

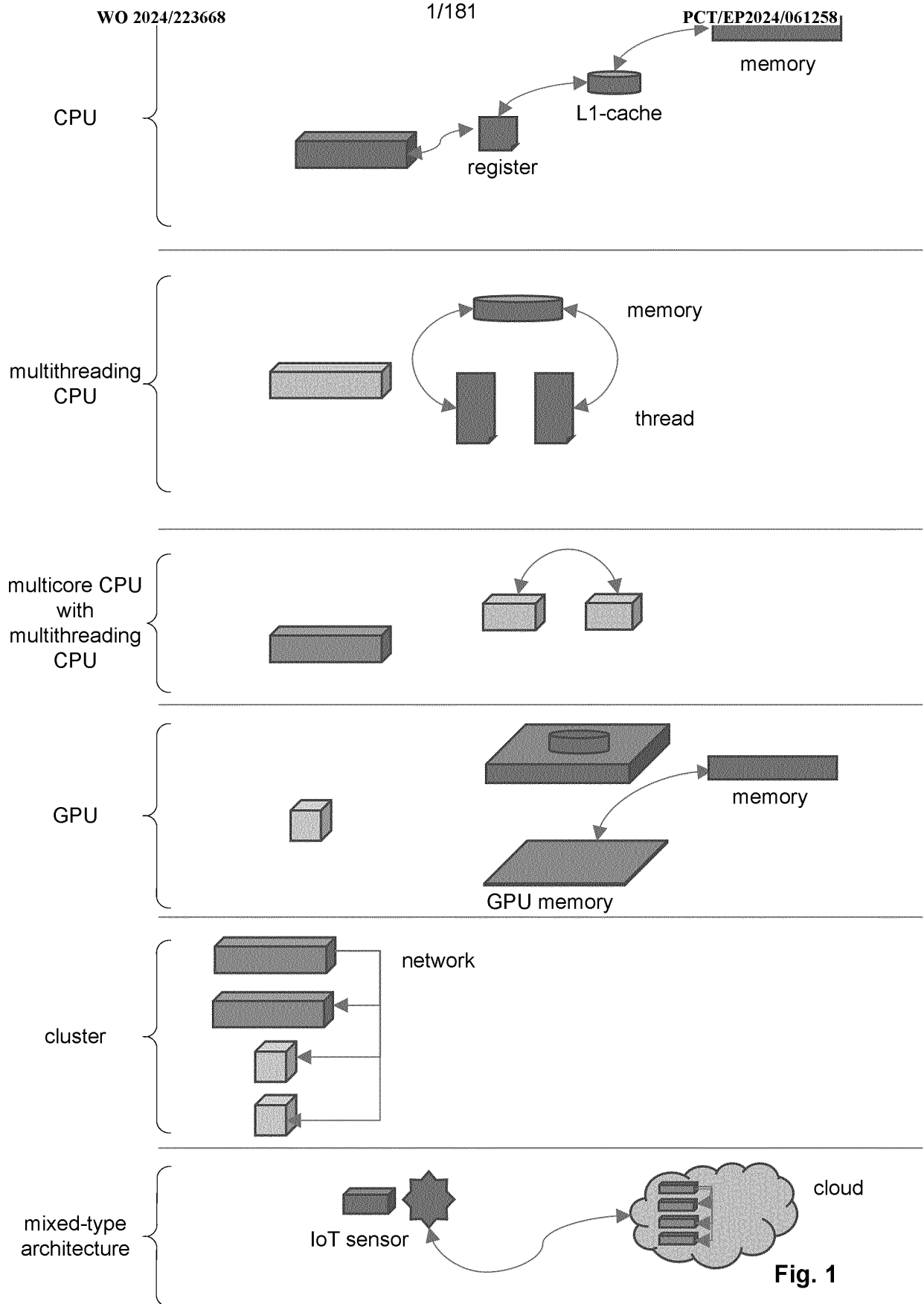


Fig. 1

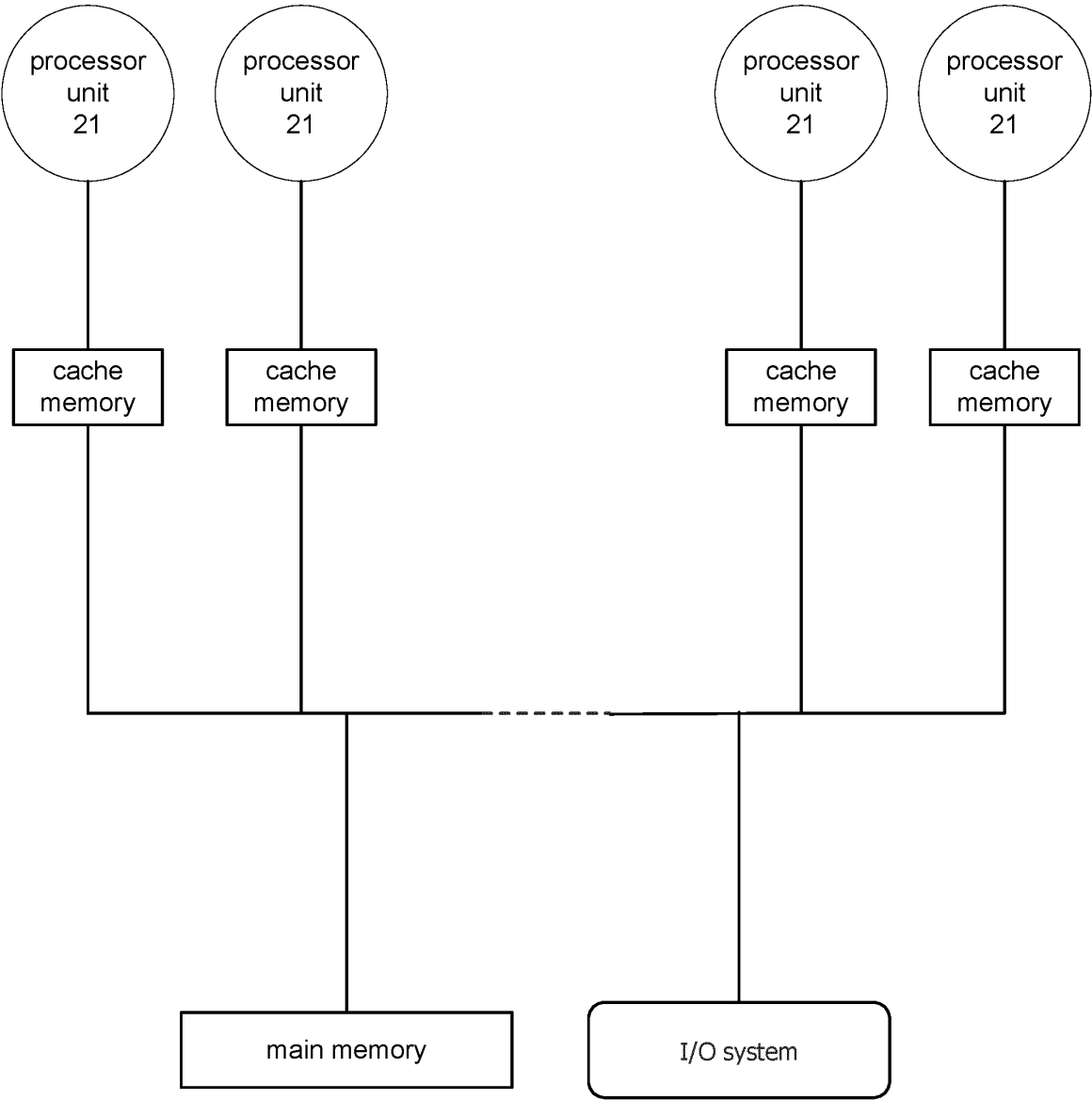


Fig. 2

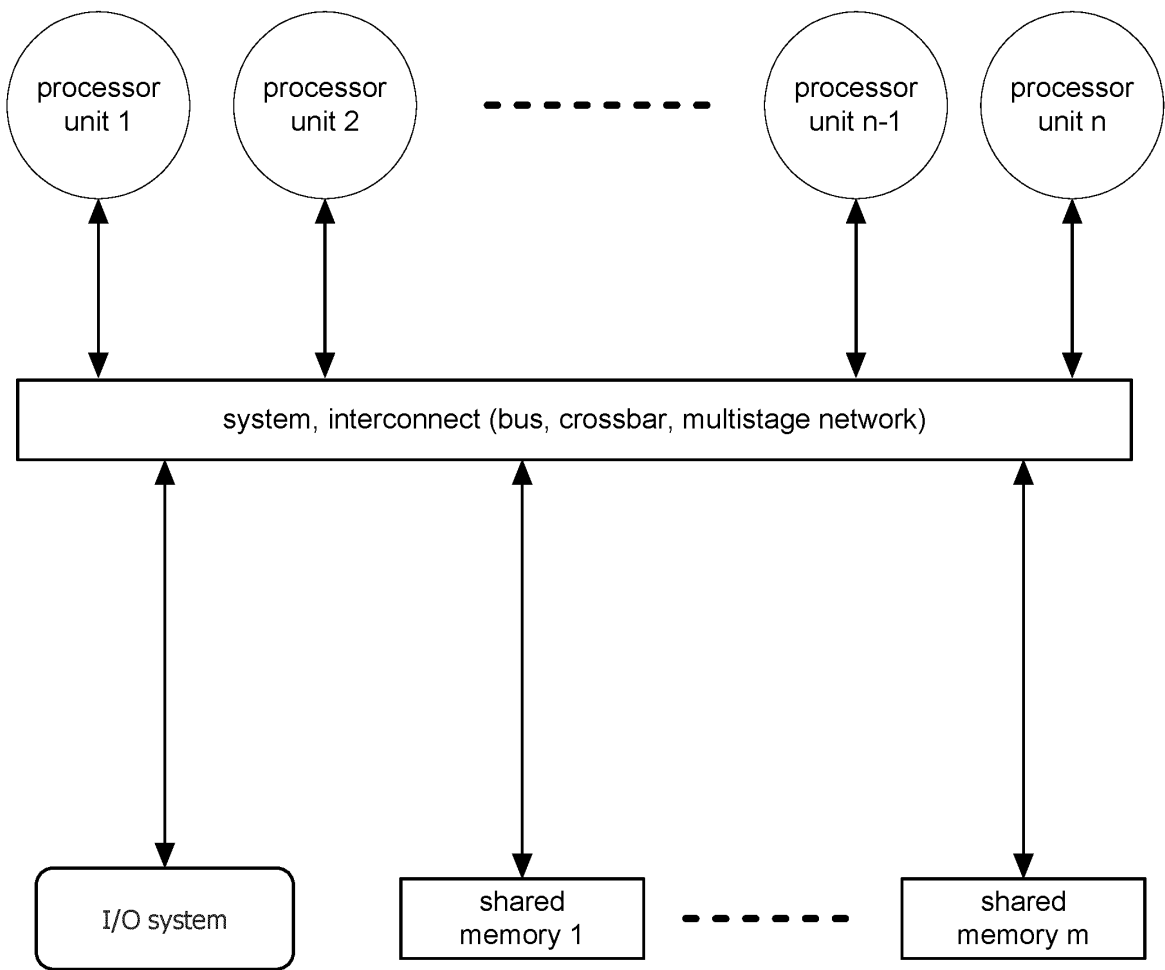


Fig. 3

