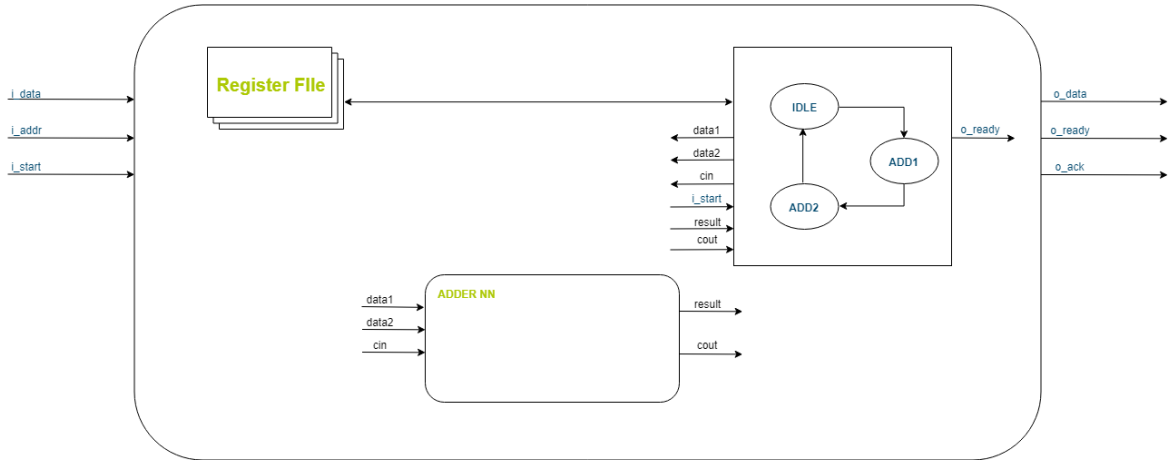
AEDVICES Consulting
166B Rue du Rocher de Lorzier
38430 Moirans– France
trainings@aedvices.com
http://www.aedvices.com

Training on
Functional Verification Methodology
Using UVM

N-BIT BINARY FULL ADDER

The AEDV-FA01 is a full adder composed of a Finite State Machine that controls the input/output signals from an unique smaller adder to perform a N+N addition using a M+M adder (M =N/2) and of a Register File where is loaded the the two operands, the Carry in, the result and the Carry Out.



Signals Description

The full adder is composed of four input and three output signals, where four are control signals (*i_start*, *i_we*, *o_ready*, *o_ack*) and the other three are register access ones (*i_data*, *i_addr*, *o_data*).

Input

- *i_data* - Data input;
- *i_addr* - Register address to be written or read;
- *i_we* - "Write Enable", when set, indicates a writing access, otherwise a reading one;
- *i_start* - This signal indicates when start the computation(addition).

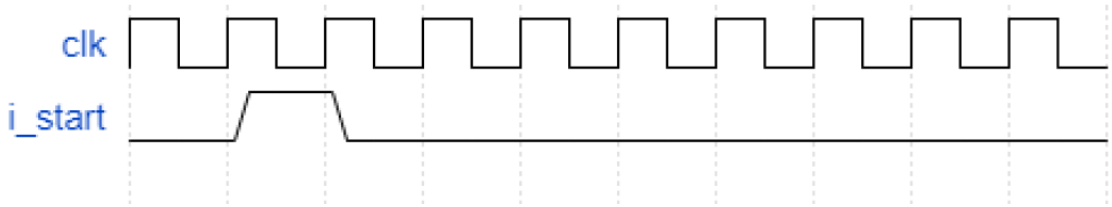
Output

- *o_data* - The output data of a reading access;
- *o_ready* - This signal is set whenever an addition is completed;
- *o_ack* - This signal is set whenever a writing access is successfully executed (Writing in writable registers).

