

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)) 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 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 stated requirements.
 I keep it **procedural / pseudo-Python**, focus on **string-based variable names**, **no array
 indexing**, **no assignment of function results**, and **only function calls with explicit
 result names**.

There is **no conditional logic** in your original code, so **If_V / Else_V** are **not used**.

Transformed Code

```
python
```

```
element_base = "temp_"
```

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

```
i = 0
```

```
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)
```

```
temp_real_name = element_base + "real_" + str(i)
temp_imag_name = element_base + "imag_" + str(i)
```

```
Multiplication_V(
    mul_temp_real_0_name,
    "array_cos_wire_" + str(i),
    "X_real_" + str(i)
)
```

```
Multiplication_V(
    mul_temp_real_1_name,
    "array_sin_wire_" + str(i),
    "X_imag_" + str(i)
)
```

```
Substraction_V(
    mul_temp_real_name,
    mul_temp_real_0_name,
    mul_temp_real_1_name
)
```

```
Addition_V(
    temp_real_name,
    "0",
    mul_temp_real_name
)
```

```
Multiplication_V(
    mul_temp_imag_0_name,
    "array_sin_wire_" + str(i),
    "X_real_" + str(i)
)
```

```
Multiplication_V(
    mul_temp_imag_1_name,
    "array_cos_wire_" + str(i),
```

```

        "X_imag_" + str(i)
    )

Addition_V(
    mul_temp_imag_name,
    mul_temp_imag_0_name,
    mul_temp_imag_1_name
)

Addition_V(
    temp_imag_name,
    "0",
    mul_temp_imag_name
)

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

for i in range(1, 32):

    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)

    temp_real_name       = element_base + "real_" + str(i)
    temp_real_prev_name = element_base + "real_" + str(i - 1)

    temp_imag_name       = element_base + "imag_" + str(i)
    temp_imag_prev_name = element_base + "imag_" + str(i - 1)

    Multiplication_V(

```



ChatGPT ▾

[Log in](#)[Sign up for free](#)

```

        "array_sin_wire_" + str(i),
        "X_real_" + str(i)
    )

Multiplication_V(

```

```

    Multiplication_V(
        mul_temp_real_1_name,
        "array_sin_wire_" + str(i),
        "X_imag_" + str(i)
    )

```

```
)
```

```
Subtraction_V(  
    mul_temp_real_name,  
    mul_temp_real_0_name,  
    mul_temp_real_1_name  
)
```

 Copy code

```
Multiplication_V(  
    mul_temp_imag_0_name,  
    "array_sin_wire_" + str(i),  
    "X_real_" + str(i)  
)
```

```
Multiplication_V(  
    mul_temp_imag_1_name,  
    "array_cos_wire_" + str(i),  
    "X_imag_" + str(i)  
)
```









```
Addition_V(  
    mul_temp_imag_name,  
    mul_temp_imag_0_name,  
    mul_temp_imag_1_name  
)
```

```
Addition_V(  
    temp_real_name,  
    temp_real_prev_name,  
    mul_temp_real_name  
)
```

```
Addition_V(  
    temp_imag_name,  
    temp_imag_prev_name,  
    mul_temp_imag_name  
)
```

✓ Requirement Checklist

- ✓ String concatenation for all variable names
- ✓ No array indexing

-  No brackets used
 -  Dynamic name generation via `"name_" + str(i)`
 -  No assignments from function calls
 -  Only `Multiplication_V`, `Subtraction_V`, `Addition_V` used
 -  One result + two operands per function
 -  No new variable naming conventions introduced
 -  Explicit variable-name generation for every operation
 -  Conditional functions not used (no conditionals present)
-

If you want the **same transformation including an `If_V` / `Else_V` example**, or want this **auto-generated for any loop size**, I can extend it cleanly.

