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",

"Substraction_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))

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", "Substraction_V" and "Addition_V"

Below is a **direct transformation** of your code that follows **all stated constraints**.
I keep the structure readable, explicitly generate **string-based variable names**, avoid assignments from function calls, avoid array indexing, and only use the allowed function calls with **(result, op1, op2)** ordering.

I assume the function signatures are:

scss

```scss
Multiplication_V(result, operand1, operand2)
Substraction_V(result, operand1, operand2)
Addition_V(result, operand1, operand2)
```

Copy code

No conditional logic exists in your original code, so **If_V / Else_V** functions are **not introduced**.

Transformed Code

python

```python
element_base = "temp_"

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

i = 0
```

```
current_ele_name_real = element_base + "real_" + str(i)
current_ele_name_imag = element_base + "imag_" + str(i)

mul_temp_real_0_name = "mul_temp_real_0_" + str(i)
mul_temp_real_1_name = "mul_temp_real_1_" + str(i)
mul_temp_real_name   = "mul_temp_real_" + str(i)

mul_temp_imag_0_name = "mul_temp_imag_0_" + str(i)
mul_temp_imag_1_name = "mul_temp_imag_1_" + str(i)
mul_temp_imag_name   = "mul_temp_imag_" + str(i)

array_cos_name = "array_cos_wire_" + str(i)
```

```
x_real_name = "X_real_" + str(i)
x_imag_name = "X_imag_" + str(i)

zero_name = "0"

Multiplication_V(mul_temp_real_0_name, array_cos_name, x_real_name)
Multiplication_V(mul_temp_real_1_name, array_sin_name, x_imag_name)

Substraction_V(mul_temp_real_name, mul_temp_real_0_name, mul_temp_real_1_n

Addition_V(current_ele_name_real, zero_name, mul_temp_real_name)

Multiplication_V(mul_temp_imag_0_name, array_sin_name, x_real_name)
Multiplication_V(mul_temp_imag_1_name, array_cos_name, x_imag_name)

Addition_V(mul_temp_imag_name, mul_temp_imag_0_name, mul_temp_imag_1_name)

Addition_V(current_ele_name_imag, zero_name, mul_temp_imag_name)


# -------- i = 1 to 31 --------

for i in range(1, 32):

    current_ele_name_real = element_base + "real_" + str(i)
    current_ele_name_imag = element_base + "imag_" + str(i)

    prev_ele_name_real = element_base + "real_" + str(i - 1)
```

Copy code

```
        prev_ele_name_imag = element_base + "imag_" + str(i - 1)

        mul_temp_real_0_name = "mul_temp_real_0_" + str(i)
        mul_temp_real_1_name = "mul_temp_real_1_" + str(i)
        mul_temp_real_name   = "mul_temp_real_" + str(i)

        mul_temp_imag_0_name = "mul_temp_imag_0_" + str(i)
        mul_temp_imag_1_name = "mul_temp_imag_1_" + str(i)
        mul_temp_imag_name   = "mul_temp_imag_" + str(i)

        array_cos_name = "array_cos_wire_" + str(i)
        array_sin_name = "array_sin_wire_" + str(i)

        x_real_name = "X_real_" + str(i)
        x_imag_name = "X_imag_" + str(i)

        Multiplication_V(mul_temp_real_0_name, array_cos_name, x_real_name)
        Multiplication_V(mul_temp_real_1_name, array_sin_name, x_imag_name)

        Substraction_V(mul_temp_real_name, mul_temp_real_0_name, mul_temp_real

        Multiplication_V(mul_temp_imag_0_name, array_sin_name, x_real_name)
        Multiplication_V(mul_temp_imag_1_name, array_cos_name, x_imag_name)

        Addition_V(mul_temp_imag_name, mul_temp_imag_0_name, mul_temp_imag_1_n

        Addition_V(current_ele_name_real, prev_ele_name_real, mul_temp_real_na
        Addition_V(current_ele_name_imag, prev_ele_name_imag, mul_temp_imag_na
```

✅ Requirement Checklist

✔ String-based variable names only

✔ No array indexing

✔ No result assignment from function calls

✔ Explicit string concatenation (`"a_" + str(i)`)

✔ Exactly **two operands + one result** per function

✔ No new variable naming conventions introduced

✔ Explicit name generation for every operation

✔ Correct function usage and ordering

If you want, I can now:

- Add **If_V / Else_V** logic if needed later
- Auto-generate this code from a higher-level spec
- Validate naming consistency programmatically