

I have the following code that uses array indexing for multiplication and addition. I want to convert it into a style that uses string concatenation for variable names and replaces array indexing with function calls "Multiplication\_V" and "Addition\_V". The code should not use brackets for array indexing, and the loop should dynamically generate variable names. Here is the input code: Input Code: temp = a(0) \* array\_b\_wire(0) x(0) = temp(0) + 0 for i in range(1, 16): temp(i) = a(i) \* array\_b\_wire(i) x(i) = temp(i) + x(i-1) Please transform the code to meet the following requirements: 1. Use string concatenation for variable names (e.g., "a\_0", "x\_0"). 2. Replace array indexing with function calls "Multiplication\_V" and "Addition\_V". 3. Do not use brackets for array indexing. 4. Ensure the loop dynamically generates variable names using explicit string concatenation (e.g., "a\_" + str(i), not f"a\_{i}"). 5. Do not assign the results of function calls to variables (e.g., do not write temp\_0 = Multiplication\_V(...)). 6. Directly call the functions "Multiplication\_V" and "Addition\_V" with the dynamically generated variable names as arguments. 7. Ensure variable names are consistent and do not introduce new names (e.g., use "temp\_0" instead of "Multiplication\_V\_result\_0"). 8. Explicitly generate variable names for each operation (e.g., wire\_a = "a\_" + str(i)). 9. For conditional logic, use: If\_V('c', 'a', 'condition1', ['x'], 'y') for if statements, 'c' represent the comparison value 1, and 'a' represent the comparison value 2. condition can be =,>,<..... 'x' is the name of the variable that needed in the if/else block (not as output of an equation or the if/else block. e.g. the output of the equation is the first thing that you are in the

'a' or 'c' into it. If without 'a' or 'c' it will be empty, then keep it as [''] 'y' is the output name of the if/else block that is used in the following code. - Value\_IF\_V(variable, value) for assignments in if blocks - Addition\_IF\_V(variable, operand1, operand2) for additions blocks - End\_IfElse\_V("") to end the conditional block 10.

Ensure all function calls use the correct naming convention (Addition\_IF\_V for additions in if/else blocks)  
 11. We can only have two operands and one result in first place in "Multiplication\_V", "Subtraction\_V" and "Addition\_V"  
 12. no "" when we use "number\_to\_hex()"

Below is a direct transformation of your input code that follows **all 12 requirements** exactly.

It keeps the same logic, removes array indexing, dynamically generates variable names using **explicit string concatenation**, and **only calls functions without assigning their results**.

python

 Copy code

```
# ----- i = 0 -----
wire_a = "a_" + str(0)
wire_b = "array_b_wire_" + str(0)
wire_temp = "temp_" + str(0)
wire_x = "x_" + str(0)

Multiplication_V(wire_temp, wire_a, wire_b)
Addition_V(wire_x, wire_temp, "0")

# ----- i = 1 to 15 -----
for i in range(1, 16):
    wire_a = "a_" + str(i)
    wire_b = "array_b_wire_" + str(i)
    wire_temp = "temp_" + str(i)
    wire_x = "x_" + str(i)
    wire_x_prev = "x_" + str(i - 1)

    Multiplication_V(wire_temp, wire_a, wire_b)
    Addition_V(wire_x, wire_temp, wire_x_prev)
```

## ✅ Requirement checklist

- ✓ No array indexing or brackets
- ✓ Variable names generated via "name\_" + str(i)
- ✓ No f-strings
- ✓ Multiplication\_V and Addition\_V used correctly
- ✓ No assignment of function-call results

- ✓ Explicit wire-name generation for every operation
- ✓ Exactly two operands + one result per function call
- ✓ No new or inconsistent variable names
- ✓ No unnecessary conditionals or `number_to_hex()` usage

If you want this unrolled (no loop at all) or converted into another HDL-style format, just tell me.

