FPGA-ROUTE REPORT

寿晨宸 2100012945

ALGORITHM

我实现了 Parallel New Bidirectional A*

伪代码如下：

```
# shared variables
finished=False
solution=None
F1=0.0,F2=0.2,L=MAX
l_par=Lock()

# parallel thread 1
min_heap_s.push(src)
while not finised and not min_heap_s.empty():
    u=min_heap_s.pop()
    if not is_visited(u):
        if (f1(u)<L) and (g1(u)+F2-h2(u)<L):
            for v in u.children():
                if is_visited(v):
                    continue
                g1(v)=g1(u)+d1(v)
                f1(v)=g1(v)+h1(v)
                set_visited_forward(v)
                min_heap_s.push(v)
                if is_visited_backward(v):
                    if g1(v)+g2(v)<L:
                        l_par.lock()
                        if g1(v)+g2(v)<L:
                            L=g1(v)+g2(v)
                            solution=v
                        l_par.unlock()
        set_visited(u)
    if not min_heap_s.empty():
        F1=f1(min_heap_s.top())
    else:
        finished=True

# parallel thread 2
min_heap_t.push(sink)
while not finised and not min_heap_t.empty():
    u=min_heap_t.pop()
    if not is_visited(u):
        if (f2(u)<L) and (g2(u)+F1-h1(u)<L):
            for v in u.parents():
                if is_visited(v):
                    continue
                g2(v)=g2(u)+d2(v)
                f2(v)=g2(v)+h2(v)
                set_visited_backward(v)
                min_heap_s.push(v)
                if is_visited_forward(v):
                    if g1(v)+g2(v)<L:
                        l_par.lock()
                        if g1(v)+g2(v)<L:
                            L=g1(v)+g2(v)
                            solution=v
                        l_par.unlock()
        set_visited(u)
    if not min_heap_t.empty():
        F2=f2(min_heap_t.top())
    else:
        finished=True
```

其中，cost-setting 如下

```
# h1: compute_forward_future_cost
h1(u):wirelength_weight*distance_to_sink/(1+count_connection_of_user(u))
# h2: compute_backward_future_cost
h2(u)=wirelength_weight*distance_to_source/(1+count_connection_of_user(u))
# d1: compute_forward_dis
d1(u)=get_node_cost(u)+(1-
wirelength_weight)*v.len()/(1+count_connection_of_user(u))
# d2: compute_forward_dis
d2(u)=get_node_cost(u)+(1-
wirelength_weight)*v.len()/(1+count_connection_of_user(u))
# update g1&f1: evaluate_forward_cost(u,v)
g1(v)=g1(u)+d1(v)
f1(v)=g1(v)+h1(v)
# update g2&f2: evaluate_backward_cost(u,v)
g2(v)=g2(u)+d2(v)
f2(v)=g2(v)+h2(v)
```

RESULTS

CRITICAL PATH WIRELENGTH

	Serial A*	PNBA*
boom_med_pb	340	321
boom_soc_v2	705	715
corescore_500	462	467
corescore_1700	777	725

	Serial A*	PNBA*
mlcad_d181	603	563
vtr_mcml	255	253

NUM ITERATIONS

	Serial A*	PNBA*
boom_med_pb	10	11
boom_soc_v2	28	20
corescore_500	8	8
corescore_1700	30	18
mlcad_d181	10	10
vtr_mcml	10	11

ROUTE TIME(S)

	Serial A*	PNBA*	Speed Up
boom_med_pb	25.2177	17.7336	1.422x
boom_soc_v2	383.164	196.492	1.950x
corescore_500	58.9812	54.0718	1.091x
corescore_1700	353.687	282.435	1.252x
mlcad_d181	118.146	104.681	1.129x
vtr_mcml	39.7677	28.062	1.412x