In [1]:
```
import qiskit_metal as metal
from qiskit_metal import designs
from qiskit_metal.qlibrary.qubits.transmon_pocket import TransmonPocket
from qiskit_metal.qlibrary.tlines.meandered import RouteMeander
```

In [2]:
```
design = designs.DesignPlanar()
gui = metal.MetalGUI(design)
print("Qiskit metal is here")
```

```
Qiskit metal is here
```

In [21]:
```
# የ Qubit መለኪያዎች (Parameters)
# እነዚህን በመቀየር የ qubit ን ባህሪ መቆጣጠር ትችላለህ
# ይህ መስመር የቆዩትን ዲዛይኖች በሙሉ ያጠፋል (The Reset Button)
design.delete_all_components()
options = dict(
    pos_x = '0um',
    pos_y = '0um',
    pad_width = '300um', # የ Qubit ሳህን ስፋት
    pad_height = '200um', # የ Qubit ሳህን ቁመት
    pad_gap = '30um'      # በሳህኑ እና በ ground መካከል ያለው ክፍተት
)
options_q2 = dict(
    pos_x= '1000um',
    pos_y= '0um',
    pad_width='300um',
    pad_height='200um',
    pad_gap= '30um'
)
# Q1 የሚባል Qubit መፍጠር
q1 = TransmonPocket(design, 'Q1', options=options)

q2 = TransmonPocket(design, 'Q2', options=options_q2)
# GUIውን አድስና ውጤቱን ተመልከት
gui.rebuild()
gui.autoscale()
```

In [4]:
```
#በቃለ መጠይቁ ወቅት እንዲህ ሊሉህ ይችላሉ፡ "የ qubit ንድፍ (layout) ካለኝ በኋላ ለምን Qiskit Metal

#መልሱ፡ ምክንያቱም በ Qiskit Metal ውስጥ መጠኖችን በቀላሉ መቀየር (Dynamic resizing) ትችላለህ።

#Python
```

In [5]:
```
#አዲስ Q2 ከመፍጠርህ በፊት የነበረውን ማጥፋት ትችላለህ። ይህንን ኮድ መጀመሪያ ተጠቀም፡-
#design.delete_component('Q2')
```

In [6]:
```
#ኢንተርቪው አድራጊው "How do you handle design iterations in code?" (የዲዛይን ለውጦችን በኮድ

#"በ Qiskit Metal ውስጥ design.overwrite_enabled = True የሚለውን በመጠቀም ወይም የድሮውን
```

In [22]:
```
# Cell 2: CPW Bus Connection
design.overwrite_enabled = True
```

```
# ለሶኬቶቹ የሚያስፈልጉ ሙሉ መለኪያዎች (Including the missing cpw_gap)
full_pin_opts = dict(
    pad_width='100um',
    pad_height='30um',
    pad_gap='15um',
    cpw_width='10um',       # የሽቦው ስፋት
    cpw_gap='6um',          # <--- ይህ ነው የጎደለው! (The gap to ground)
    pad_cpw_extent='25um',
    cpw_extend='100um',
    pocket_extent='5um',
    pocket_rise='0um',
    pad_cpw_shift='0um'
)

# ለ Q1 እና Q2 ሶኬቶችን ማያያዝ
q1.options.connection_pads.bus_to_q2 = dict(loc_W=1, loc_H=-1, **full_pin_opts)
q2.options.connection_pads.bus_to_q1 = dict(loc_W=-1, loc_H=-1, **full_pin_opts)

# የ Bus መስመሩን መዘርጋት
bus_options = dict(
    pin_inputs=dict(
        start_pin=dict(component='Q1', pin='bus_to_q2'),
        end_pin=dict(component='Q2', pin='bus_to_q1')
    ),
    fillet='99um',
    total_length='7mm'
)

# መስመሩን መፍጠር
bus = RouteMeander(design, 'Bus_1', options=bus_options)

# GUIውን አድስ
gui.rebuild()
gui.autoscale()
```

```
01:51PM 30s WARNING [__init__]: Pin bus_to_q2 does not exist in component Q1. Bus_1
has not been built. Please check your pin_input values.
```

In [23]:

```
# 1. ፒኖቹ መኖራቸውን ለማረጋገጥ ስማቸውን እንይ
print(f"Q1 pins: {q1.pins.keys()}")
print(f"Q2 pins: {q2.pins.keys()}")

# 2. የ Bus መለኪያዎች (ያለ ምንም ስህተት)
bus_options = dict(
    pin_inputs=dict(
        start_pin=dict(component='Q1', pin='bus_to_q2'),
        end_pin=dict(component='Q2', pin='bus_to_q1')
    ),
    fillet='99um',
    total_length='7mm',
    lead=dict(start_straight='50um', end_straight='50um') # ከፒኑ ወጣ እንዲል
)

# 3. መስመሩን መፍጠር
bus = RouteMeander(design, 'Bus_1', options=bus_options)
```

```
# 4. GUIውን አድስ
gui.rebuild()
gui.autoscale()
```

```
01:53PM 06s INFO [connect_meandered]: Zero meanders for Bus_1
Q1 pins: dict_keys(['bus_to_q2'])
Q2 pins: dict_keys(['bus_to_q1'])
01:53PM 06s INFO [connect_meandered]: Zero meanders for Bus_1
```

In [24]:
```
# GDS ረንደረርን መጥራት
gds_export = design.renderers.gds

# ፋይሉን ማውጣት
gds_export.export_to_gds('quantum_chip_design.gds')

print("የ GDS ፋይል በትክክል ተፈጥሯል! አሁን Cadence Virtuoso ላይ መክፈት ትችላለህ።")
```

```
02:18PM 32s WARNING [import_junction_gds_file]: Not able to find file:"../resources/Fake_Junctions.GDS".  Not used to replace junction. Checked directory:"C:\Users\resources".
የ GDS ፋይል በትክክል ተፈጥሯል! አሁን Cadence Virtuoso ላይ መክፈት ትችላለህ።
```

In [25]:
```
import os
print(f"ፋይሉ እዚህ ቦታ ይገኛል፡- {os.getcwd()}")
```

```
ፋይሉ እዚህ ቦታ ይገኛል፡- C:\Users\Dawit
```

In [26]:
```
from IPython.display import FileLink
FileLink('quantum_chip_design.gds')
```

Out[26]: quantum_chip_design.gds

In [ ]: