A 8-bit Sequential Multiplier project

1 Introduction

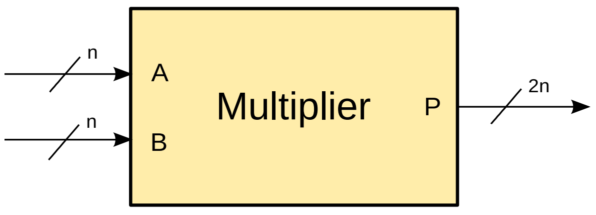
The objective of this project is to design 8 bit multiplier from initial conception to simulation. In this case, it has been taken several steps further and synthesis. Design and simulate an 8-by-8 bit shift/add multiplier should be provied. The result is a completely synthesized 8-by-8 bit shift/add multiplier with various design options for speed and area. Test bench for each module and system design should provide.

2 Binary multiplication

Example of a 4-bit unsigned multiplication (11 × 9 = 99):

```
Multiplicand          1011
Multiplier      ×     1001
                ----------
                      1011
                     0000
                    0000
                +  1011
                ----------
Product           01100011
```

The next figure is the generic entity of a *combinatorial* multiplier. Notice that the output bitwidth is the sum of the bitwidth of each input operand.

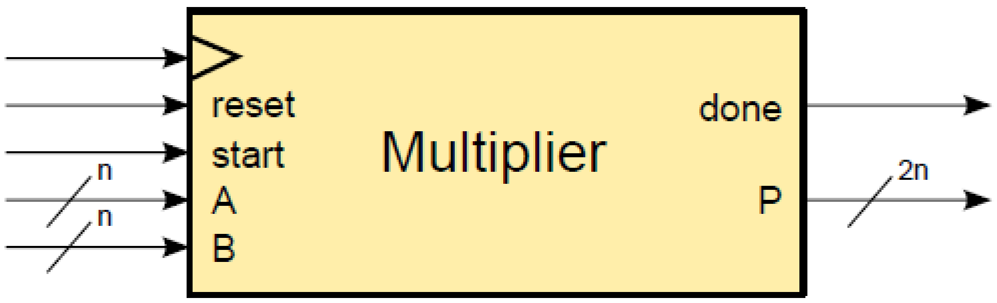


3 Sequential multiplier

A sequential multiplier requires some additional signals for synchronization purpose.

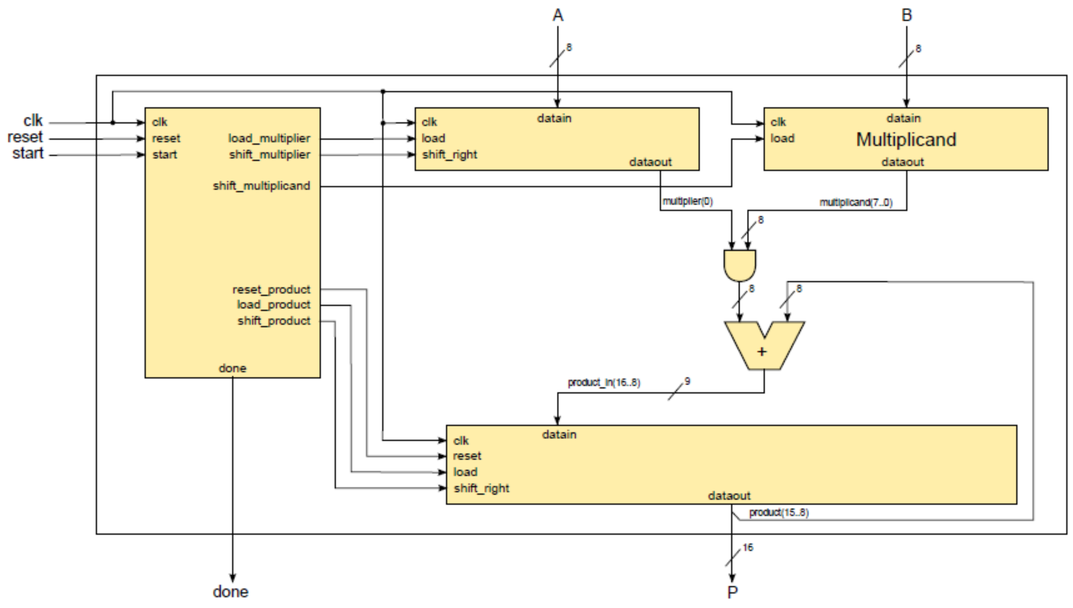
- Input **clk**: clock signal to synchronize the system.
- Input **reset**: asynchronous reset signal to initialize the system.
- Input **start**: synchronous signal that must be high to start a new operation.