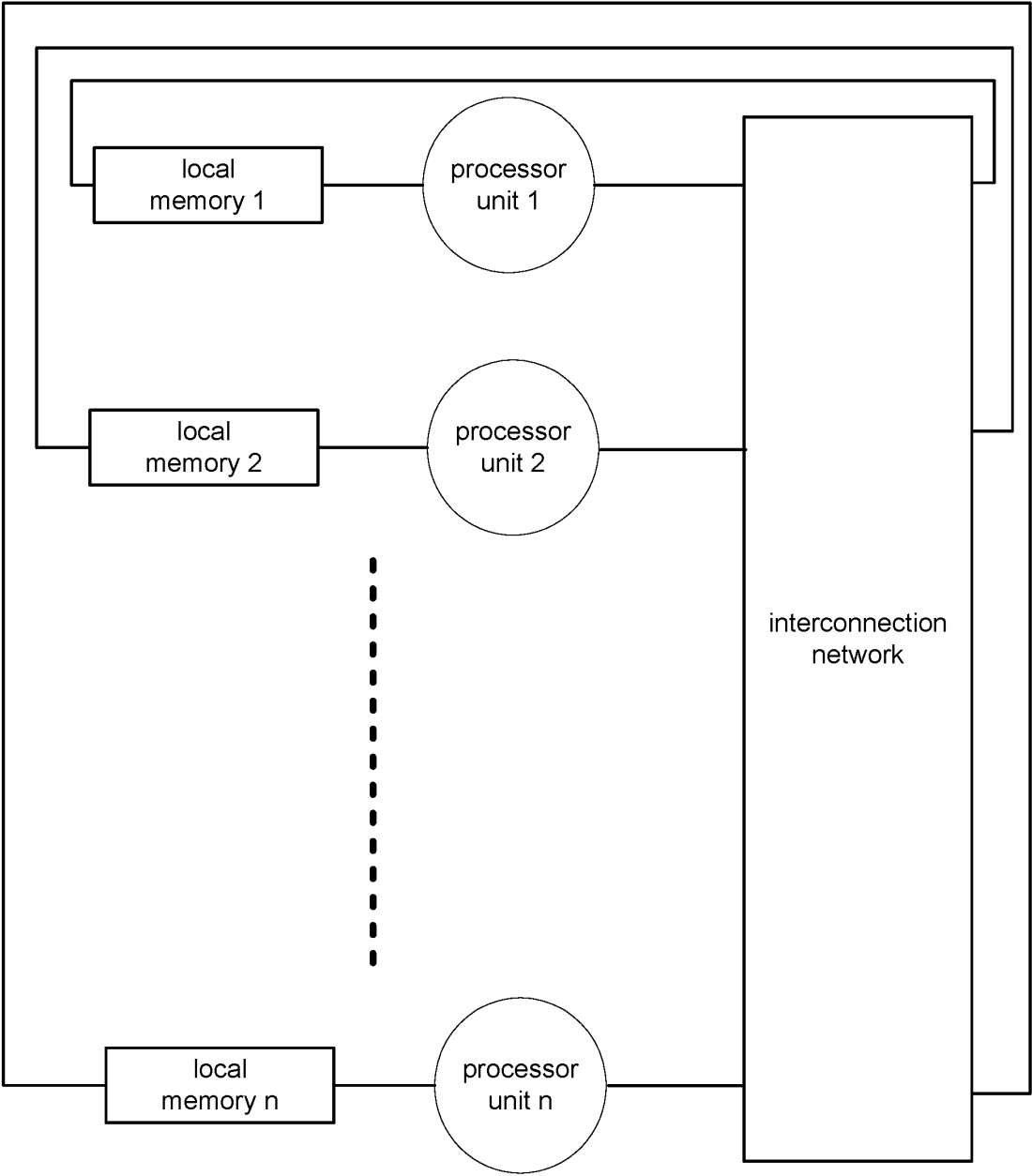


Fig. 4

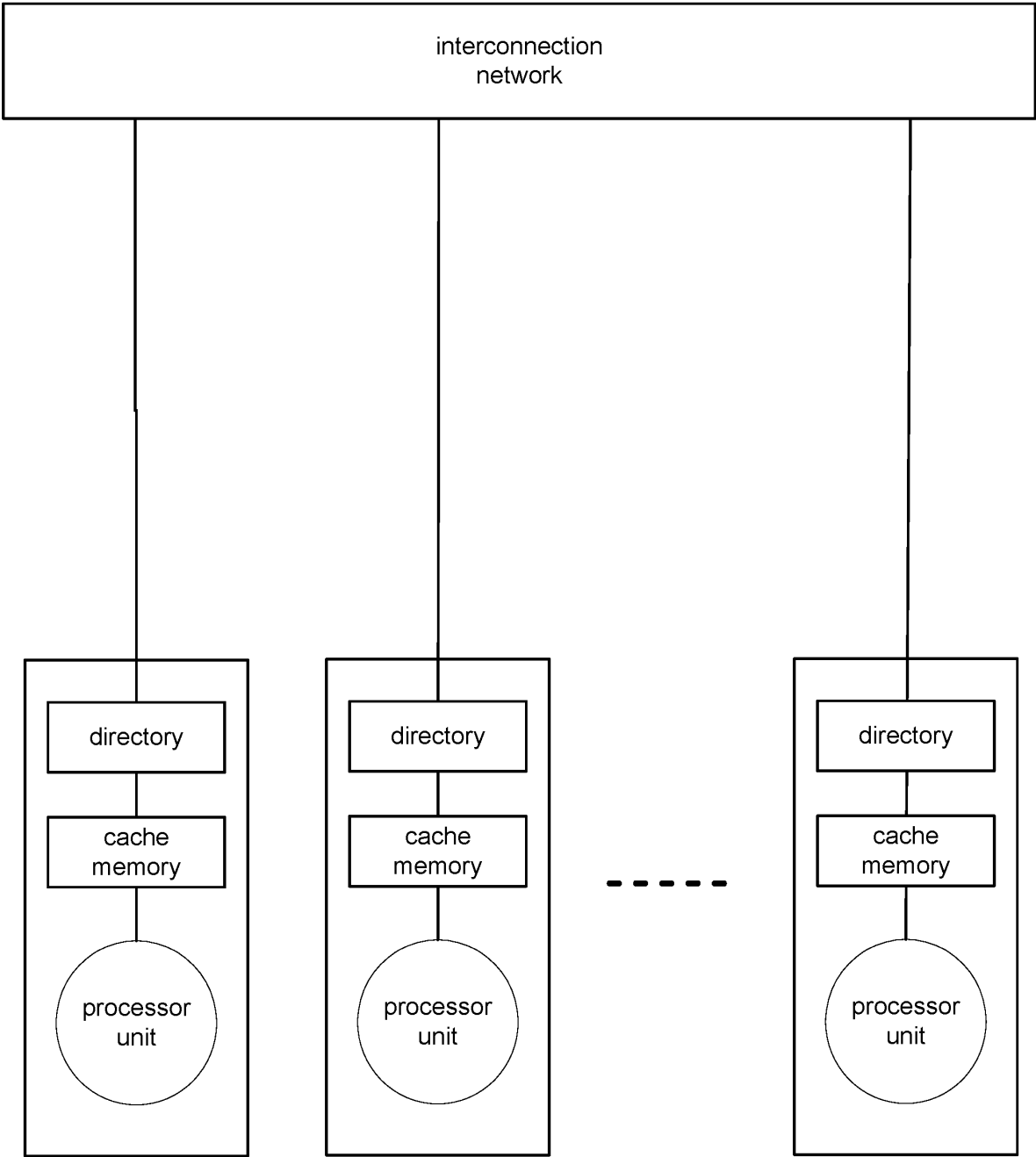


Fig. 5

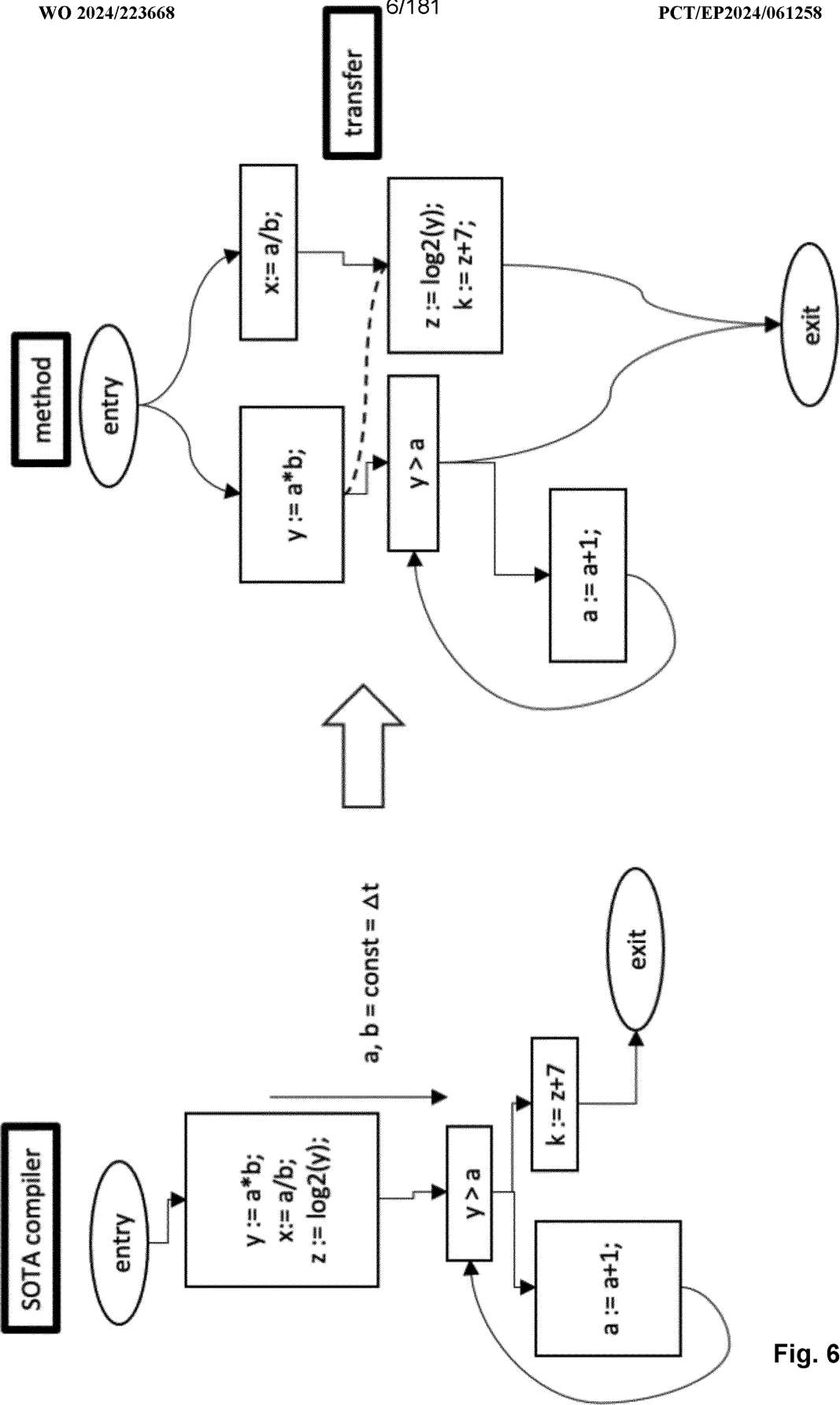


Fig. 6

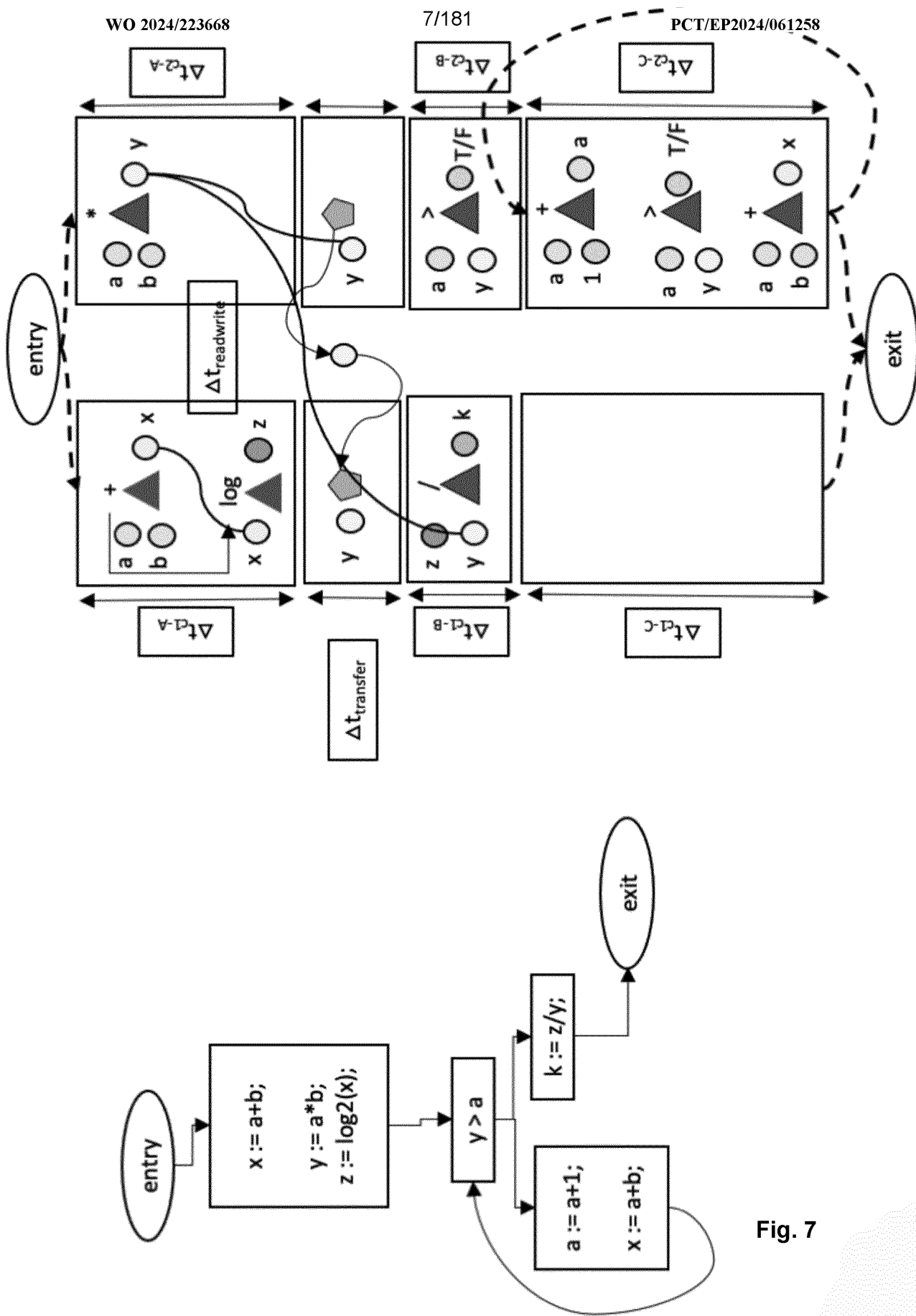


