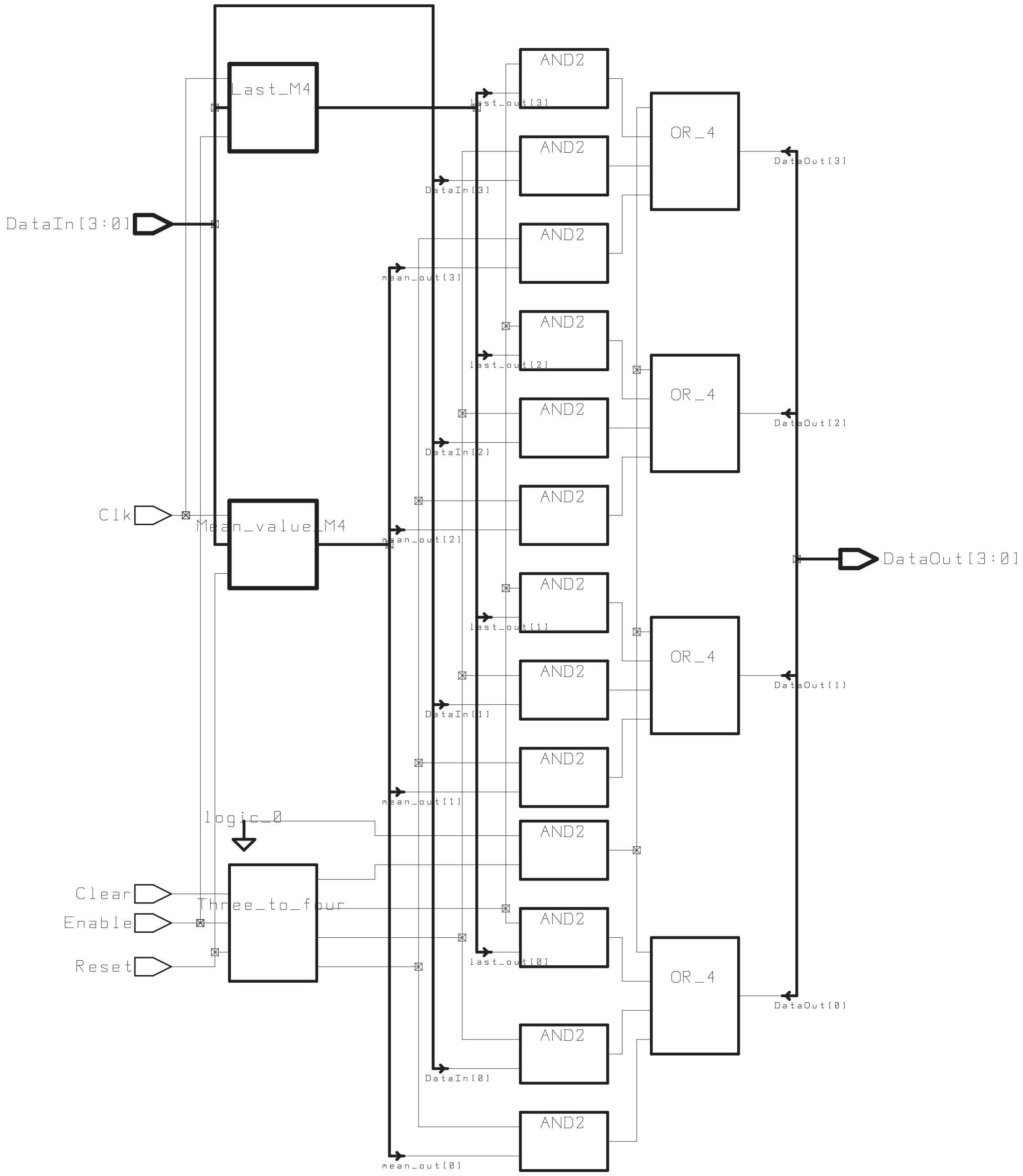

Figure 10-1: MinMax Structure

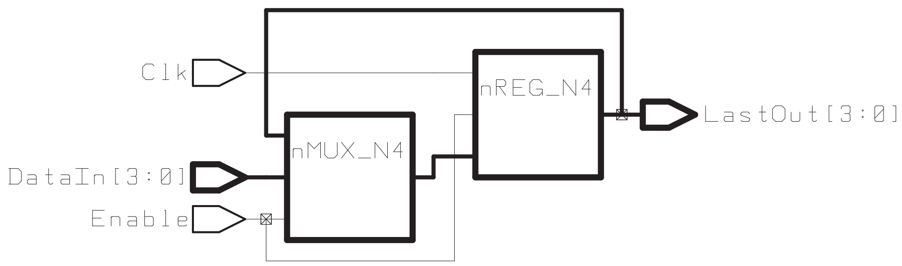

Figure 10-2: LastValue

- Condition_4 := ~Reset & (Enable & ~Clear) : DataOut = mean_out

The condition signals are the and-connected with the corresponding signals to be or-ed for output.

Fig. 10-5 shows different example simulation runs.

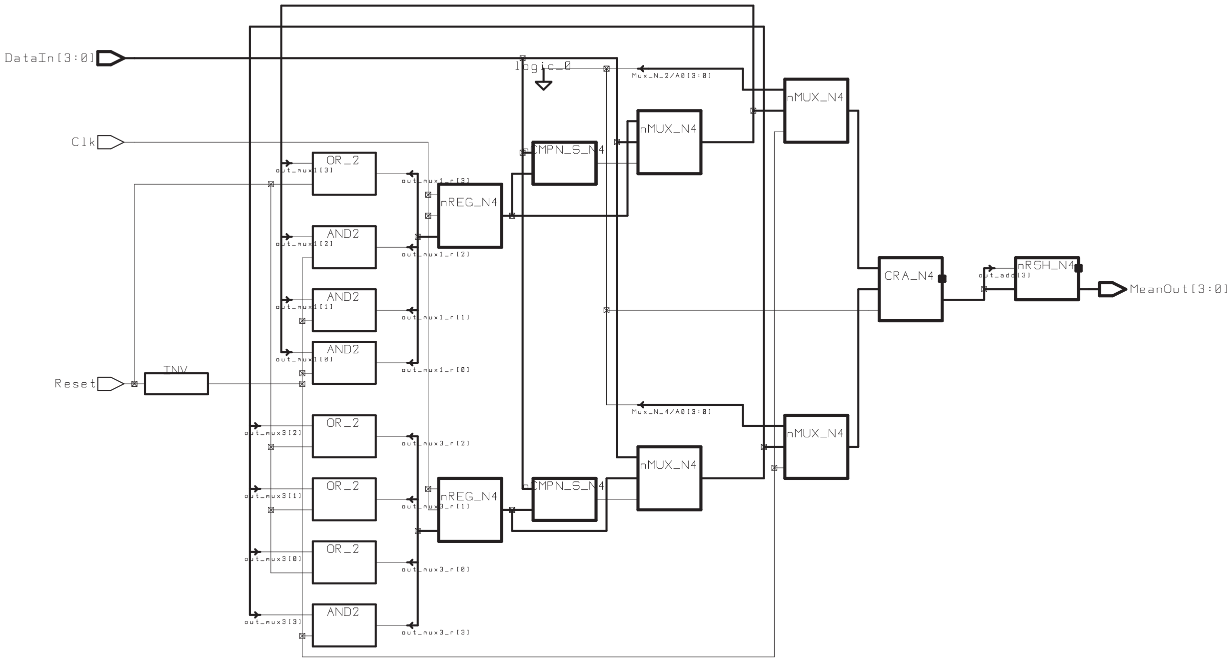

Figure 10-3: MeanValue

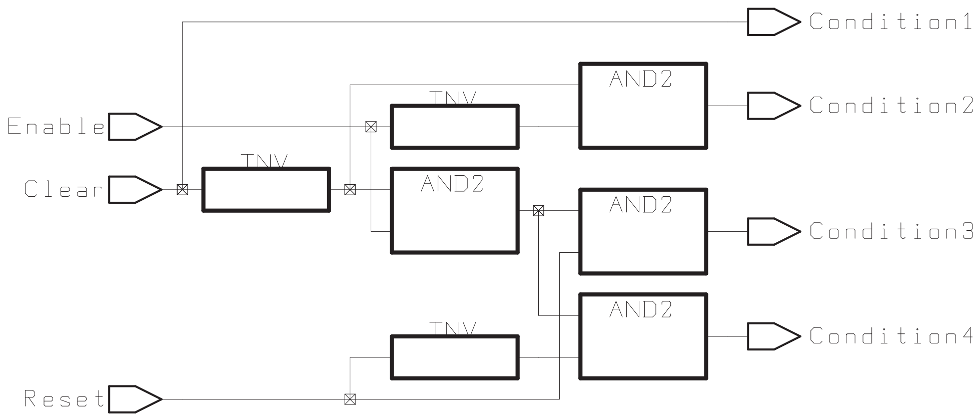

Figure 10-4: 3to4 “glue logic”

10.4 Status and Acknowledgments

Thanks to L. Claesen and J. Staunstrup for providing this example. Thanks to G. Janssen for pointing out the problems of the original specification.

10.4.1 Comments to the Specification of MIN_MAX

The specification given in section 10.2 is not as unambiguous as is seems at the first glace. To point this out, some comments from G. Janssen from Eindhoven University are added below.

- If *clear* is false and *enable* is false then *out* equals the last value of *in* before ENABLE became false.

