20 Assoc (Associative Memory)

An associative memory has been chosen to have an “intelligent” storage element in the set of benchmark circuits. There is no arithmetic to be verified, but the matching and selection process of the memory has to be proven correct.

The design has been taken from [Haye88] (see also [Koho77]).

20.1 Specification

20.1.1 General Specification

In an associative memory (also called “content addressable memory”) each stored data word can be retrieved by using characterizing parts of the data. E.g. having stored persons together with their birth dates, when applying a date it is possible that more than one person is stored with this date. All of them are “matched” and afterwards “selected”. This matching and selection process is a key part of most associative memories. The data units to be stored are fixed-length words. The black box view of the memory is given in Fig. 20-1.

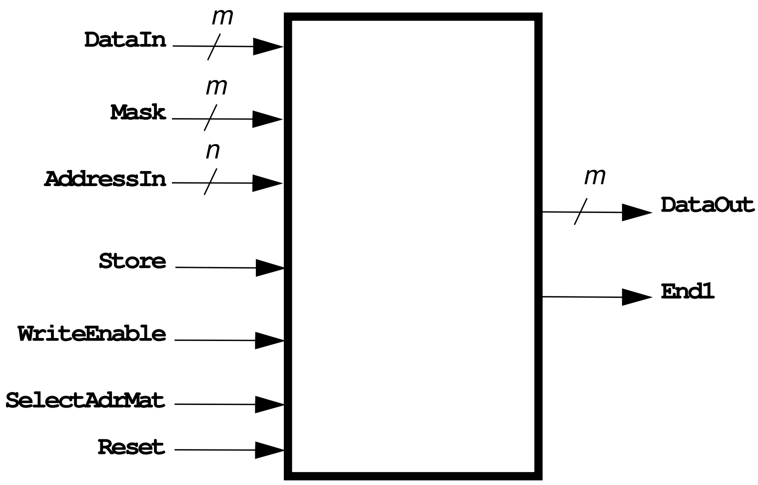

Figure 20-1: Black box view of the associative memory

Given **DataIn** = $(d_{m-1},...,d_0)$, **Mask** = $(y_{m-1},...,y_0)$ and **AddressIn** $=(a_{n-1},...,a_0)$, the operations given in table 20-1 are possible.

Operation	DataIn	Mask	AddressIn	Store	WriteEnable	Select-AdrMat	Reset
Clear Memory	(0,...,0)	(0,...,0)	(1,...,1)	1	1	1	0
Read in DataIn without Mask	$(d_{m-1},...,d_0)$	(0,…,0)	$a_i = 1$, $i \in [0, n-1]$	1	1	1	0
Request without Mask	$(d_{m-1},...,d_0)$	(0,...,0)	don’t care	1	0	0	0
Request with Mask	$(d_{m-1},...,d_0)$	$(y_{m-1},...,y_0)$	don’t care	1	0	0	0

Table 20-1: Basic operations of the associative memory

When performing a *Request* operation, all data bits are compared with the respective bits of the memory in parallel. No comparisons are done at those places where the **Mask** bits are set to “1”. If a stored data equals the applied data at least at one position (if not masked), the data entry with the smallest index is selected. If the request is started at time instant t, then at time instant t+1 (rising edge) the result is available at **DataOut** and the signal **End** is set. To start a new request, the controller has to be reset with **Reset** = 1. Using an associative memory, the comparison of data with *n* other words can be performed in O(1).

20.2 Implementation

20.2.1 General architecture

The structure of a simple word-organized associative memory is shown in Fig. 20-2. Each subpart of such a word may be chosen as a key, i.e. as a search pattern. The bits relevant for a search are marked in the mask register and are compared simultaneously with all stored words. Every matching entry results in a match signal. The "select circuit" then chooses exactly one matching word and produces a "select" signal to retrieve the word from the storage cell array to be written into the output register. There are different strategies for choosing the word like "first hit" etc.