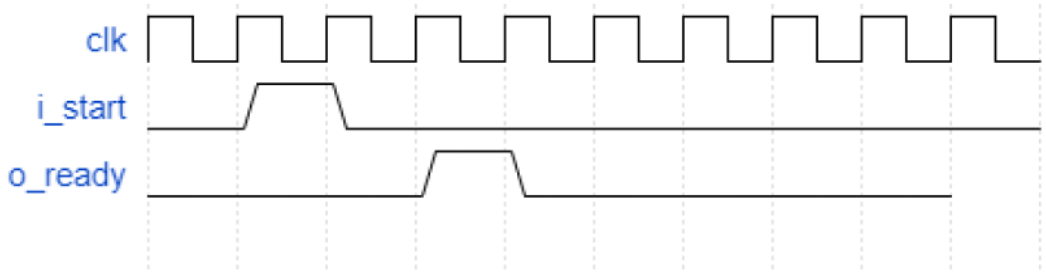
Signals Behaviour

The signals must behave like the following specifications:

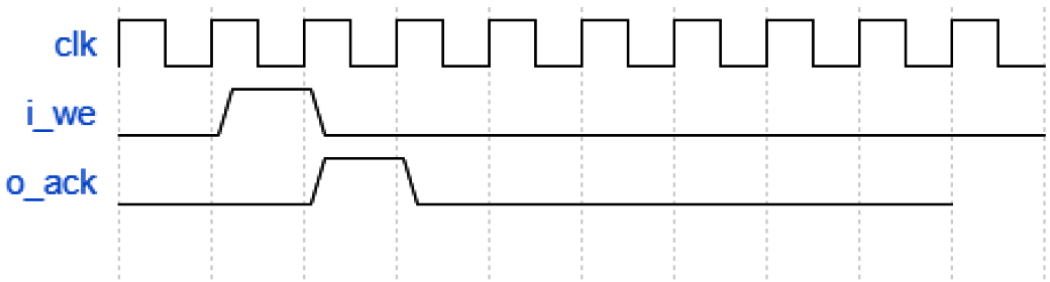
The first specification is about the time that the *i_start* signal should be high (not more than one cycle):



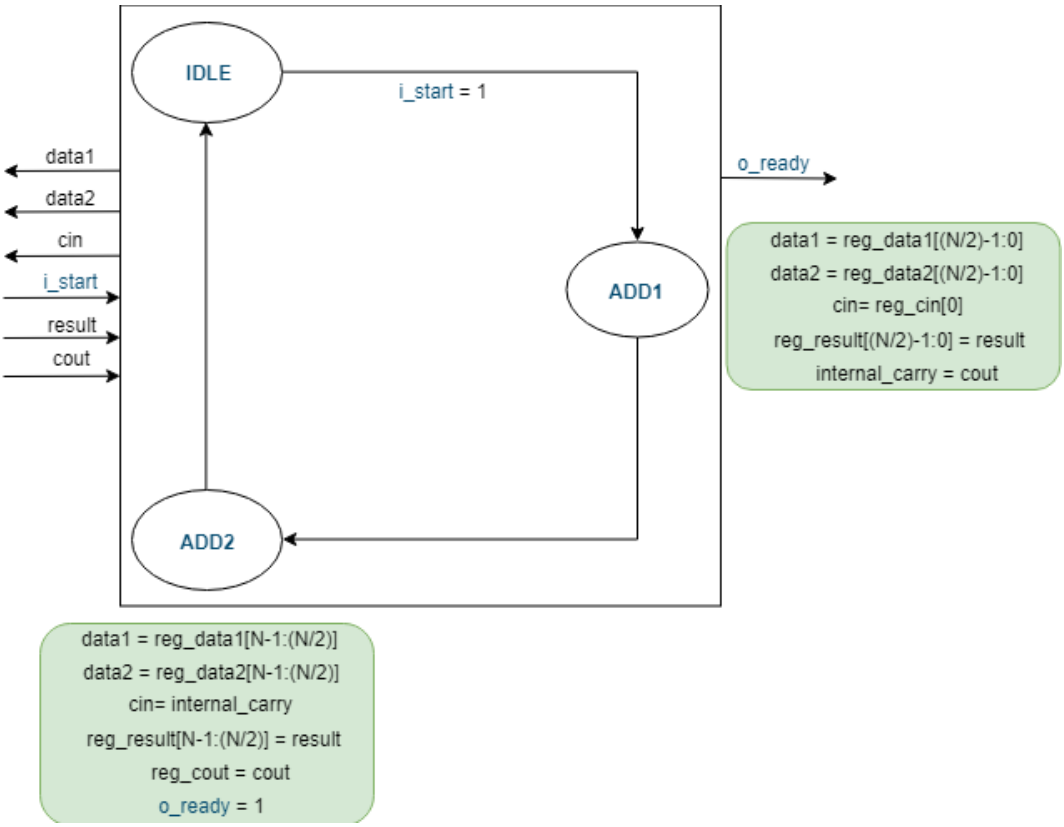
The second one: the *o_ready* signal must be active 2 clock cycles after the*i_start* positive edge..



