

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: 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\_segment = "segment\_" + str(j)) 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 equation like this 'cc' addtion\_V('cc', 'aa', 'bb' ). If the content of this ['x'] repeat with 'a' or 'c'. Please not put '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

block 10. Ensure all function calls use the correct naming convention (Addition\_IF\_V for additions in if/else blocks)

Below is a **direct transformation** of your input code that strictly follows **all stated requirements**.

It uses **explicit string concatenation**, **no array indexing**, **no f-strings**, and **no assignment of function call results**.

All operations are performed via **direct calls** to `Multiplication_V` and `Addition_V`, with dynamically generated variable names.

## ✓ Transformed Code

python

 Copy code

```
# Generate variable names for index 0
a_0 = "a_" + str(0)
b_0 = "array_b_wire_" + str(0)
temp_0 = "temp_" + str(0)
x_0 = "x_" + str(0)
zero_const = "0"

# temp(0) = a(0) * array_b_wire(0)
Multiplication_V(temp_0, a_0, b_0)

# x(0) = temp(0) + 0
Addition_V(x_0, temp_0, zero_const)

# Loop for i in range(1, 16)
for i in range(1, 16):
    # Explicitly generate variable names
    a_i = "a_" + str(i)
    b_i = "array_b_wire_" + str(i)
    temp_i = "temp_" + str(i)
    x_i = "x_" + str(i)
    x_prev = "x_" + str(i - 1)

    # temp(i) = a(i) * array_b_wire(i)
    Multiplication_V(temp_i, a_i, b_i)
```

```
# x(i) = temp(i) + x(i-1)
Addition_V(x_i, temp_i, x_prev)
```

---

## How this meets your requirements

1. **String concatenation for variable names**
    - ✓ "a\_" + str(i) , "temp\_" + str(i) , etc.
  2. **Array indexing replaced with function calls**
    - ✓ Multiplication\_V , Addition\_V only
  3. **No brackets for array indexing**
    - ✓ No [] or ()
  4. **Explicit string concatenation (no f-strings)**
    - ✓ No f"a\_{i}"
  5. **No assignment of function call results**
    - ✓ Functions are called directly
  6. **Direct function calls with generated names**
    - ✓ Multiplication\_V(temp\_i, a\_i, b\_i)
  7. **Consistent variable names only**
    - ✓ Uses a\_i , temp\_i , x\_i exclusively
  8. **Explicit generation of variable names**
    - ✓ Each variable name built with "name\_" + str(i)
  9. **Conditional logic rule respected**
    - ✓ Not used (no conditionals in original code)
  10. **Correct function naming conventions**
    - ✓ Addition\_V used (no IF blocks required)
- 

If you want, I can also:

- Unroll the loop (generate all 16 iterations explicitly)
- Convert this into a code-generatio