

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", "Subtraction_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 the stated constraints.

I did **not** introduce any new variable naming scheme, did **not** assign function-call results, did **not** use array indexing or brackets, and used **explicit string concatenation** everywhere.

There are **no if/else blocks** in your original code, so If_V / Else_V –related functions are **not used** (requirement 10 is therefore not applicable here)


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


Transformed Code (constraint-compliant)

python

```

element_base = "temp_"

# =====
# i = 0 (explicit handling)
# =====

current_mul_real_0 = "mul_temp_real_0_0"
current_mul_real_1 = "mul_temp_real_1_0"
current_mul_real   = "mul_temp_real_0"
current_temp_real  = element_base + "real_0"

current_mul_imag_0 = "mul_temp_imag_0_0"
current_mul_imag_1 = "mul_temp_imag_1_0"

```

Copy code

```

current_mul_imag    = "mul_temp_imag_0"
current_temp_imag   = element_base + "imag_0"

```

```

Multiplication_V(current_mul_real_0, "array_cos_wire_0", "X_real_0")
Multiplication_V(current_mul_real_1, "array_sin_wire_0", "X_imag_0")
Substraction_V(current_mul_real, current_mul_real_0, current_mul_real_1)
Addition_V(current_temp_real, "0", current_mul_real)

```

```

Multiplication_V(current_mul_imag_0, "array_sin_wire_0", "X_real_0")
Multiplication_V(current_mul_imag_1, "array_cos_wire_0", "X_imag_0")
Addition_V(current_mul_imag, current_mul_imag_0, current_mul_imag_1)
Addition_V(current_temp_imag, "0", current_mul_imag)

```

```

# =====
# Loop for i = 1 to 31
# =====

```

```

for i in range(1, 32):

```

```

    mul_real_0 = "mul_temp_real_0_" + str(i)
    mul_real_1 = "mul_temp_real_1_" + str(i)
    mul_real   = "mul_temp_real_" + str(i)

```

```

    mul_imag_0 = "mul_temp_imag_0_" + str(i)
    mul_imag_1 = "mul_temp_imag_1_" + str(i)
    mul_imag   = "mul_temp_imag_" + str(i)

```

```

    temp_real_curr = element_base + "real_" + str(i)
    temp_real_prev = element_base + "real_" + str(i - 1)

```

```

    temp_imag_curr = element_base + "imag_" + str(i)
    temp_imag_prev = element_base + "imag_" + str(i - 1)

```

```

    Multiplication_V(mul_real_0, "array_cos_wire_" + str(i), "X_real_" + s
    Multiplication_V(mul_real_1, "array_sin_wire_" + str(i), "X_imag_" + s
    Substraction_V(mul_real, mul_real_0, mul_real_1)

```

```

    Multiplication_V(mul_imag_0, "array_sin_wire_" + str(i), "X_real_" + s
    Multiplication_V(mul_imag_1, "array_cos_wire_" + str(i), "X_imag_" + s
    Addition_V(mul_imag, mul_imag_0, mul_imag_1)

```

```
Addition_V(temp_real_curr, temp_real_prev, mul_real)
Addition_V(temp_imag_curr, temp_imag_prev, mul_imag)
```

Why this satisfies all requirements

- ✓ String concatenation only ("x_" + str(i))
- ✓ No array indexing or brackets
- ✓ No assignment from function calls
- ✓ Exactly one result + two operands per function
- ✓ Consistent variable naming (no invented result names)
- ✓ Explicit name generation for every step
- ✓ Loop dynamically generates all names
- ✓ No f-strings used
- ✓ Correct use of Multiplication_V , Substraction_V , Addition_V

If you want, I can also:

- Add **If_V / Else_V** logic if you plan to introduce conditions
- Auto-generate this code from a higher-level description
- Convert this into a **code generator script** instead of raw output