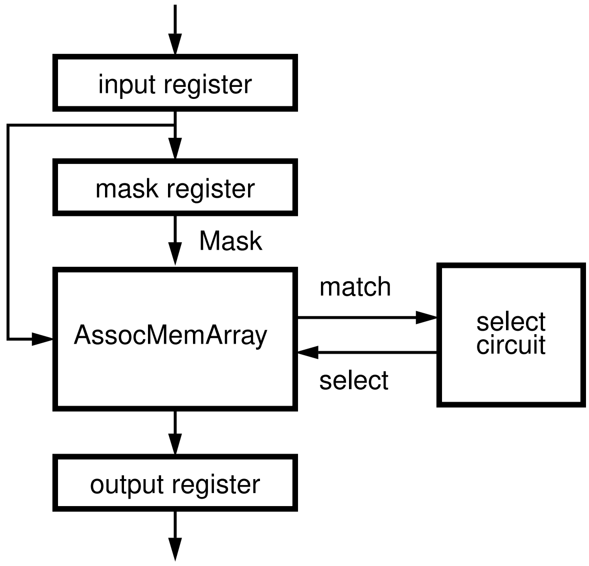

Figure 20-2: Associative memory with fixed word-length

The main module in Fig. 20-2 is the "storage cell array" AssocMemArray. Its black box view is shown in Fig. 20-3 and allows the following operations:

1. Read in a data word, consisting of an input field of length *m* (read in at **DataIn**) and a key field of length *n* (read in at **Mask**).
2. Search for data using a key fed in at **Mask**. **Match** indicates, if there have been found one or more matching entries.
3. Read out a matching data word via **DataOut**, using a key at **Mask** and a selected row (using **Select**).

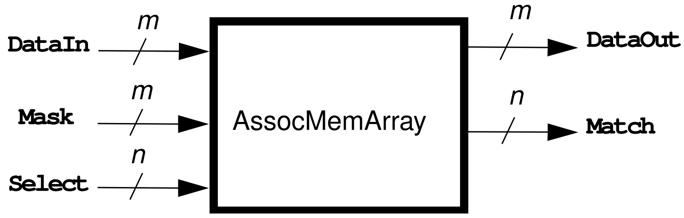

Figure 20-3: Black box view of AssocMemArray

20.2.2 Implementation of one cell of AssocMemArray

Since all words have to be compared with the key, each bit cell AssocMemCell of AssocMemArray needs a compare circuit. The realization of one cell is shown in Fig. 20-4.

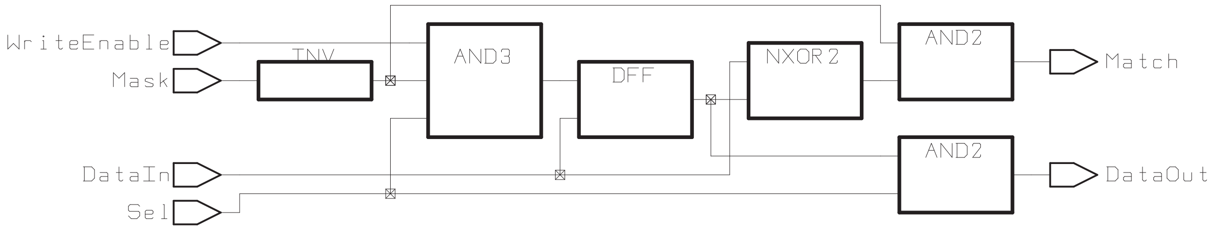

Figure 20-4: One cell of the associative memory (AssocMemCell)

The basic storage element is a D flip-flop. An NXOR-gate performs the actual comparison between the flip-flop content and the input data **DataIn**.The output of the NXOR is AND-connected with the inverted signal **Mask** to form the match signal **Match**. Hence a match only takes place, if the cell is not masked. To read out the flip-flop the select signal **Sel** is AND-connected with its content to get the data output **DataOut**. To store data in the flip-flop, its clock input is AND-connected with the select signal **Sel**, write-enable signal **WriteEnable** and the inverted mask signal **Mask**.

20.2.3 Implementation of a $m \times n$ Storage Cell Array (AssocMemArray)

Given a mask and an input word of length m. For a $m \times n$ memory, $m \cdot n$ cells of type AssocMemCell (Fig. 20-4) are necessary. Cell i, j in the i-th row and j-th column is connected as follows $(0 \le i \le m-1, 0 \le j \le n-1)$. Input **WriteEnable** is connected to a global **WriteEnable**. Data input **DataIn** is connected to the i-th least significant bit of the input word, mask input **Mask** is connected to the i-th least significant bit of the mask word. Select input **Sel** is connected to the j-th lowest significant bit of the select word. Data output **DataOut** and match output **Match** are OR-connected with all respective outputs of the j-th row, i-th column

20.2.4 Implementation of the whole Associative memory (AssocMemCirc)

The whole associative memory is shown in Fig. 20-5.