Fig. 7

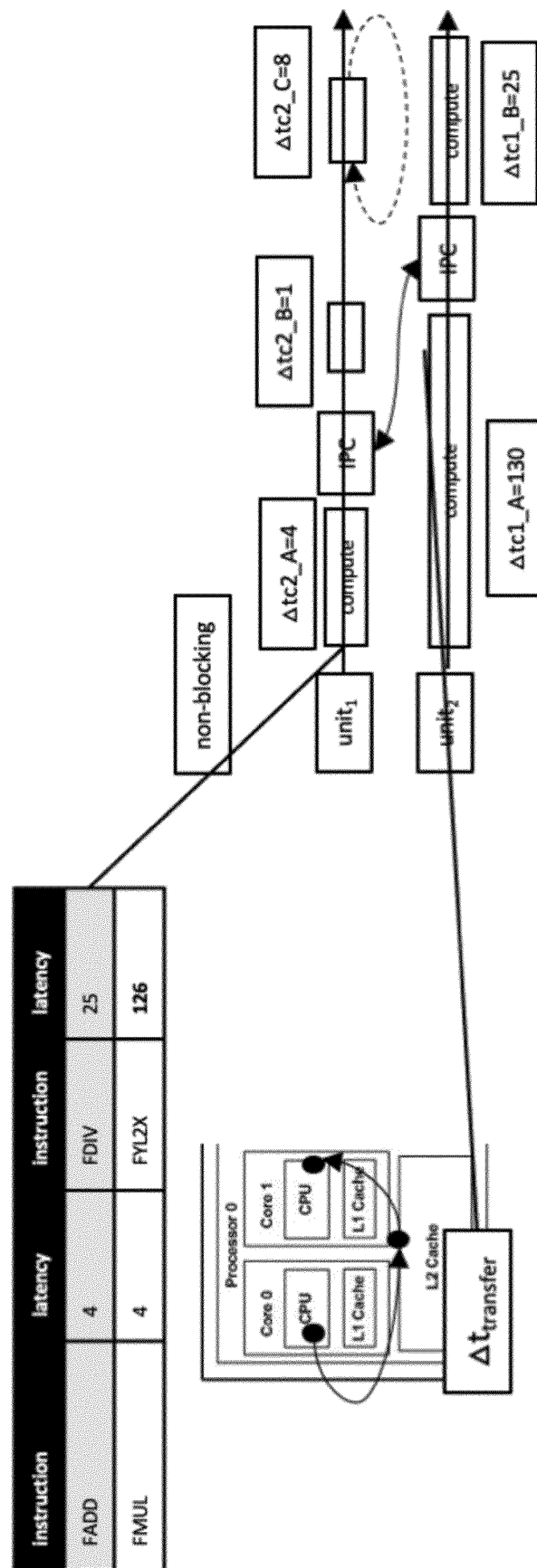


Fig. 8

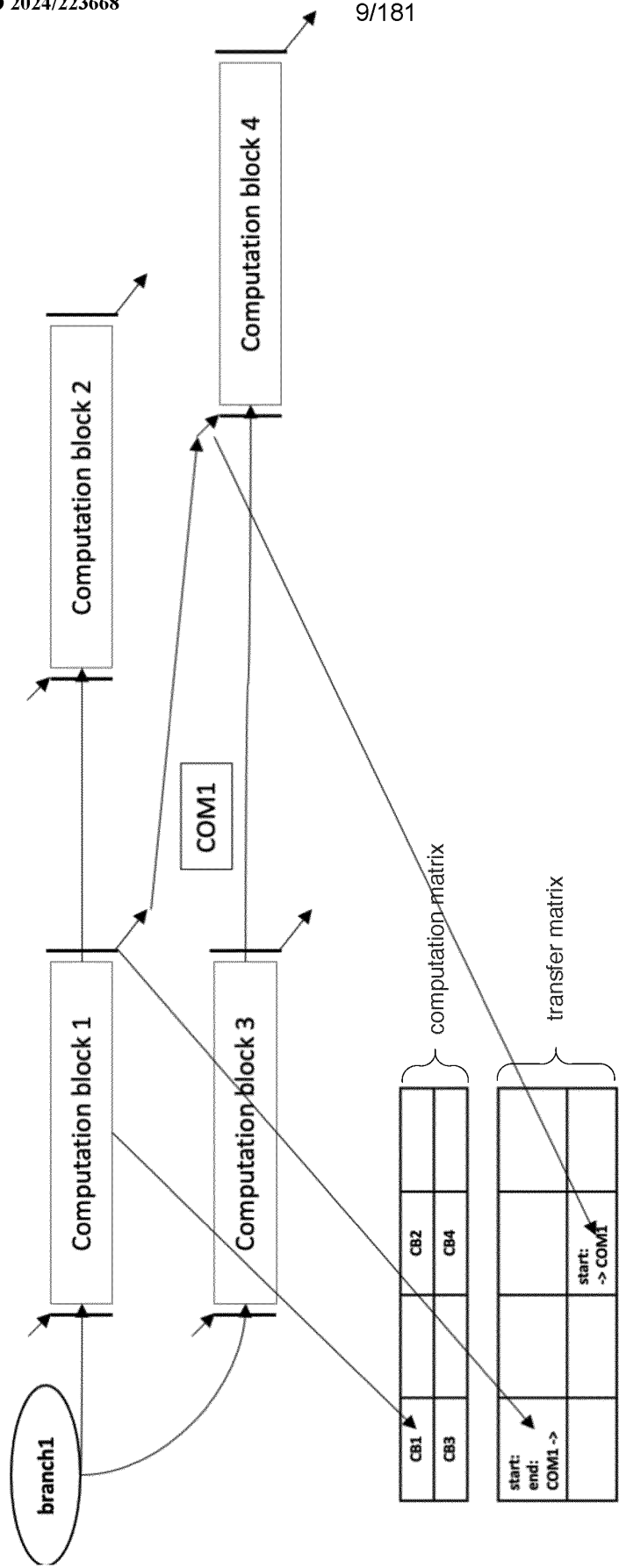


Fig. 9

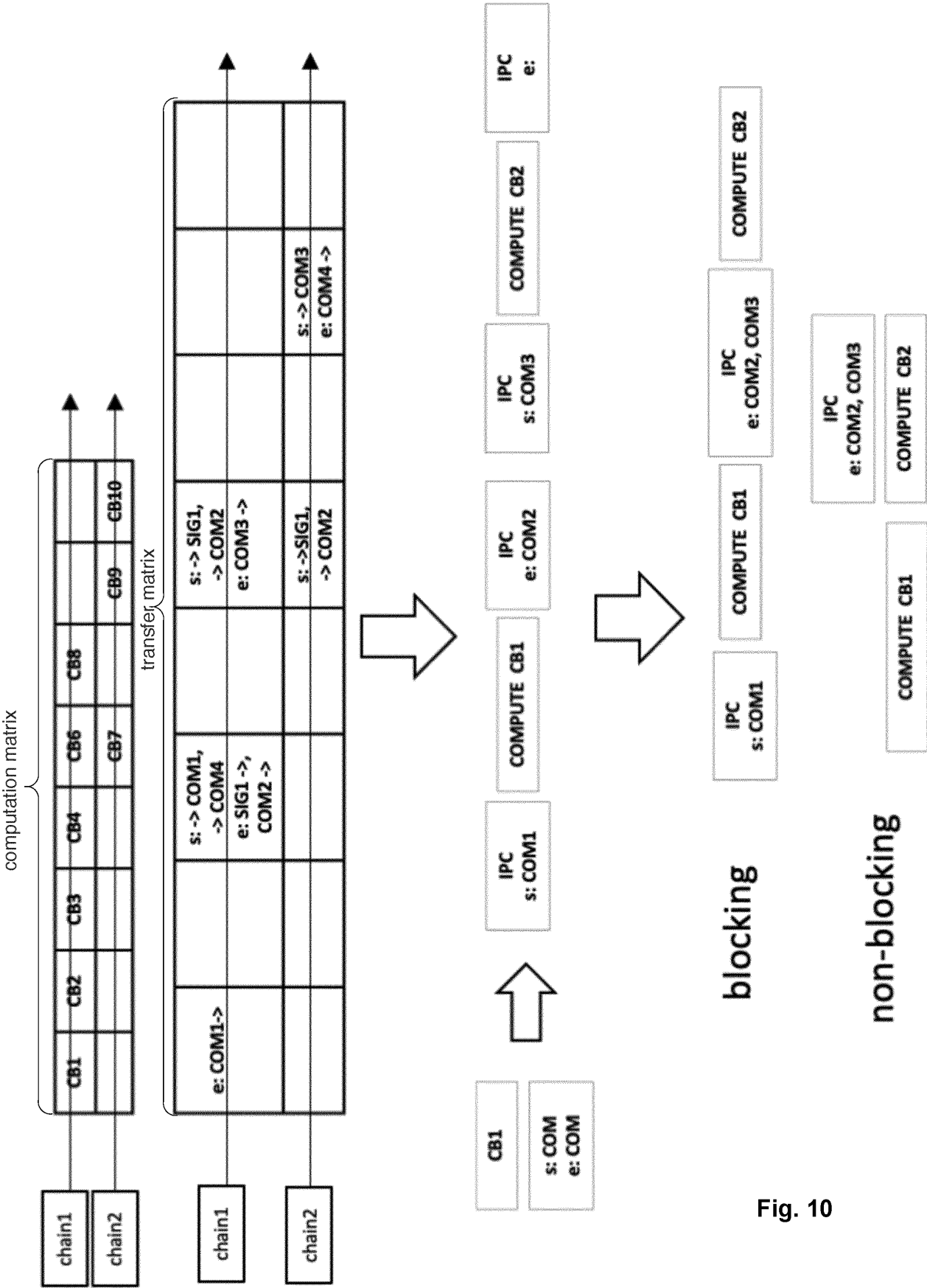


