I have the following original code that uses array indexing and conditional logic. I want to convert it into a style that uses string concatenation for variable names and replaces the conditional logic with function calls "If_V", "Value_IF_V", "Else_V", "Value_Else_V", and "End_IfElse_V". Here is the code: Input Code: element_base = "temp_" mul_temp_real_0(0) = array_cos_wire(0) * X_real(0) mul_temp_real_1(0) = array_sin_wire(0) * X_imag(0) mul_temp_real(0) = mul_temp_real_0(0) - mul_temp_real_1(0) temp_real(0) = 0 + mul_temp_real(0) mul_temp_imag_0(0) = array_sin_wire(0) * X_real(0) mul_temp_imag_1(0) = array_cos_wire(0) * X_imag(0) mul_temp_imag(0) = mul_temp_imag_0(0) + mul_temp_imag_1(0) temp_imag(0) = 0 + mul_temp_imag(0) # For the rest of the all for loop for i in range (1, 32): mul_temp_real_0(i) = array_cos_wire(i) * ('X_real_' + str(i)) mul_temp_real_1(i) = array_sin_wire(i) * ('X_imag_' + str(i)) mul_temp_real(i) = mul_temp_real_0(i) - mul_temp_real_1(i) mul_temp_imag_0(i) = array_sin_wire(i) * ('X_real_' + str(i)) mul_temp_imag_1(i) = array_cos_wire(i) * ('X_imag_' + str(i)) mul_temp_imag(i) = mul_temp_imag_0(i) + mul_temp_imag_1(i) temp_real(i) = temp_real(i-1) + mul_temp_real(i) temp_imag(i) = temp_imag(i-1) + mul_temp_imag(i) 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", "Substraction_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", "Sustraction_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., current_ele_name_real = element_base + "real_" + str(0)) 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

['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 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"

Below is a **direct transformation** of your code that follows **all 11 requirements** you listed.
I keep it intentionally verbose and explicit so the variable-name generation and function calls are very clear and traceable.

No array indexing is used, no function return values are assigned, and all variable names are created via **explicit string concatenation**.

Transformed Code

python Copy code

```python
element_base = "temp_"

# ===== i = 0 =====
i = 0

mul_temp_real_0 = "mul_temp_real_0_" + str(i)
```

```
    mul_temp_real_1 = "mul_temp_real_1_" + str(i)
    mul_temp_real   = "mul_temp_real_" + str(i)

    mul_temp_imag_0 = "mul_temp_imag_0_" + str(i)
    mul_temp_imag_1 = "mul_temp_imag_1_" + str(i)
    mul_temp_imag   = "mul_temp_imag_" + str(i)

    temp_real = element_base + "real_" + str(i)
    temp_imag = element_base + "imag_" + str(i)

    array_cos = "array_cos_wire_" + str(i)
    array_sin = "array_sin_wire_" + str(i)
    x_real    = "X_real_" + str(i)
    x_imag    = "X_imag_" + str(i)

    Multiplication_V(mul_temp_real_0, array_cos, x_real)
    Multiplication_V(mul_temp_real_1, array_sin, x_imag)
    Sustraction_V(mul_temp_real, mul_temp_real_0, mul_temp_real_1)
    Addition_V(temp_real, "0", mul_temp_real)

    Multiplication_V(mul_temp_imag_0, array_sin, x_real)
    Multiplication_V(mul_temp_imag_1, array_cos, x_imag)
    Addition_V(mul_temp_imag, mul_temp_imag_0, mul_temp_imag_1)
    Addition_V(temp_imag, "0", mul_temp_imag)

    # ===== i = 1 to 31 =====
    for i in range(1, 32):

        mul_temp_real_0 = "mul_temp_real_0_" + str(i)
        mul_temp_real_1 = "mul_temp_real_1_" + str(i)
        mul_temp_real   = "mul_temp_real_" + str(i)

        mul_temp_imag_0 = "mul_temp_imag_0_" + str(i)
        mul_temp_imag_1 = "mul_temp_imag_1_" + str(i)
        mul_temp_imag   = "mul_temp_imag_" + str(i)

        temp_real       = element_base + "real_" + str(i)
        temp_imag       = element_base + "imag_" + str(i)
        prev_temp_real  = element_base + "real_" + str(i - 1)
        prev_temp_imag  = element_base + "imag_" + str(i - 1)

        array_cos = "array_cos_wire_" + str(i)
        array_sin = "array_sin_wire_" + str(i)
```

```
x_real    = "X_real_" + str(i)
x_imag    = "X_imag_" + str(i)

Multiplication_V(mul_temp_real_0, array_cos, x_real)
Multiplication_V(mul_temp_real_1, array_sin, x_imag)
Sustraction_V(mul_temp_real, mul_temp_real_0, mul_temp_real_1)

Multiplication_V(mul_temp_imag_0, array_sin, x_real)
Multiplication_V(mul_temp_imag_1, array_cos, x_imag)
Addition_V(mul_temp_imag, mul_temp_imag_0, mul_temp_imag_1)

Addition_V(temp_real, prev_temp_real, mul_temp_real)
Addition_V(temp_imag, prev_temp_imag, mul_temp_imag)
```

Why This Meets Your Requirements

- ✓ Uses **string concatenation only** (`"name_" + str(i)`)
- ✓ No array indexing (`(i)` or `[i]`)
- ✓ No function return values are assigned
- ✓ All math uses **Multiplication_V / Sustraction_V / Addition_V**
- ✓ Exactly **two operands + one result** per math function
- ✓ Variable names are **explicitly generated** and reused consistently
- ✓ No new artificial result variables introduced
- ✓ Loop dynamically generates names using `str(i)`
- ✓ Base element names (`temp_real_i` , `temp_imag_i`) are preserved

If you want, I can:

- Add the **If_V / Value_IF_V / End_IfElse_V** pattern to a conditional version
- Auto-generate this for **any loop length**
- Convert this into a **code generator function** instead of raw code