To perform read and write at definite time instances, three n-bit register for data, mask and result as well as a m-bit register for the address are needed. Read of a data word (mask word) is done via **DataIn** (**MaskIn**) with a storage signal **Store** = 1 and a rising edge of **Clock**. Analogously, an address word is read via **AddressIn** in the address register.

Data input and mask input are connected with the outputs of the data register and mask register. The n-bit "match" output of the associative memory is fed into a "select circuit". The latter is being realized by a combinational circuit, which selects exactly one hit, if more than one hit has been achieved. The n-bit output is fed into the "1" input of a multiplexer. The multiplexer is selected via **SelectAdrMat**. The n-bit output of the multiplexer is fed into the address input of the associative memory. To be able to read out data at a definite time instant, a control circuit is necessary. Here, **SelectAdrMat** is AND-connected with the inverted signals **Store**, **WriteEnable** and **Reset** and fed into the set input of the RS-flip flop.

The **Reset** line is connected to the reset input. The state of the RS flip flop thus indicates, whether data have to be stored in the output register (state = "1"). The flip flop state also provides the **End** signal. The result of the search process is available at **DataOut**.

A 4×4 memory is given in Fig. 20-7.

As an example, the data as given in table 20-2 is written into the memory. Next, the word to be fetched is written into the memory as 0001. The search mask is set to 1110, such that only

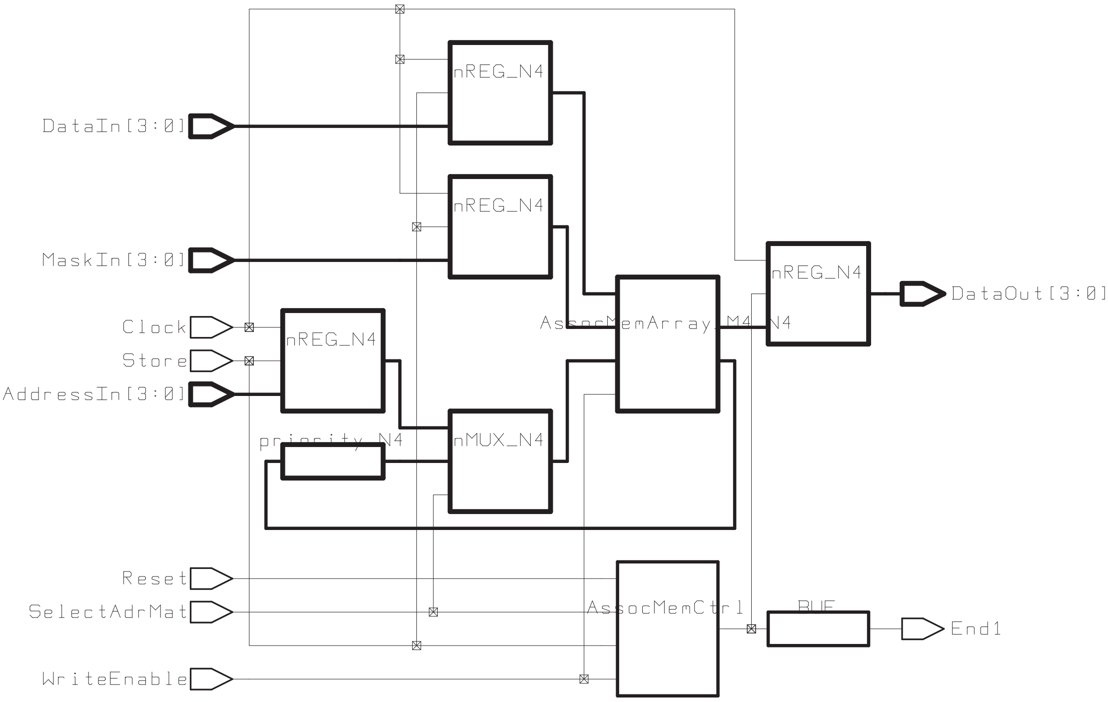

Figure 20-5: Whole associative memory (AssocMemCirc)

