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) * b(i)
    x(i) = temp(i) + x(i-1)
```

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

(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_a = "a_" + str(i)).

Below is one valid transformation that satisfies **all 8 requirements**.

I assume the function signatures are of the form:

- `Multiplication_V(destination_var, operand1_var, operand2_var)`
- `Addition_V(destination_var, operand1_var, operand2_var)`

This matches your constraint that **results are not assigned**, but are instead written into explicitly named variables such as `"temp_0"` and `"x_0"` .

✅ Transformed Code

python Copy code

```python
# i = 0 (explicitly written out)
wire_a = "a_" + str(0)
wire_b = "array_b_wire_" + str(0)
wire_temp = "temp_" + str(0)
wire_x = "x_" + str(0)

Multiplication_V(wire_temp, wire_a, wire_b)
Addition_V(wire_x, wire_temp, "0")

# Loop for i = 1 to 15
for i in range(1, 16):
    wire_a = "a_" + str(i)
    wire_b = "b_" + str(i)
    wire_temp = "temp_" + str(i)
    wire_x = "x_" + str(i)
    wire_x_prev = "x_" + str(i - 1)

    Multiplication_V(wire_temp, wire_a, wire_b)
    Addition_V(wire_x, wire_temp, wire_x_prev)
```

🔍 How this meets your requirements

1. **String concatenation for variable names**
   ✔ Uses `"a_" + str(i)` , `"x_" + str(i)` , etc.
2. **Array indexing replaced with function calls**
   ✔ No `a(i)` or `x(i)` remain; all operations use `Multiplication_V` and `Addition_V` .

3. **No brackets for indexing**

   ✓ No `[]` or `()` used for indexing.

4. **Explicit string concatenation (no f-strings)**

   ✓ Uses `"name_" + str(i)` everywhere.

5. **No assignment of function call results**

   ✓ No `=` with `Multiplication_V` or `Addition_V`.

6. **Functions called directly with generated names**

   ✓ Variable names are passed as strings directly to the functions.

7. **Consistent variable naming**

   ✓ Uses `temp_i`, `x_i`, `a_i`, `b_i`, `array_b_wire_0` only—no new names introduced.

8. **Explicit variable-name generation per operation**

   ✓ Each loop iteration explicitly defines `wire_a`, `wire_b`, `wire_temp`, etc.

If you want, I can also:

- Unroll the loop completely (i = 0 … 15)
- Adapt this to a specific HDL / codegen backend
- Change the function signatures if your API is different