Nice — you're now inside **step 2 of `parse_spd`**: extracting the **nodes** (the via locations, their net type, and which layer they belong to).

Let's dissect `_extract_nodes` carefully.

---

## 1. The regex pattern

```
pythonCopy codepattern = (
    r"(Node\d+)"                  # raw node name (e.g. "Node013")
    r"(?:::)?([A-Za-z0-9_]+)?"    # optional "::NET" (e.g. "::GND" or
"::PWR1")
    r"\s+X\s*=\s*([-\d\.eE\+]+)mm"
    r"\s+Y\s*=\s*([-\d\.eE\+]+)mm"
    r"\s+Layer\s*=\s*Signal(\d+)"
)
```

It matches lines like:

```
cppCopy codeNode10::GND X=28.000000mm Y=30.000000mm Layer=Signal1
```

Captures:

- `(Node\d+)` → `"Node10"`
- `([A-Za-z0-9_]+)?` → `"GND"`
- `([-\d\.eE\+]+)` → `28.0`
- `([-\d\.eE\+]+)` → `30.0`
- `(\d+)` → `1`

---

## 2. Loop over matches

```
pythonCopy codefor raw, tag, x_str, y_str, layer_str in node_lines:
    tag_l = (tag or "").lower()
    is_power = tag_l.startswith("pwr")
    info = {
        "type": 1 if is_power else 0,
        "net":  tag_l if tag_l else None,
        "x": float(x_str), "y": float(y_str),    # still in mm
        "layer": int(layer_str)                  # 1-based Signal##
    }
    node_info[canon_node(raw)] = info
```

- `tag_l`: the net name in lowercase (`"gnd"`, `"pwr"`, `"pwr2"`, etc.).
- `is_power`: checks if net starts with `"pwr"`.
    - If yes → `"type": 1` (power via).
    - If no → `"type": 0` (ground or empty).
- `x`, `y`: coordinate values in **mm** (conversion to meters happens later).
- `layer`: the Signal index (1, 2, 3, ...).

The key is the **canonical node name**:

```
pythonCopy codecanon_node("Node013")  → "Node13"
```

(removes leading zeros).

---

## 3. Example from your `b4_1.spd`

SPD lines:

```
makefileCopy codeNode10::GND  X=28.000000mm  Y=30.000000mm
Layer=Signal1
Node11::GND  X=28.000000mm  Y=30.000000mm  Layer=Signal2
Node12::PWR  X=25.000000mm  Y=25.000000mm  Layer=Signal3
Node13::PWR  X=25.000000mm  Y=25.000000mm  Layer=Signal5
Node14::GND  X=28.000000mm  Y=25.000000mm  Layer=Signal4
```

Parsed output:

```
pythonCopy code{
 'Node10': {'type': 0, 'net': 'gnd', 'x': 28.0, 'y': 30.0, 'layer':
1},
 'Node11': {'type': 0, 'net': 'gnd', 'x': 28.0, 'y': 30.0, 'layer':
2},
 'Node12': {'type': 1, 'net': 'pwr', 'x': 25.0, 'y': 25.0, 'layer':
3},
 'Node13': {'type': 1, 'net': 'pwr', 'x': 25.0, 'y': 25.0, 'layer':
5},
 'Node14': {'type': 0, 'net': 'gnd', 'x': 28.0, 'y': 25.0, 'layer':
4},
 ...
}
```

This matches exactly what you saw in the log:

```
csharpCopy code[SPD] Nodes extracted:
 14 examples:
 [('Node10', {'type': 0, 'net': 'gnd', 'x': 28.0, 'y': 30.0, 'layer':
1}),
  ('Node11', {'type': 0, 'net': 'gnd', 'x': 28.0, 'y': 30.0, 'layer':
2}),
  ('Node12', {'type': 1, 'net': 'pwr', 'x': 25.0, 'y': 25.0, 'layer':
3}),
  ('Node13', {'type': 1, 'net': 'pwr', 'x': 25.0, 'y': 25.0, 'layer':
5}),
  ('Node14', {'type': 0, 'net': 'gnd', 'x': 28.0, 'y': 25.0, 'layer':
4})]
```

---

## 4. Why this matters

This `node_info` dict is crucial because it's the **master table of vias**:

- Where each via is located (`x`, `y`).
- Which layer it connects to.
- Whether it's **PWR** or **GND**.
- Which net it belongs to.

Later steps use this dictionary to:

- Build start/stop layer arrays (`start_layers`, `stop_layers`).
- Separate IC vs decap vias.
- Assign them to ports.

---

 **Summary:**
`_extract_nodes` scans for `NodeXX` lines in the SPD file, and builds a structured dictionary mapping each node to its **type, net, coordinates (mm), and layer**. This is the foundation for figuring out which vias exist and how they connect into the PDN model.

---

Do you want me to also show how these nodes later get turned into **IC vias vs decap vias** in `_fill_ic_decap_vias`? That's the next step after extracting them.