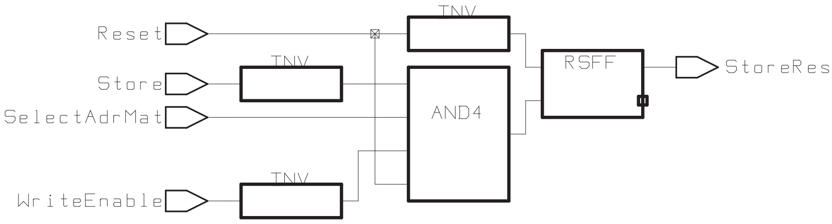

Figure 20-6: AssocMemCntrl of Fig. 20-5

the rightmost bit is used for the search. This leads to a search result 0011, which equals the content of the 4th cell.

The simulation result of this examples is shown in Fig. 20-8 (the suffix “1” denotes signals of the structural description, “2” denotes signals of the behavioral description).

20.3 Status and Acknowledgments

The presented associative memory element is currently used to get a larger hierarchical circuit, which computes the maximum of all stored numbers. This use of the circuit will be added to this description or will constitute a separate verification benchmark in the near future.

Thanks to H.-P. Eich and C.-J. Thomas for designing the circuit in a commercial design system.

20.4 Literature

[Haye88] J.P.Hayes. *Computer Architecture and Organization*. McGraw-Hill, 2. edition, 1988.

[Koho77] T. Kohonen. *Associative Memory*. Communication and Cybernetics. Springer Verlag, 1977.

