# SystemC RTL Multiplier and Approximate Multiplier Report

**University of Tehran – Electrical and Computer Engineering Department**
**Course:** Object-Oriented Modeling of Electronic Circuits (ECE 342 - Spring 1404)
**Students :** Amir Abbas Moumeni Zadeh – Mahdis Mirzaei
**Due Date:** Ordibehesht 19, 1404

## 1. Objective

The goal of this project is to develop a Radix-2 sequential 8-bit multiplier and incorporate it into a 16-bit approximate multiplier (VAM16). The final design includes both RTL-level and BFM-level models, implemented and tested using SystemC.

## 2. Part 1 – 8-bit Sequential Multiplier

### 2.1 Design Description

This module multiplies two 8-bit inputs, `A` and `B`, producing a 16-bit output `W`. It uses `startB` and `readyB` handshaking and runs on a positive clock edge with asynchronous reset.

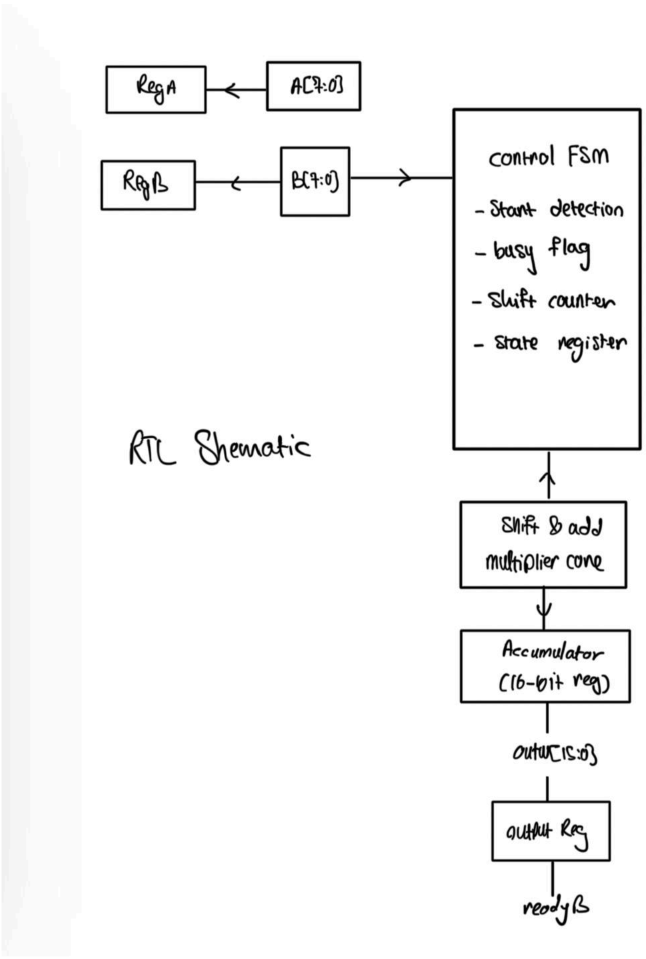
### 2.2 Interface

- **Inputs:**
  `A[7:0]`, `B[7:0]`, `clk`, `rst`, `startB`
- **Outputs:**
  `W[15:0]`, `readyB`

### 2.3 SystemC Code Overview

```
SC_MODULE(Multiplier) {
    sc_in<bool> clk, rst, startB;
    sc_in<sc_uint<8>> A, B;
    sc_out<sc_uint<16>> W;
    sc_out<bool> readyB;
    // ...
};
```

**Behavior:**
The `multiply_process()` method is triggered on clock edges. When `startB` is high, the multiplier loads inputs, performs multiplication, and asserts `readyB` when done.



## 3. Part 2 – Approximate Multiplier (VAM16)

### 3.1 Concept

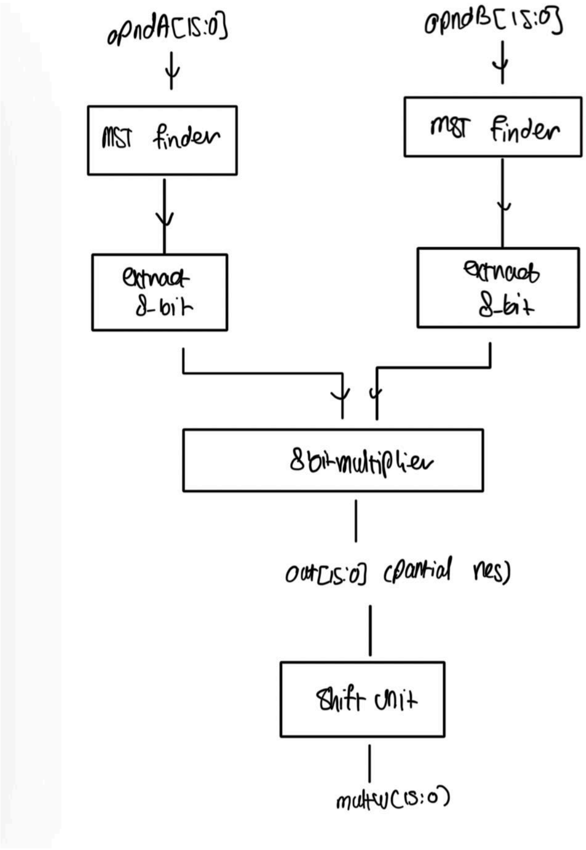
The VAM16 design extracts the most significant 8-bit portion of each 16-bit operand, based on the position of the first ‘1’ from the left. It multiplies the 8-bit segments using the base multiplier and left-shifts the result according to the number of ignored bits, producing a 16-bit approximate output.

### 3.2 SystemC Module Structure

cpp
CopyEdit

```
SC_MODULE(ApproxMultiplier) {
    sc_in<sc_uint<16>> A, B;
    sc_out<sc_uint<16>> W;
    sc_in<bool> start, clk, rst;
    sc_out<bool> ready;
    // Internal signals and 8-bit multiplier instance
};
```

- Extracts 8-bit segments using bitwise scan
- Feeds segments into `Multiplier`
- Shifts result left by ignored bits



---

# 4. Part 3 – Bus Functional Model (BFM)

## 4.1 BFM Description

`ApproxMulBFM` models the approximate multiplier's behavior cycle-accurately using SystemC's CTHREAD mechanism. It reads `start_sig`, performs the MSB extraction, multiplication, and zero-padding, and drives `ready_sig` and `out_mul`.

### 4.2 Code Overview

```
SC_MODULE(ApproxMulBFM) {
    sc_in_clk clk;
    sc_in<bool> reset_n;
    sc_in<sc_logic> start_sig;
    sc_out<sc_logic> ready_sig;
    sc_in<sc_lv<16>> in_A, in_B;
    sc_out<sc_lv<16>> out_mul;

    void run();  // Main process
};
```

### 4.3 Helper Function

```
unsigned int count_leading_zeros(unsigned int value) const;
```

This function aids in determining the bit range to extract from each operand.

---

## 5. Simulation and Testing

### 5.1 Testbench Summary

Each module was tested using a dedicated testbench to validate:

- Correctness of output
- Proper handshaking
- Cycle-accurate response in BFM

---

## 6. Conclusion

This project implemented a SystemC-based RTL multiplier and extended it into an approximate 16-bit multiplier with hardware-efficient design. The BFM model successfully mimics the hardware behavior for simulation and testing purposes. The project demonstrates effective modular design and verification methodology in SystemC.

---