1. Is this independent of RESET?
2. What if *clear* is false and *enable* is false from the start?

Solution:

1. Yes, we don’t care about *reset*.

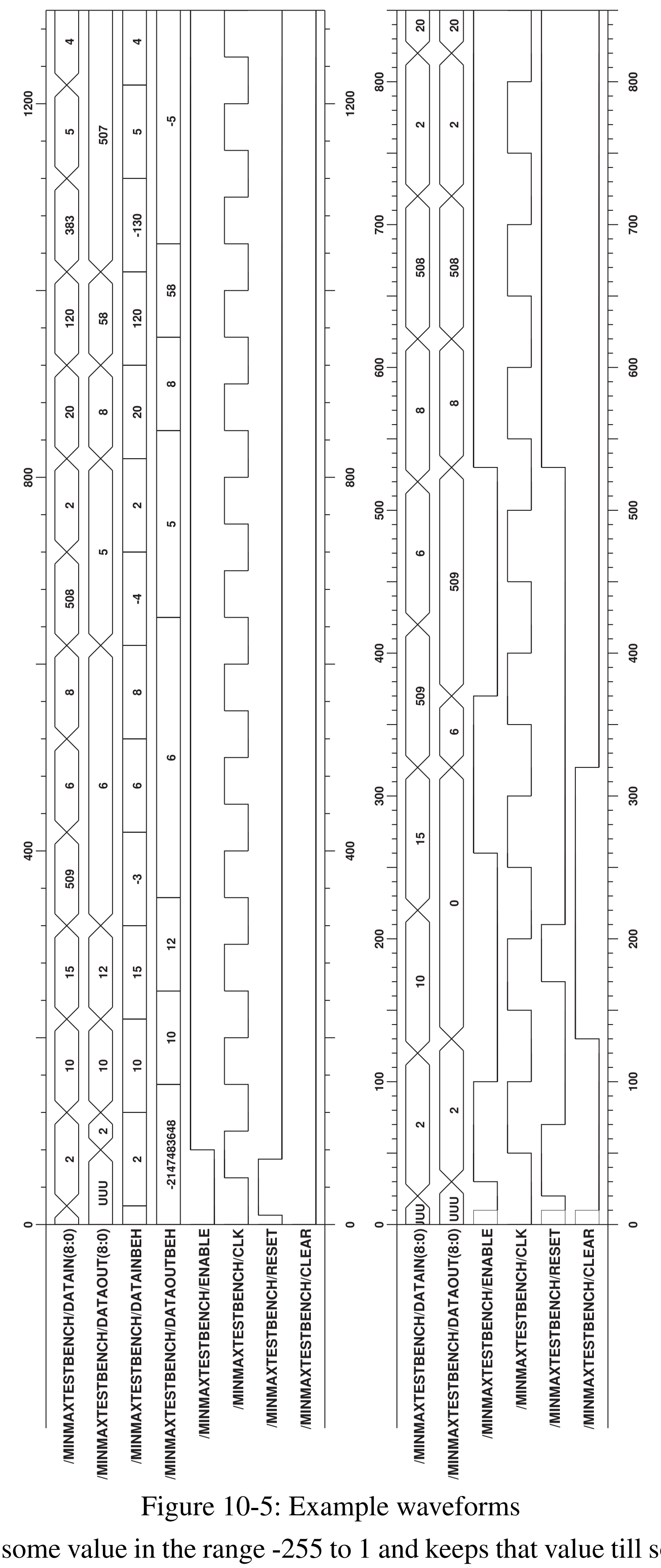

Figure 10-5: Example waveforms

2. Then *out* takes some value in the range -255 to 1 and keeps that value till some other control combination takes over.
   - If *clear* is false and *enable* is true and *reset* is true then *out* follows *in*.

1. What does “follow” precisely mean?

Solution:

1. “follow” means equals.

• If *reset* becomes false, then *out* equals, on each time point *t*, the mean value of the maximum and minimum value of *in* until that time point.

1. What must be the values of *clear* and *enable*?
2. Does “until” include or exclude the current time point t?
3. What if *reset* is false from the start?
4. When are the minimum and maximum values (re)initialised and to what value? Where does the averaging starts?
5. What if *clear* and *enable* change during *reset* is false?

Solution:

1. *clear* must be false and *enable* must be true.
2. “until” is interpreted to mean inclusive.
3. Then *min* and *max* are set to *in*.
4. Only when *reset* makes a transition from true to false are *min* and *max* reinitialised; At the time *reset* is false, *min* and *max* are set to *in*; averaging starts then afresh.

10.5 Literature

[FMVD90] L. Claesen, editor. *International Workshop on Applied Formal methods for VLSI Design*, Leuven, Belgium, 1990.

[Stau93] J. Staunstrup. IFIP WG 10.2 collection of circuit verification examples, November 1993.