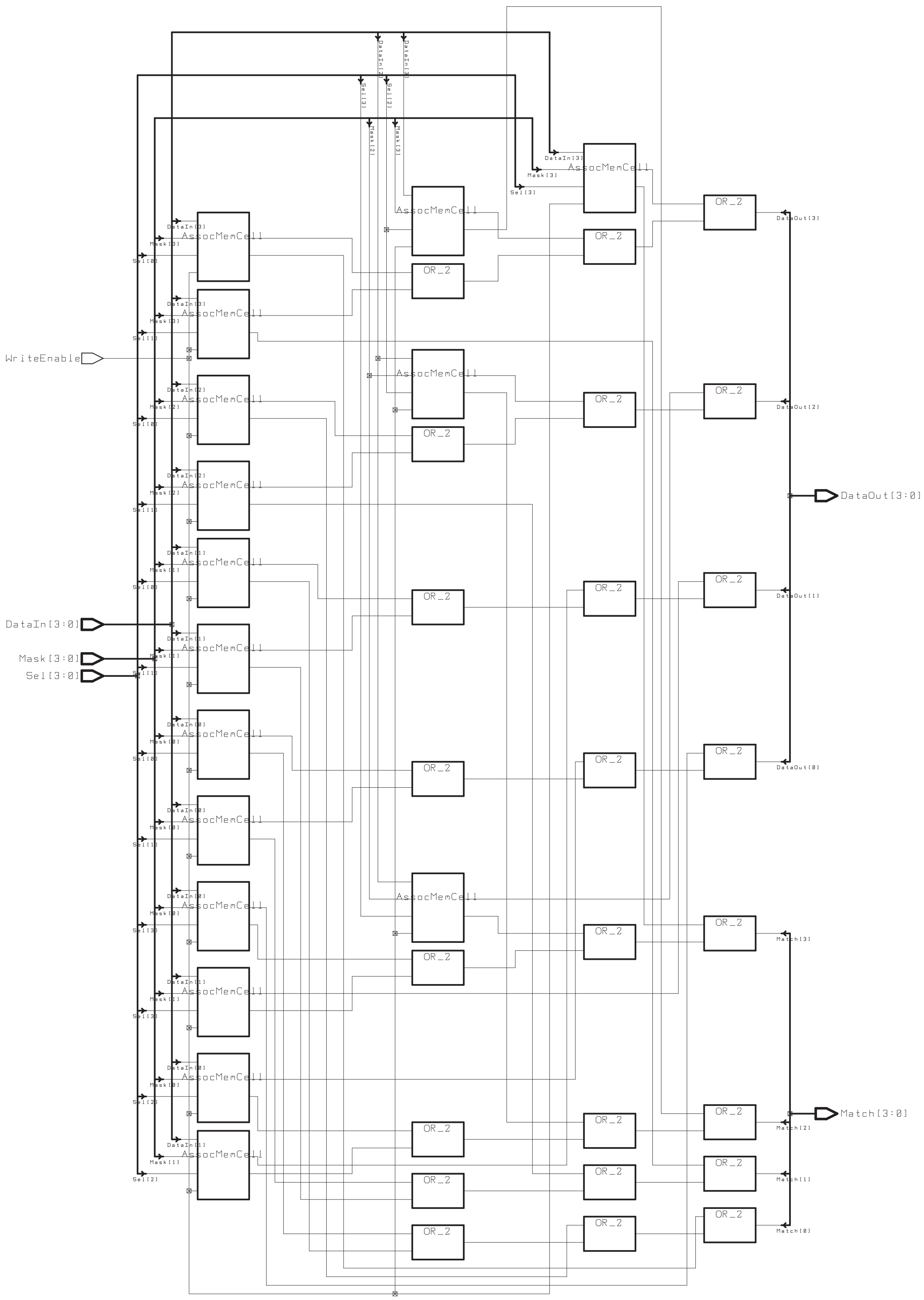

Figure 20-7: Realization of a 4×4 memory (AssocMemArray of Fig. 20-5)

cell #	data (binary)
1	0110
2	1000
3	left empty
4	0011

Table 20-2: Example Data written into the memory

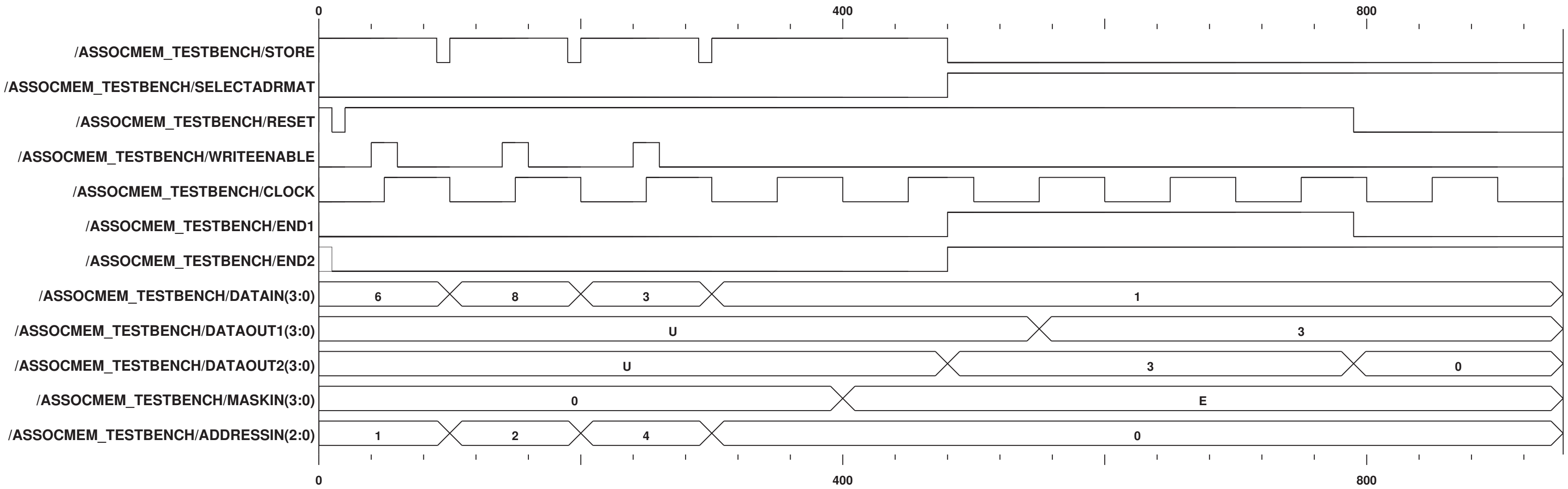

Figure 20-8: Simulation waveform of the example