Perfect — this is step **6 of `parse_spd`**, where the parser combines everything so far (nodes, via lines, IC/decap blocks) to build the **start layer, stop layer, and type arrays** for all vias. Let's document `_extract_start_stop_type`.

Documentation

`_extract_start_stop_type(via_lines, node_info, ic_blocks, decap_blocks)`

Purpose

Generate three aligned arrays that describe all **vias** in the board, preserving the `.Connect` order:

- **`start_layers`**: signal layer index where each via begins (1-based from SPD).
- **`stop_layers`**: signal layer index where each via ends (1-based).
- **`via_type`**: `1` if either end belongs to a **PWR** node, else `0` (GND).

How it works

1. Find a via by node

```
pythonCopy codedef find_via_by_node(node):
    for upper, lower in via_lines:
        if upper == node or lower == node:
            if (upper in node_info) and (lower in node_info):
                return (node_info[upper]['layer'],
                        node_info[lower]['layer'],
                        upper, lower)
```

- Looks up the first `(upper_node, lower_node)` pair in `via_lines` that matches this node.
- Uses `node_info` to find the **layer numbers** of both nodes.
- Returns `(start_layer, stop_layer, upper_name, lower_name)`.

---

2. Extract node names from `.Connect` blocks

```
pythonCopy codedef _nodes_from_block(block):
    plus_nodes, minus_nodes = [], []
    for ln in block.splitlines():
        if ln.startswith("1") or ln.startswith("2"):
            node = canon_node(m.group(1))  # e.g. "Node013" → "Node13"
            if ln.startswith("1"): plus_nodes.append(node)
            else: minus_nodes.append(node)
    return plus_nodes, minus_nodes
```

- Reads the body of each `.Connect` block (IC or decap).
- Collects node names for the **plus side (1 …)** and **minus side (2 …)**, preserving order.

---

3. Process all blocks

```
pythonCopy codedef process_blocks(blocks):
    results = []
    for blk in blocks:
        plus_nodes, minus_nodes = _nodes_from_block(blk)
        for p_node, m_node in zip(plus_nodes, minus_nodes):
            # + via
            p_result = find_via_by_node(p_node)
            if p_result:
                start, stop, u, l = p_result
                typ = 1 if (node_info[u]['type']==1 or node_info[l]
['type']==1) else 0
                results.append((start, stop, typ))
            # - via
            m_result = find_via_by_node(m_node)
            if m_result:
                start, stop, u, l = m_result
                typ = 1 if (node_info[u]['type']==1 or node_info[l]
['type']==1) else 0
                results.append((start, stop, typ))
    return results
```

- For each port block, pairs **plus and minus nodes**.
- Finds their via connectivity from `via_lines`.
- Determines via type:

    - `1` if either endpoint is **PWR**.
    - `0` if both are **GND**.
- Appends `(start_layer, stop_layer, type)` to the result list.

---

4. Merge IC and decap entries

```
pythonCopy codeic_entries    = process_blocks(ic_blocks)
decap_entries = process_blocks(decap_blocks)
all_entries   = ic_entries + decap_entries
```

- Preserves strict ordering: all IC vias first, then all decap vias.

---

5. Build numpy arrays

```
pythonCopy codestart_layers  = np.array([e[0] for e in all_entries], dtype=int)
stop_layers   = np.array([e[1] for e in all_entries], dtype=int)
via_type      = np.array([e[2] for e in all_entries], dtype=int)
```

- Final aligned arrays of via start/stop layers and types.
- Returned to the caller.

How it's used in `parse_spd`

```
pythonCopy codesl, tl, vt = _extract_start_stop_type(via_lines, node_info, ic_blocks, decap_blocks)

brd.start_layers = np.asarray(sl, np.int32) - 1
brd.stop_layers  = np.asarray(tl, np.int32) - 1
brd.via_type     = np.asarray(vt, np.int32)
```

- Subtracts `1` → convert from SPD's **1-based `Signal##` layer index** to Python's **0-based indexing**.
- Stores arrays inside the board object (`brd`).

Example

SPD excerpts:

```
phpCopy codeVia12::Through  UpperNode = Node12::PAD  LowerNode =
Node10::VIA
Via13           UpperNode = Node14       LowerNode = Node16

.Connect ic_port1
1 $Package.Node12
2 $Package.Node10
.EndC

.Connect decap_port1
1 $Package.Node14
2 $Package.Node16
.EndC
```

From `node_info`:

```
bashCopy codeNode10 → {'layer': 1, 'type': 0, 'net': 'gnd'}
Node12 → {'layer': 3, 'type': 1, 'net': 'pwr'}
Node14 → {'layer': 4, 'type': 0, 'net': 'gnd'}
Node16 → {'layer': 5, 'type': 1, 'net': 'pwr'}
```

Resulting arrays:

```
pythonCopy codebrd.start_layers = [2, 0, 3, 4]   # after subtracting 1
brd.stop_layers  = [0, 2, 4, 3]
brd.via_type     = [1, 1, 1, 1]   # all involve at least one PWR
```

Why it matters

- This step encodes **which layers each via spans** and **what kind of via it is (PWR/GND)**.
- These arrays are critical inputs for later steps (BEM inductance extraction, node-voltage method).

Summary:
`_extract_start_stop_type` converts raw via connectivity into structured arrays: start layers, stop layers, and type flags, aligned in IC+decap order. These arrays define the electrical role and vertical span of every via in the PDN model.

Would you like me to now continue documenting **`_fill_buried_vias`**, since that comes right after this (step 7 in `parse_spd`)?