Output **done**: synchronous signal that is set during 1 cycle by the multiplier when the result of the operation is available.



To implement the sequential multiplier, we will use the *Shift-and-Add* algorithm. This algorithm is inspired from the multiplication method you learned at school: you sequentially multiply the multiplicand (operand Bin the figure) by each digit of the multiplier (operand A in the figure), adding the intermediate results to the properly shifted final result.

The following figure describes the architecture of the multiplier.



The components description is listed below.

- The 8-bit register Multiplicand:
  - Operand **8** is loaded when **load** is high.
- The 8-bit shift-register Multiplier:
  - Operand **A** is loaded when **load** is high.
  - It shifts to the right when **shift-right** is high.
  - The **dataout** output is the least significant bit of the register.
- The 17-bit shift-register Product:

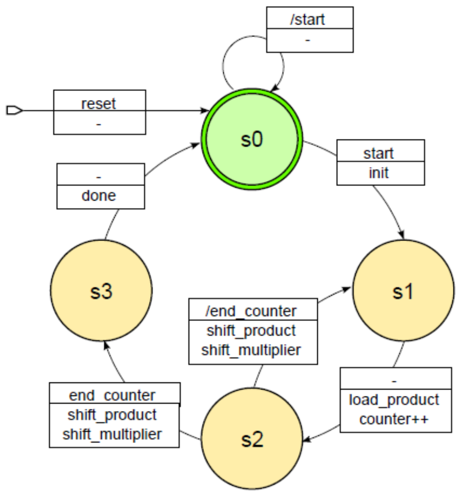
– It's initialized to 0 when **reset** is high.

– It loads the **Add** result in its most significant bits when **load** is high.
– It shifts to the right when **shift** is high.
– The **dataout** output bits are its 16 least significant bits.

- The **AND** gates:
  – It performs the AND logical operation on each **Multiplicand** bits with the least significant bit of the **Multiplier**.
- The 8-bit **Adder**:
  – It adds the result of the **AND** gates with the 8 most significant bits of the **Product** output.
- The system **Controller**:
  – It contains a state machine, and it sets the control signals of all the components.

4 State machine

The controller contains the following state machine. It includes a 3-bit saturating counter used to count the 8 steps of the addition algorithm.



- The **reset** signal resets the state machine to state **S0**.
- State **S0**: We wait until the **start** signal is active before going into state **S1** and initializing the registers where:

The **Multiplicand** and **Multiplier** registers load the input values.
– The **Product** register and the **loop counter** are initialized to 0.
• State **S1**: The next state is always **S2**.
– The ALU result is loaded in the **Product** register.
– The **loop counter** is incremented.
• State **S2**: If the **loop counter** saturates (overflow after an increment) we continue to state **S3**, otherwise
we return to state **S1**.
– In each cases, the **Multiplier** and the **Product** registers are shifted.

• State **S3**: The Multiplication is complete and the next state is **S0**.
– The **done** signal is set.

Project deliverable:
you should submit a report include:

- Introduction.
- Design procedure for your project.
- Design code and simulation code for each block.
- Test bench for the complete design.
- Synthesis for your design.
- Conclusion.

A_n