Fig. 10

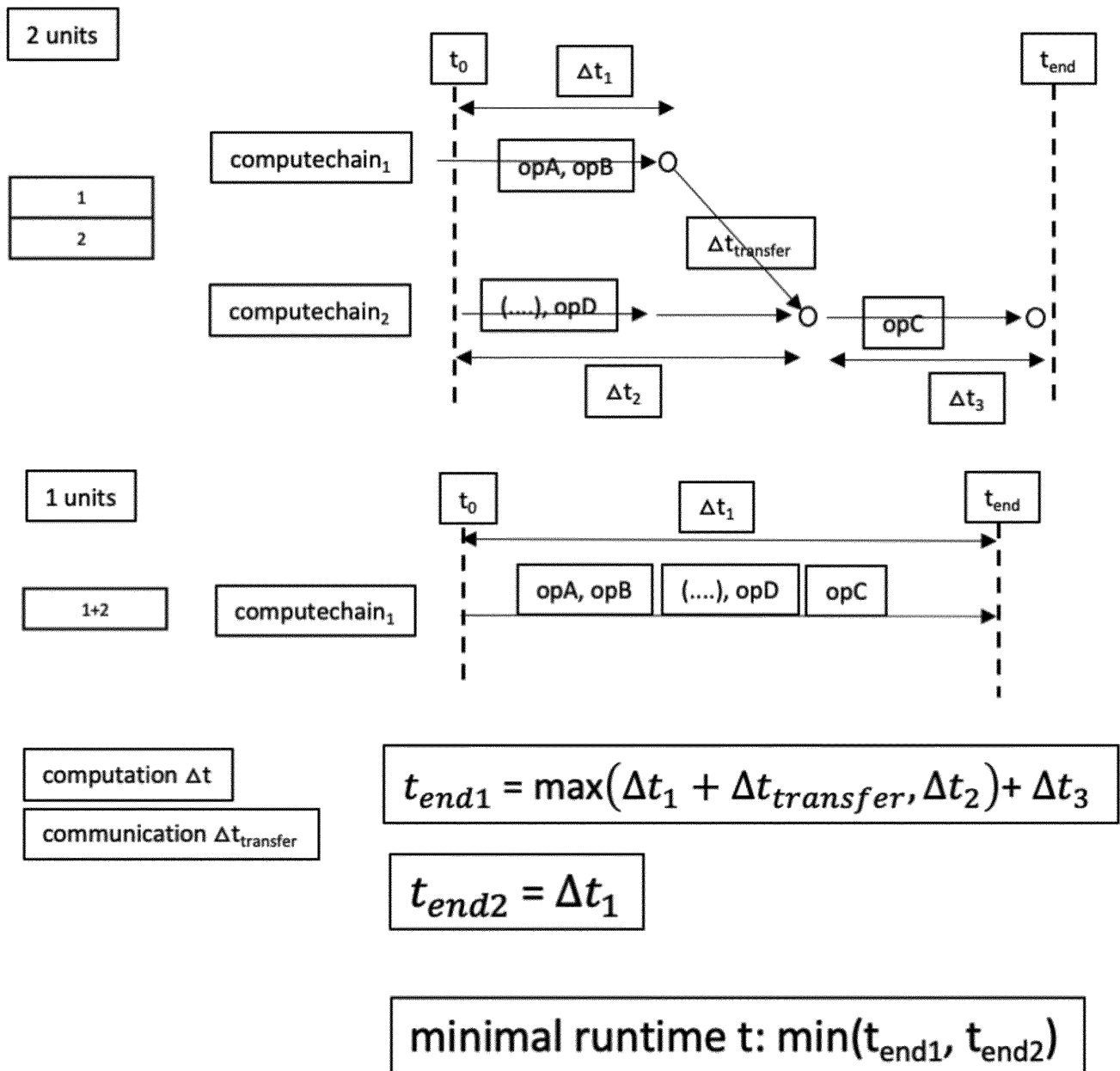


Fig. 11

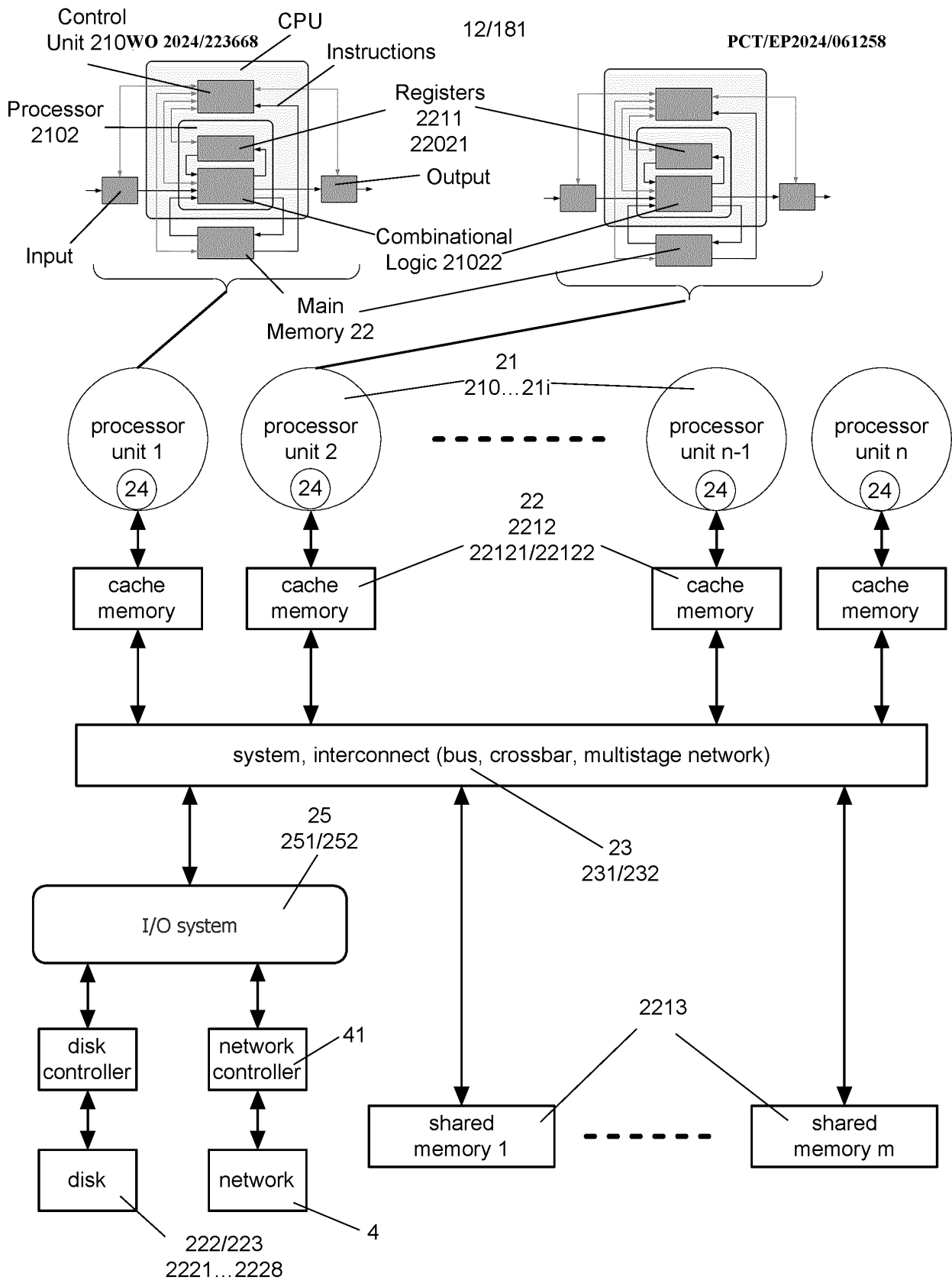


Fig. 12

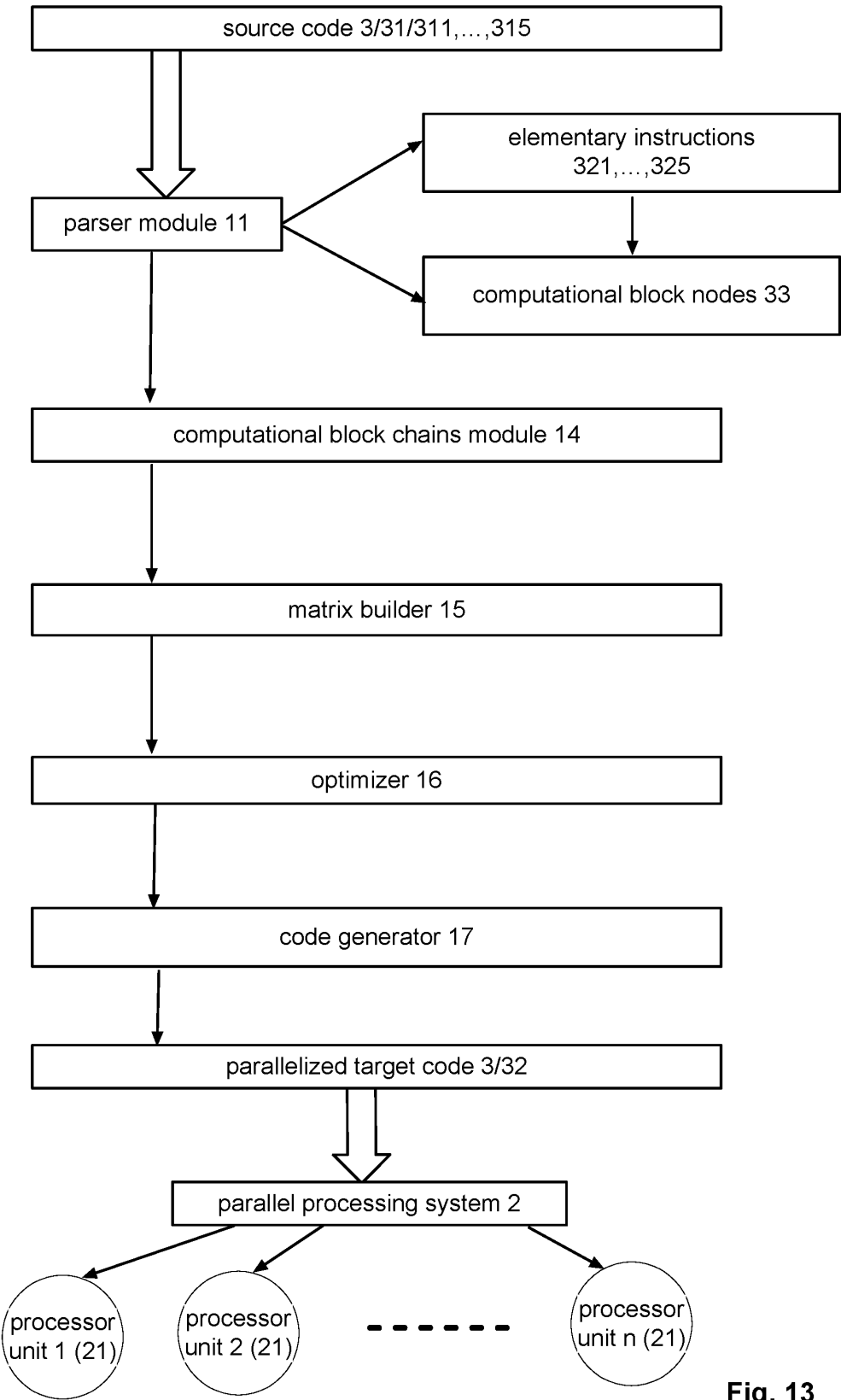


Fig. 13

