Politecnico di Torino

Documentazione

Matteo Battilana Numero Matricola
Salvatore Gabriele La Greca s281589
Giovanni Pollo s290136

Contents

1 Register File

This section will be divided into subsections in which we describe single components.

1.1 Decoder

This block receives as input the *write address* on **NBIT_ADD** bits and outputs $\mathbf{2^{NBIT\_ADD} - 1}$ bits. It has the utility of converting the address of the register at which we need to write into its enable signal.

The idea is that if the input is $0b00010$ the output will be $0b00000000000000000000000000000100$. In fact if the input is decimal 2, it means that we need to write the second register of the *GLOBAL* block. In terms of enable it can be translated by having the bit with index 2 at one. In fact in the output we see that the it with index 2 has value 1, while the others are all 0.

The output is divided (in the schematic) in order to represent the group of bits. In particular we have that:

- M - 1 DOWNTO 0: bits associated to the *GLOBAL* register
- M + N - 1 DOWNTO M: bits associated to the *IN* register
- M + 2N - 1 DOWNTO M + N: bits associated to the *LOCAL* register
- M + 3N - 1 DOWNTO M + 2N: bits associated to the *OUT* register

On the top of the schematic (Figure 1) we can see an AND logic port between *ENABLE* and *WR* signals. If both *ENABLE* and *WR* are 1, it means that our register need to work. In fact, the output of the dedocer is anded with 1 and so we maintain the value. Otherwise, if one signal between *ENABLE* and *WR* is 0, the output will be 0 and so the AND with the output of the *decoder* will return all 0.

This signal goes into the *connection matrix*, which is the next block described.

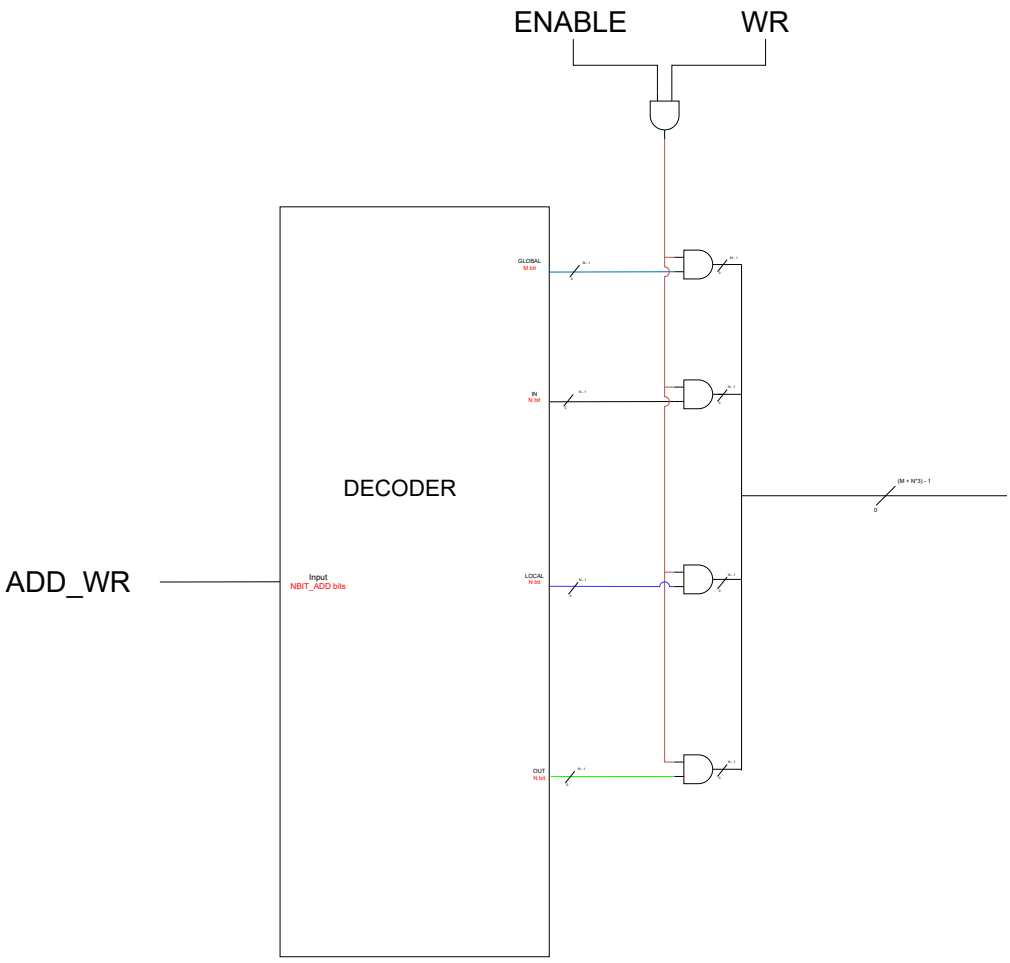

Figure 1: Schematic of the Decoder

1.2 Connection Matrix

With the previous block, we generated all our enable signals. The problem is that we have more windows. So how do we decide which window needs to be activated? Here comes the connection matrix. This block receives as inputs the signal coming from the decoder, the current window, the saved window and the address for the pop (fill) operation. The output is a signal that contains the enable signals ready for all the registers of all windows.

We have a specific structure for each block:

- GLOBAL: the global is the simplest, because it is connected directly to the output
- IN: for this block we AND the IN bits coming from the decoder with the bit (that is extended) of the related window. For example if we are evaluating the IN of the first window, we will AND the IN bits with the bit 0 of the current window.
- OUT: for this block we AND the OUT bits coming from the decoder with the bit (that is extended) of the previous related window. For example if we are evaluating the OUT of the first window, we will AND the OUT bits with the bit 4 of the current window (supposing our window has 5 bits).
- LOCAL: for this block we AND the LOCAL bits coming from the decoder with the bit (that is extended) of the related window. For example if we are evaluating the LOCAL of the first window, we will AND the LOCAL bits with the bit 0 of the current window.

For the IN and OUT we then an OR between the two outputs (the logic can be seen in the schematic), while for the LOCAL we don't have anything.

In addition to that, the connection matrix also manages the saved window, used for the pop (fill) operation. First we need to invert the addr_pop, because when we execute the pop operation, we restore data starting from the last one (we are using a STACK). The addr_pop_inverted is composed like this:

- 2N - 1 DOWNTO 0: we have the IN bits
- N - 1 DOWNTO 0: we have the LOCAL bits

The signal is splitted into two wires and is anded with the saved related saved window pointer.

In the end, we definitely OR the output of the previously described OR with the output of this AND. This is visible in the schematic.

1.3 Register File

The next block is the Register File, that is a sequence of registers. The important thing to notice in our design is how we managed the data that goes into the registers. We have two choice, data_in and from_mem. In order to choose we decided to use multiplexers. We have a multiplexer for each window. The signal used to drive the multiplexer is the saved window pointer, rotated right by 1 position and anded with the pop signal. In fact, we select from_mem only when the pop signal is 1, otherwise we need to select data_in. We use the saved window pointer shifted by 1 because..... TODO