The third one: The *o_ack* signal must be active one clock cycle after the *i_we* one is set and must following active until the following clock edge where the*i_we* is inactive indicating that the writing was performed.



Finite State Machine



The FSM is responsible to drive the correct signals to the smaller adder using the values in the register file and load the correct result in the correspondent registers. Furthermore, the FSM is started by the *i_start* signal and, when the complete result is loaded in the registers, the FSM set the *o_ready* signal.

The FSM starts in the *IDLE* state and, when *i_start* signal is set, the state changes to *ADD1* state where the first half of the data registers are driven to the smaller adder and the result is loaded in the first half of the result register. The state changes to *ADD2* where the second half is computed and loade and thus the *o_ready* signal is set.

Register File

Configuration (Not Implemented)

Address: 0x0000_0000
Access: R/W

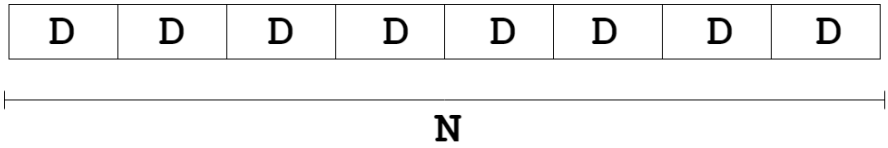
R	R	R	R	R	R	R	R

N

`Bit [N-1:0]` Reserved

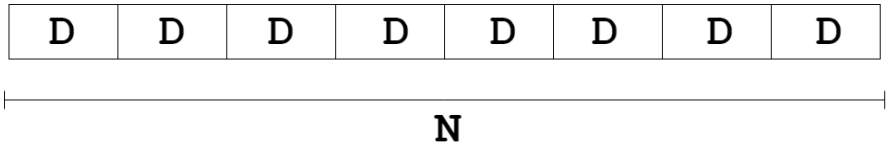
Data1

Address: 0x0000_0001
Access: R/W



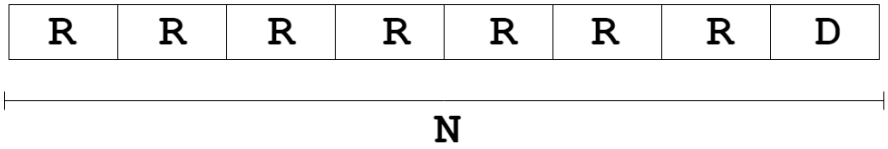
Bit [N-1:0] First Operand

Data2
Address: 0x0000_0002
Access: R/W



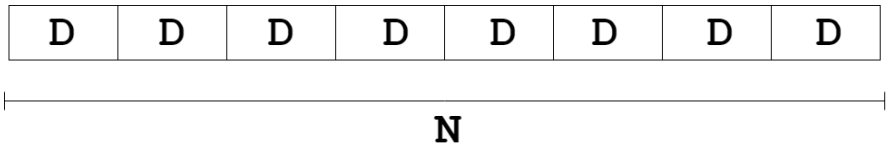
Bit [N-1:0] Second Operand

Cin
Address: 0x0000_0003
Access: R/W



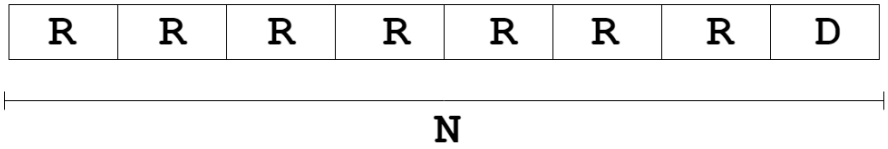
Bit [0] Carry In
Bit [N-1:1] Reserved

Result
Address: 0x0000_0004
Access: RO



Bit [N-1:1] Result

Cout
Address: 0x0000_0005
Access: RO



Bit [0] Carry Out
Bit [N-1:1] Reserved