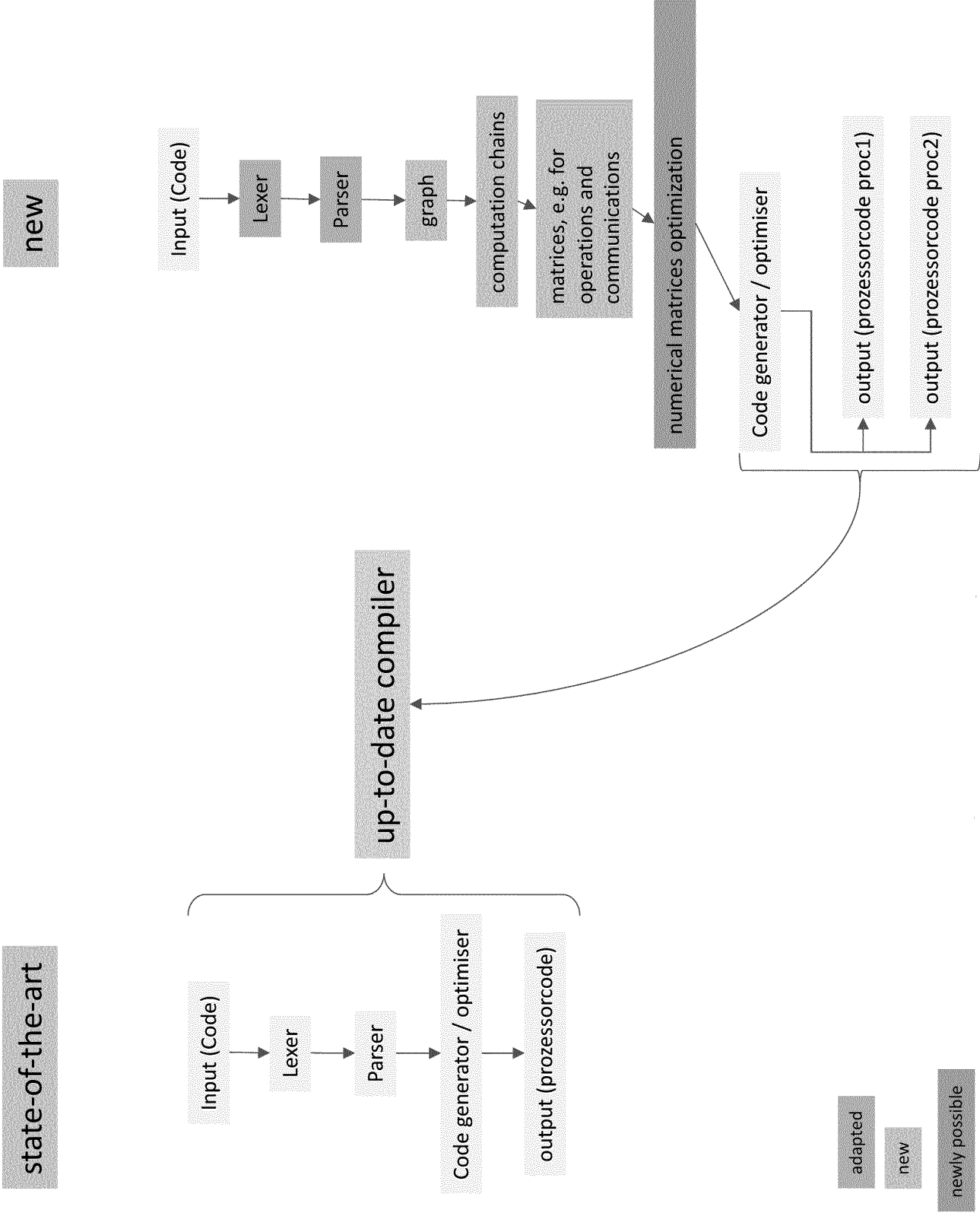


Fig. 14

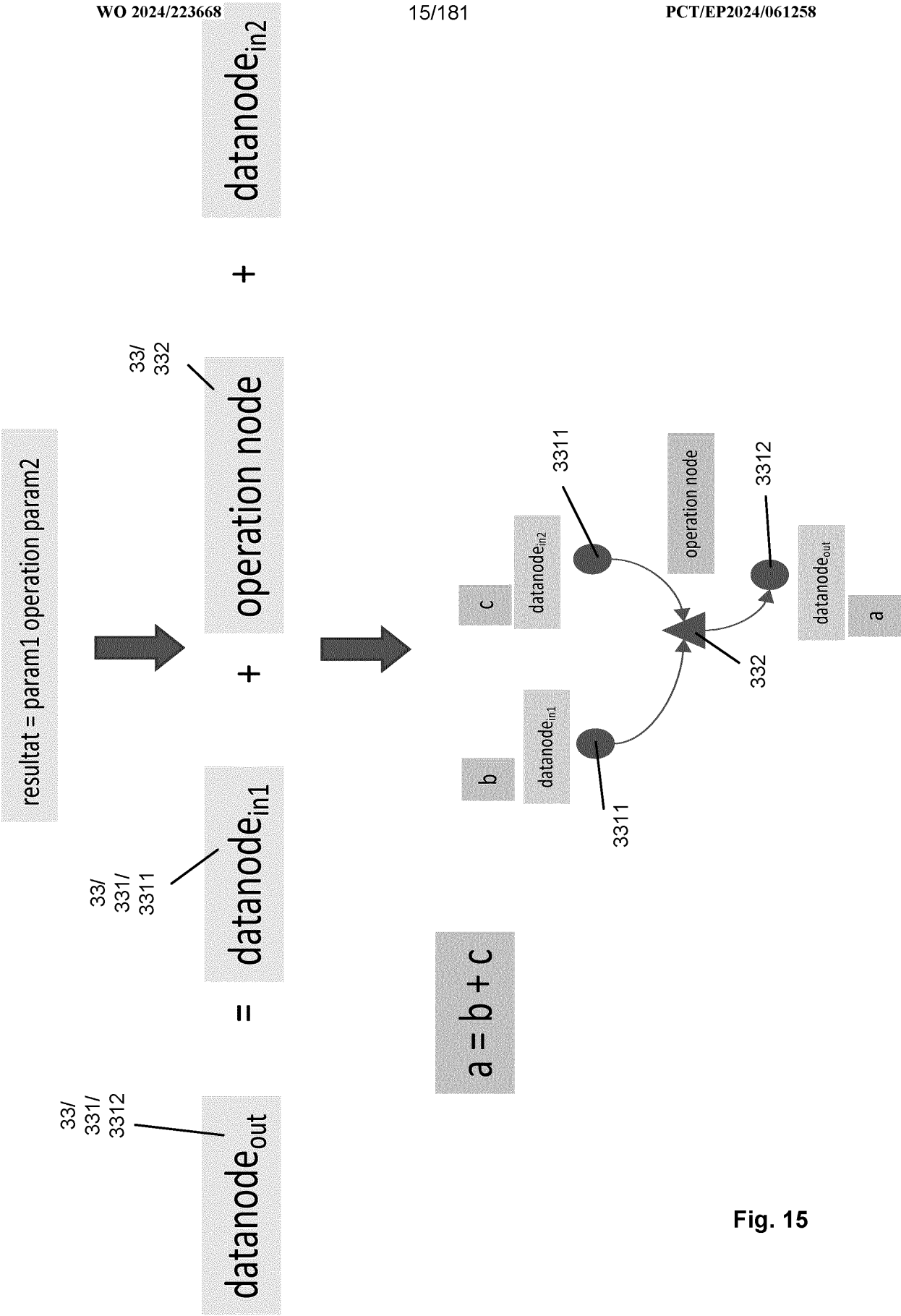


Fig. 15

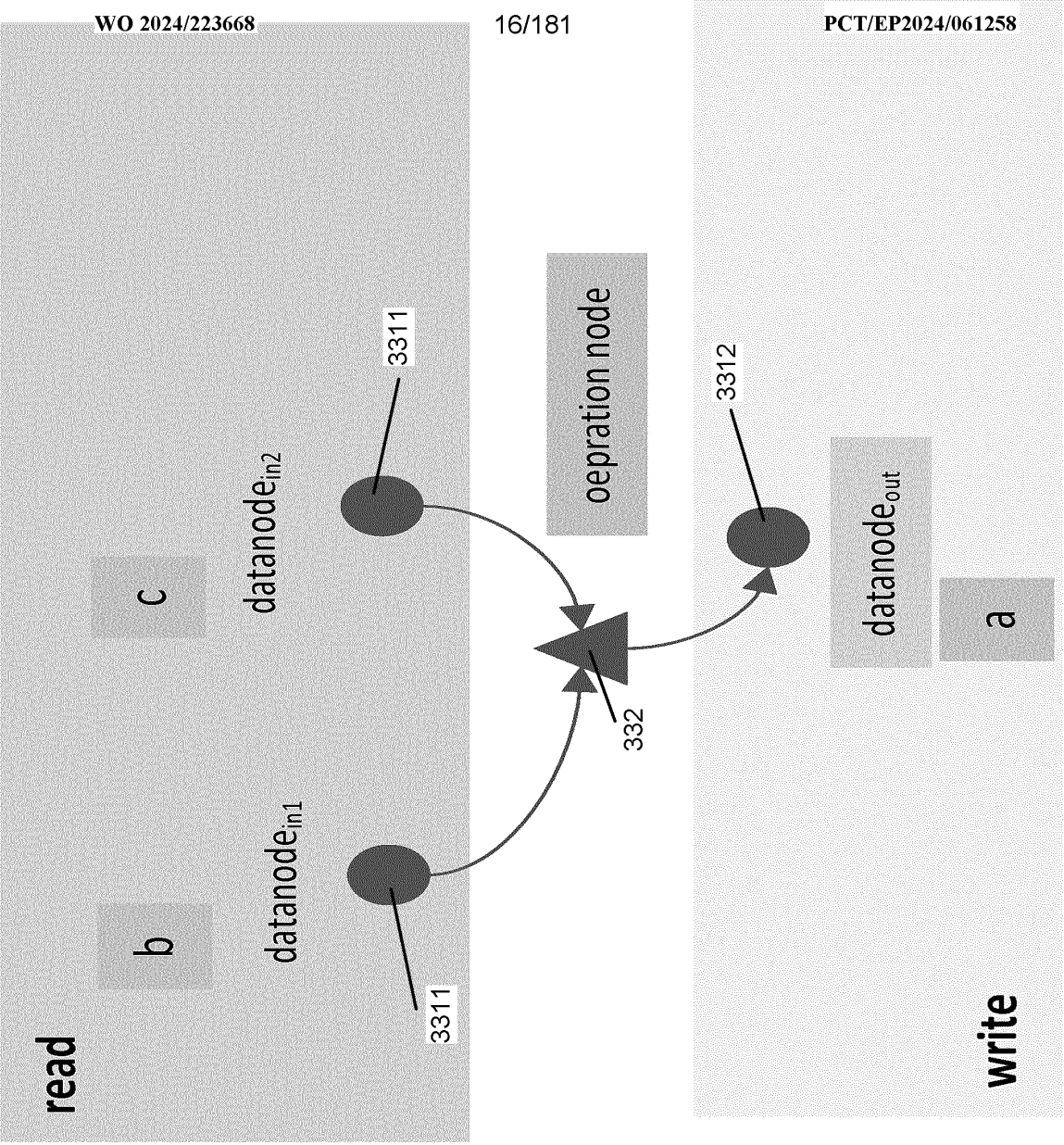


Fig. 16

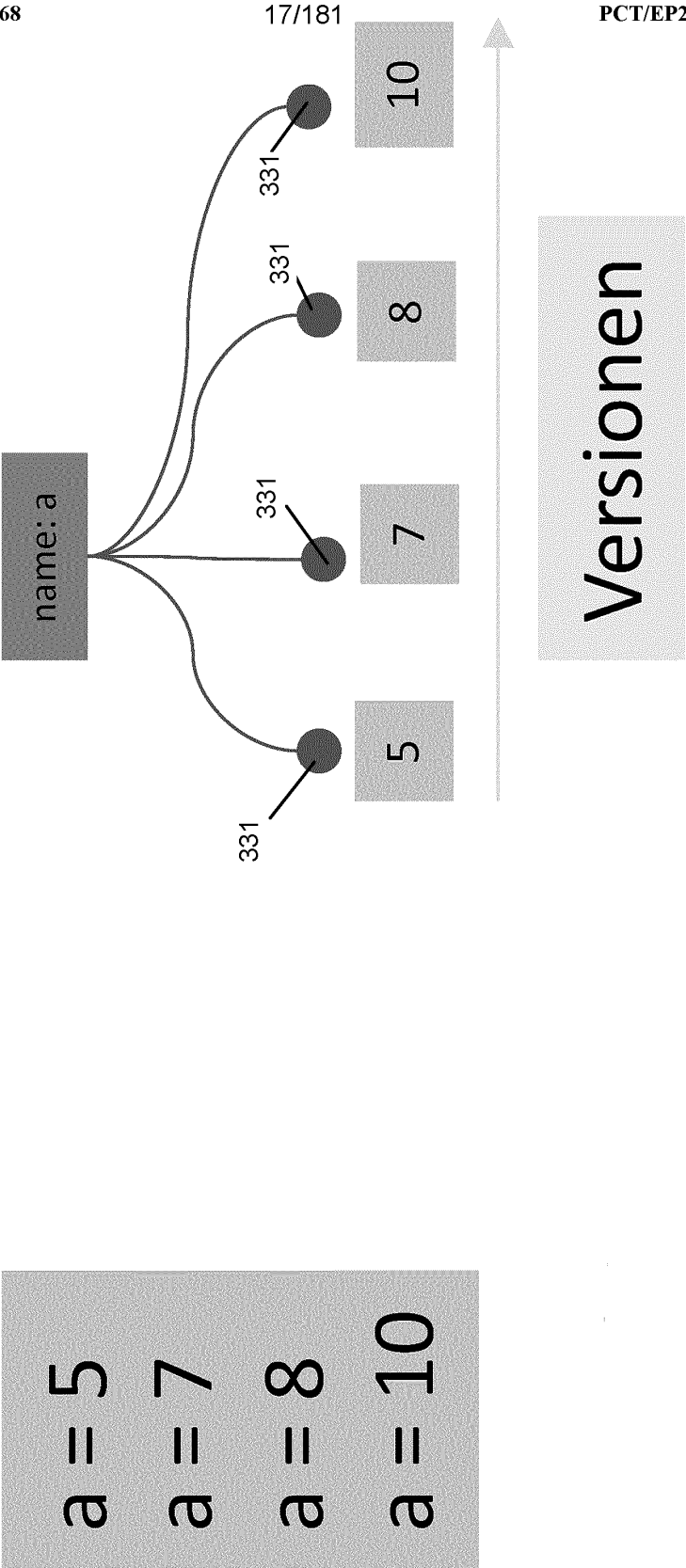


Fig. 17

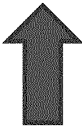
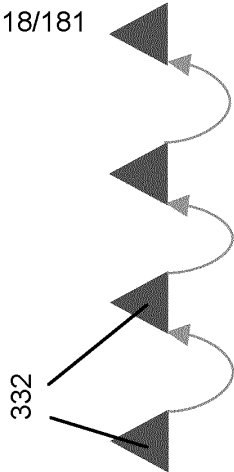
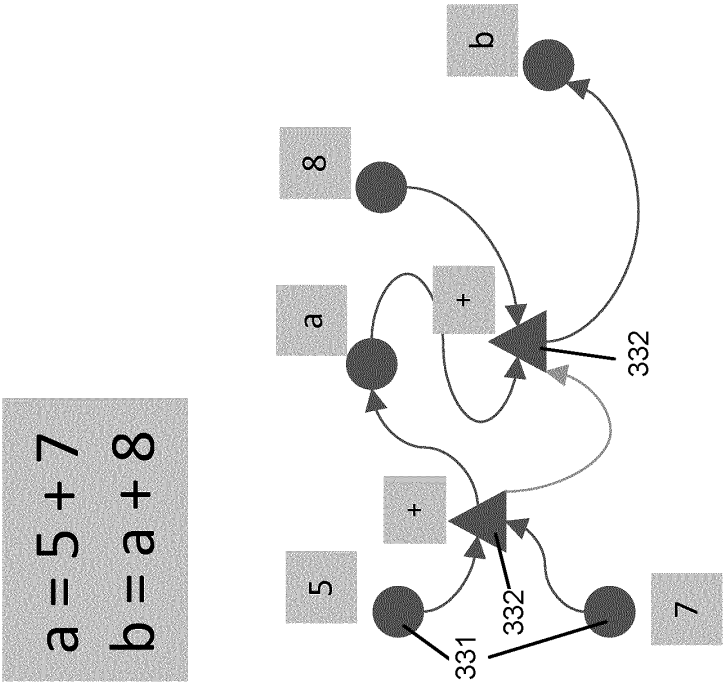


Fig. 18

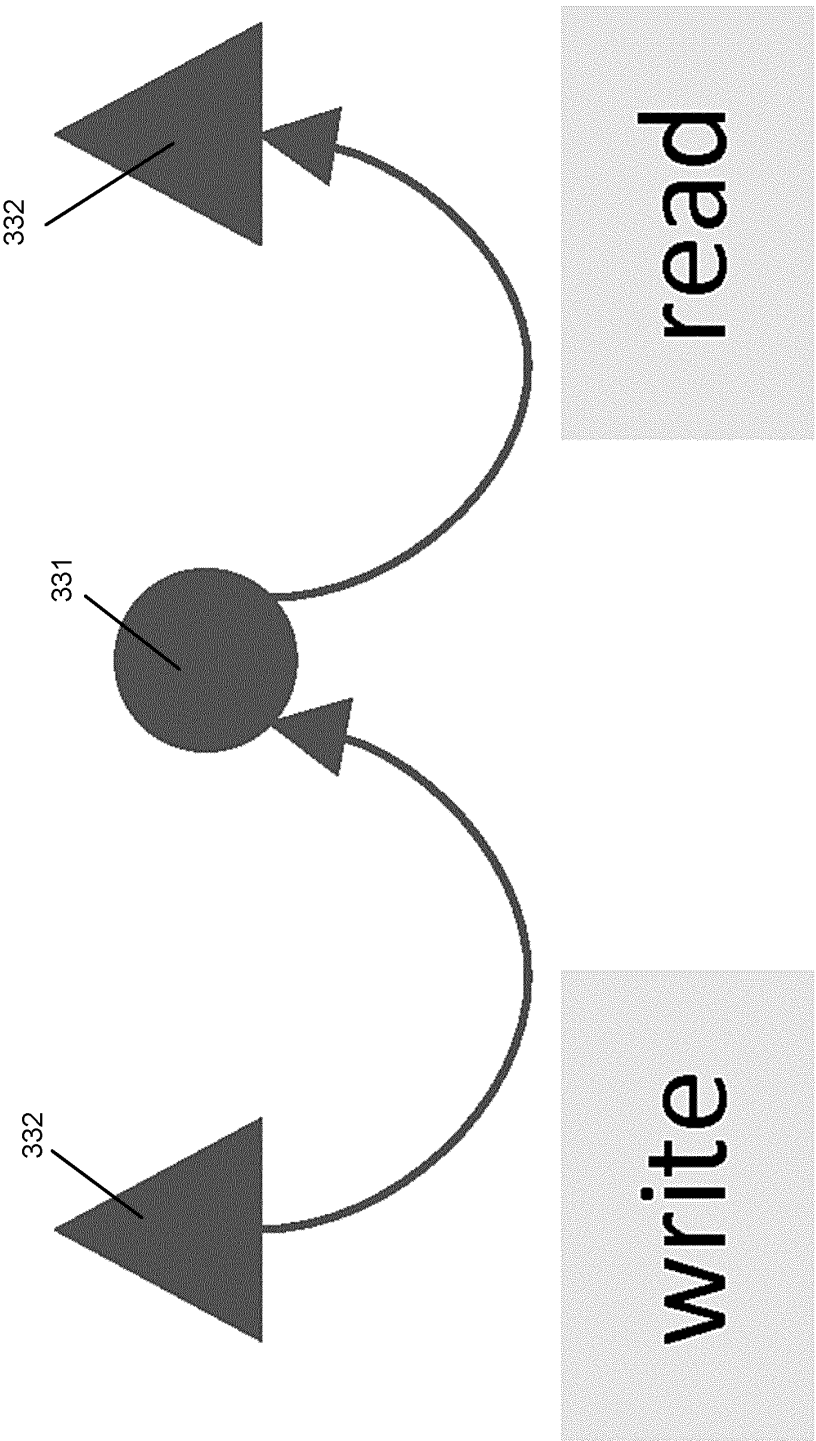
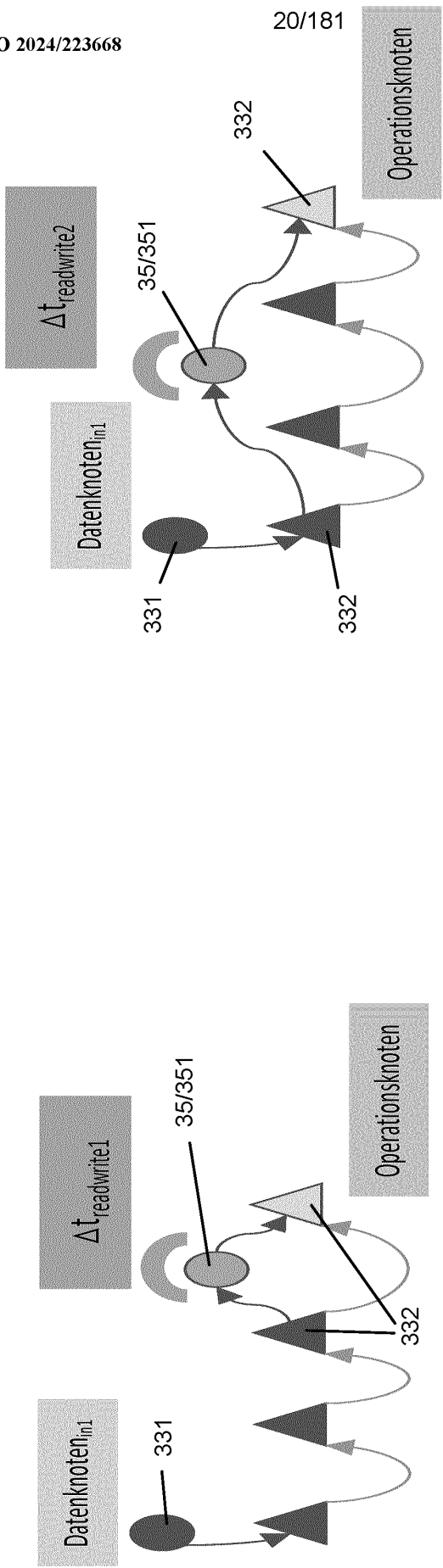


Fig. 19



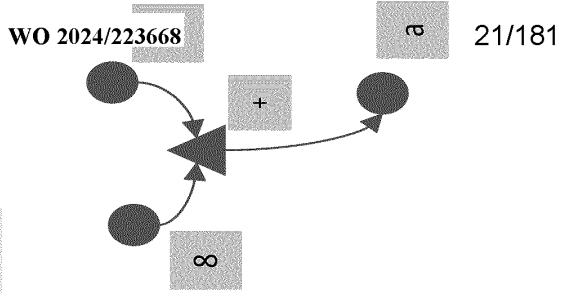
$\Delta t_{\text{readwrite}}$ as a indicator what can be done with this data node

- $\Delta t_{\text{readwrite}}$ sehr klein: same cache_line or nearby
- $\Delta t_{\text{readwrite}}$ „mittel gross“: on same #cpu unit, use prefetch
- $\Delta t_{\text{readwrite}}$ „genug gross“: possible to transfer over network

