Excellent — now you're at step **5 of `parse_spd`**, where the parser reads the **via connectivity lines**. Let's document `_extract_via_lines`.

📘 Documentation

`_extract_via_lines(text: str) -> List[Tuple[str, str]]`

Purpose

Extract all **via definitions** from the SPD file and return them as a list of node pairs:

```
cssCopy code[(upper_node, lower_node), ...]
```

Each pair tells you which **two nodes** a via connects between (top and bottom of the via span).
Node names are **canonicalized** (`Node013` → `Node13`).

How it works

1. Regex pattern

```
pythonCopy codepattern = (
    r"Via\d+(?:::\w+)?\s+"                  # via name, e.g. "Via12::Through"
    r"UpperNode\s*=\s*(Node\d+)(?:::\w+)?\s+" # upper node, e.g. "Node013::PAD"
    r"LowerNode\s*=\s*(Node\d+)(?:::\w+)?"
)
```

This matches lines like:

```
phpCopy codeVia12::Through UpperNode = Node013::PAD LowerNode = Node7::VIA
```

- `Via\d+` → matches via ID (`Via12`).
- `(?:::\w+)?` → allows optional qualifiers (`::Through`).
- `UpperNode = Node013::PAD` → extracts `Node013`.
- `LowerNode = Node7::VIA` → extracts `Node7`.

---

2. Find all matches

```
pythonCopy codepairs = re.findall(pattern, text, flags=re.IGNORECASE)
```

Returns a raw list of `(upper_node, lower_node)` strings. Example:

```
pythonCopy code[("Node013", "Node7"), ("Node11", "Node12"), ...]
```

---

3. Canonicalize node names

```
pythonCopy codereturn [(canon_node(u), canon_node(l)) for (u, l) in
pairs]
```

Applies `canon_node` → strips leading zeros (`Node013 → Node13`).

Final result example:

```
pythonCopy code[("Node13", "Node7"), ("Node11", "Node12"), ...]
```

Example from `b4_1.spd`

SPD snippet:

```
javaCopy codeVia12::Through  UpperNode = Node013::PAD  LowerNode =
Node7::VIA
Via23           UpperNode = Node11        LowerNode = Node12
```

Parsed result:

```
pythonCopy code[
  ("Node13", "Node7"),
  ("Node11", "Node12")
]
```

Why it matters

- The SPD file defines vias as **connections between nodes**.
- These `(upper_node, lower_node)` pairs are later used in `_extract_start_stop_type` to figure out:
    - Which **layer** a via starts on.
    - Which **layer** it stops on.
    - The **type** of via (PWR or GND).

This is what eventually builds:

- `brd.start_layers`
- `brd.stop_layers`
- `brd.via_type`

Summary:
`_extract_via_lines` finds every via in the SPD file and records the pair of nodes it connects. This gives the raw **top–bottom connectivity** of all vias, which is later mapped into actual layer spans and types for simulation.

Do you want me to continue with **`_extract_start_stop_type`**, which consumes these via pairs and generates the `start_layers`, `stop_layers`, and `via_type` arrays?