Fig. 20

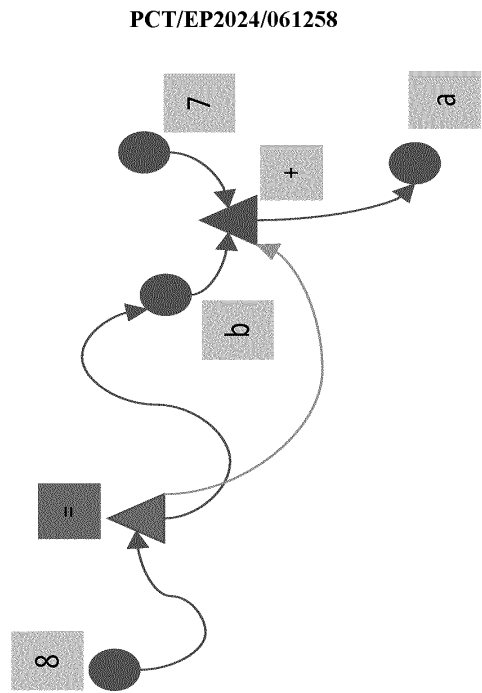
0-dependency

$$a = 8 + 7$$



1-dependency

$$b = 8$$
$$a = b + 7$$



2-dependency

$$c = 5 + 7$$
$$b = 7$$
$$a = b + c$$

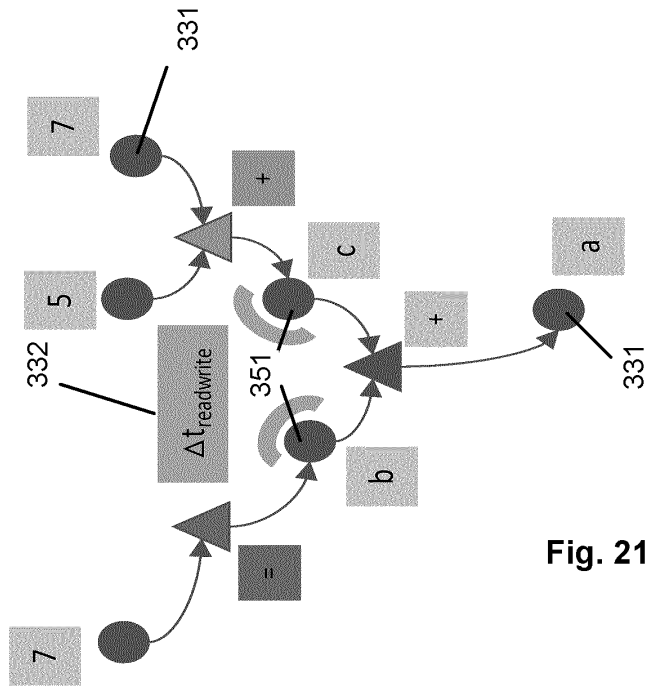


Fig. 21

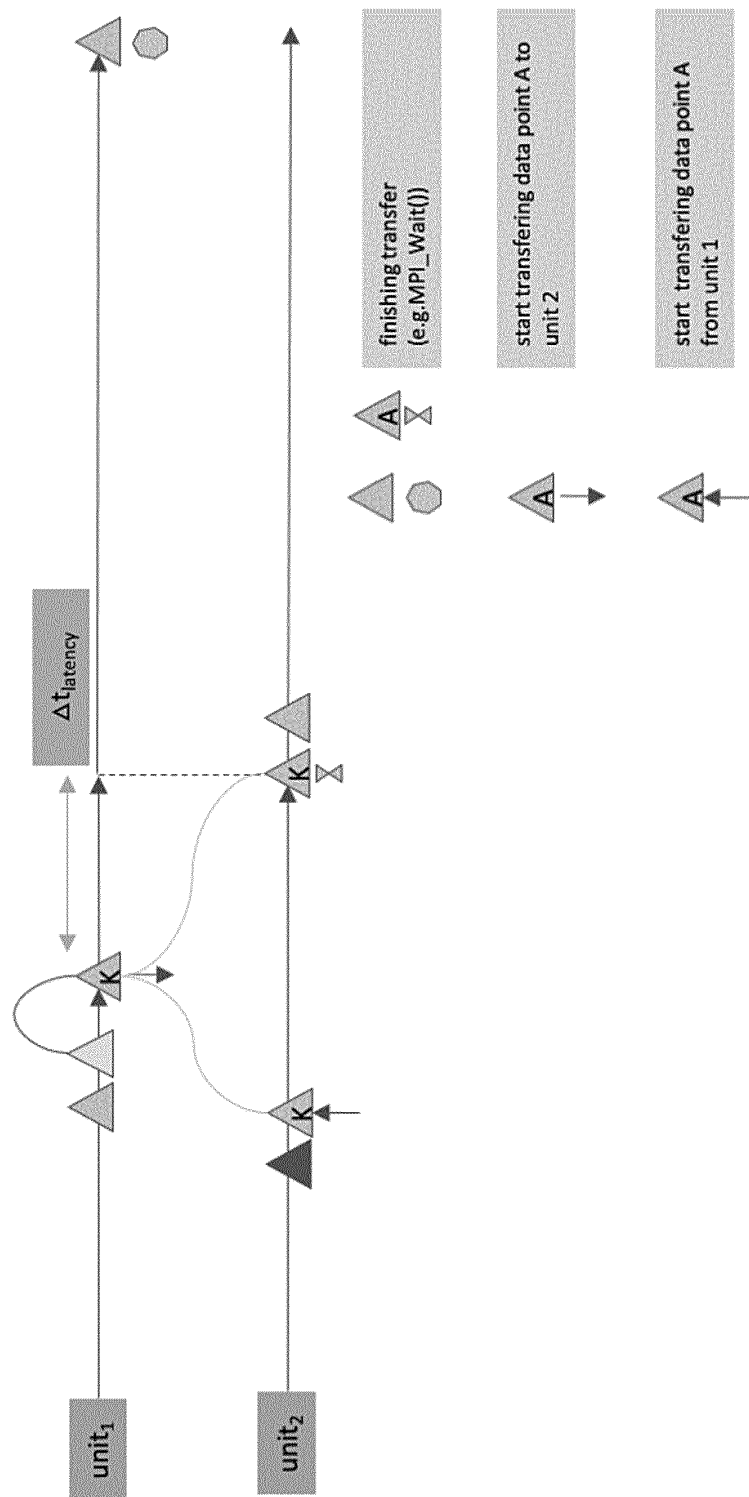
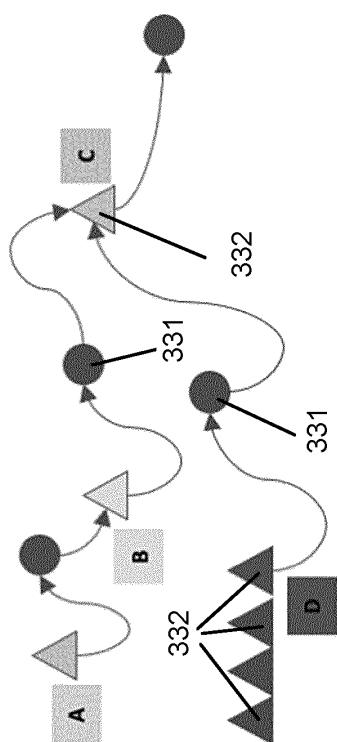


Fig. 22

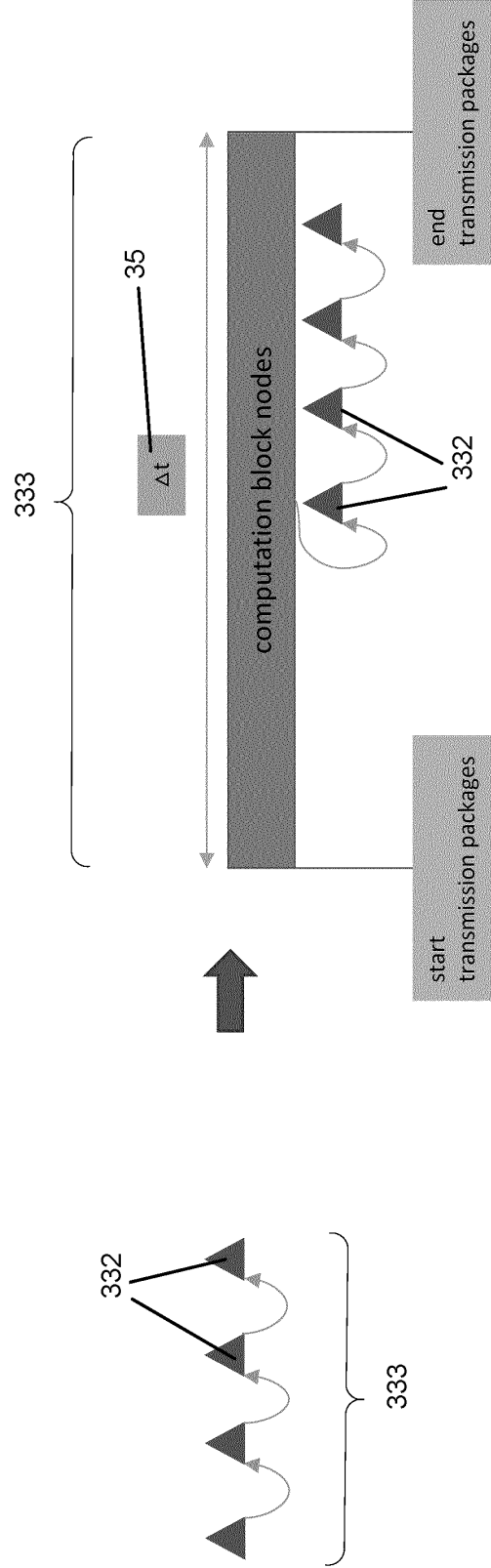


Fig. 23

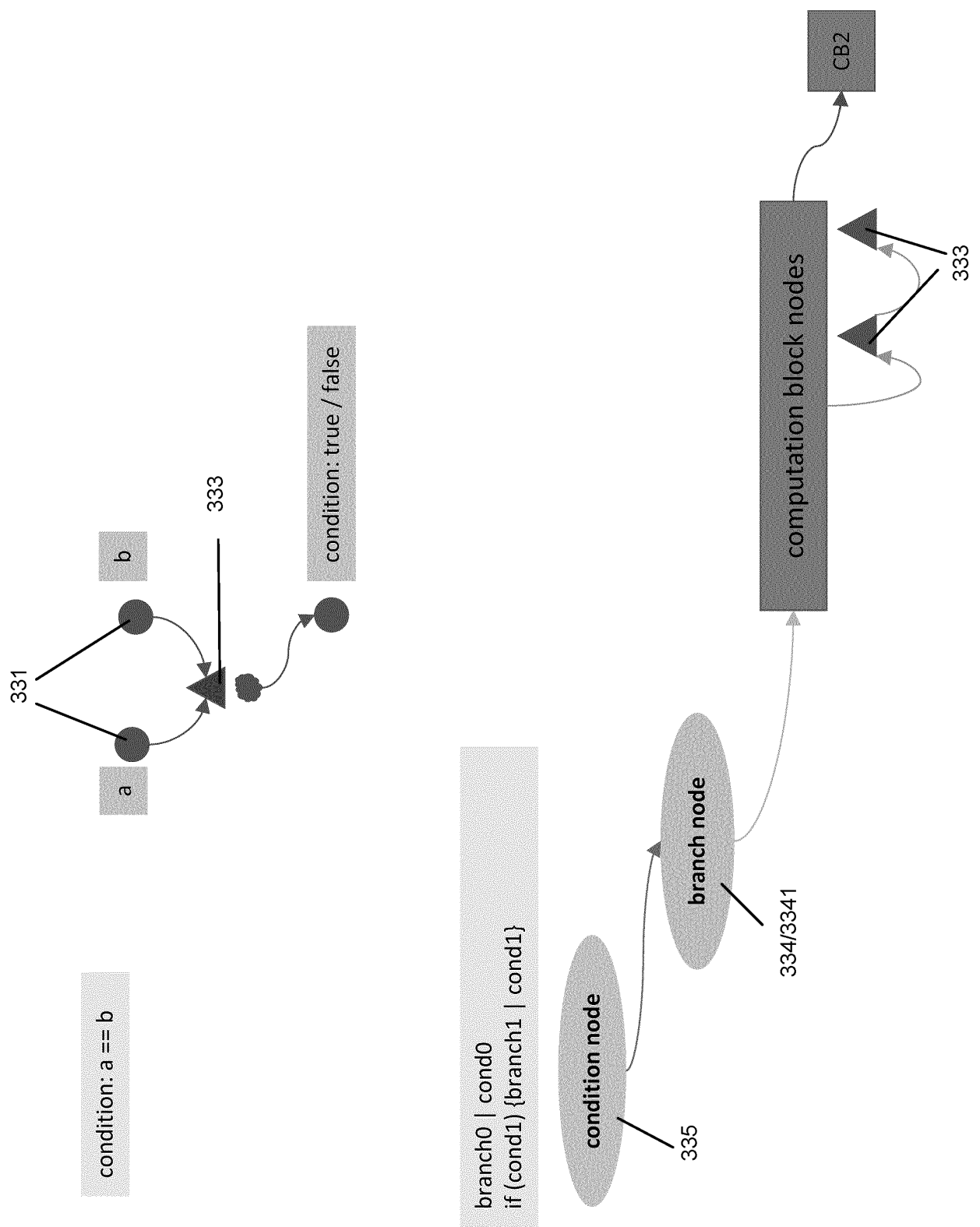


Fig. 24

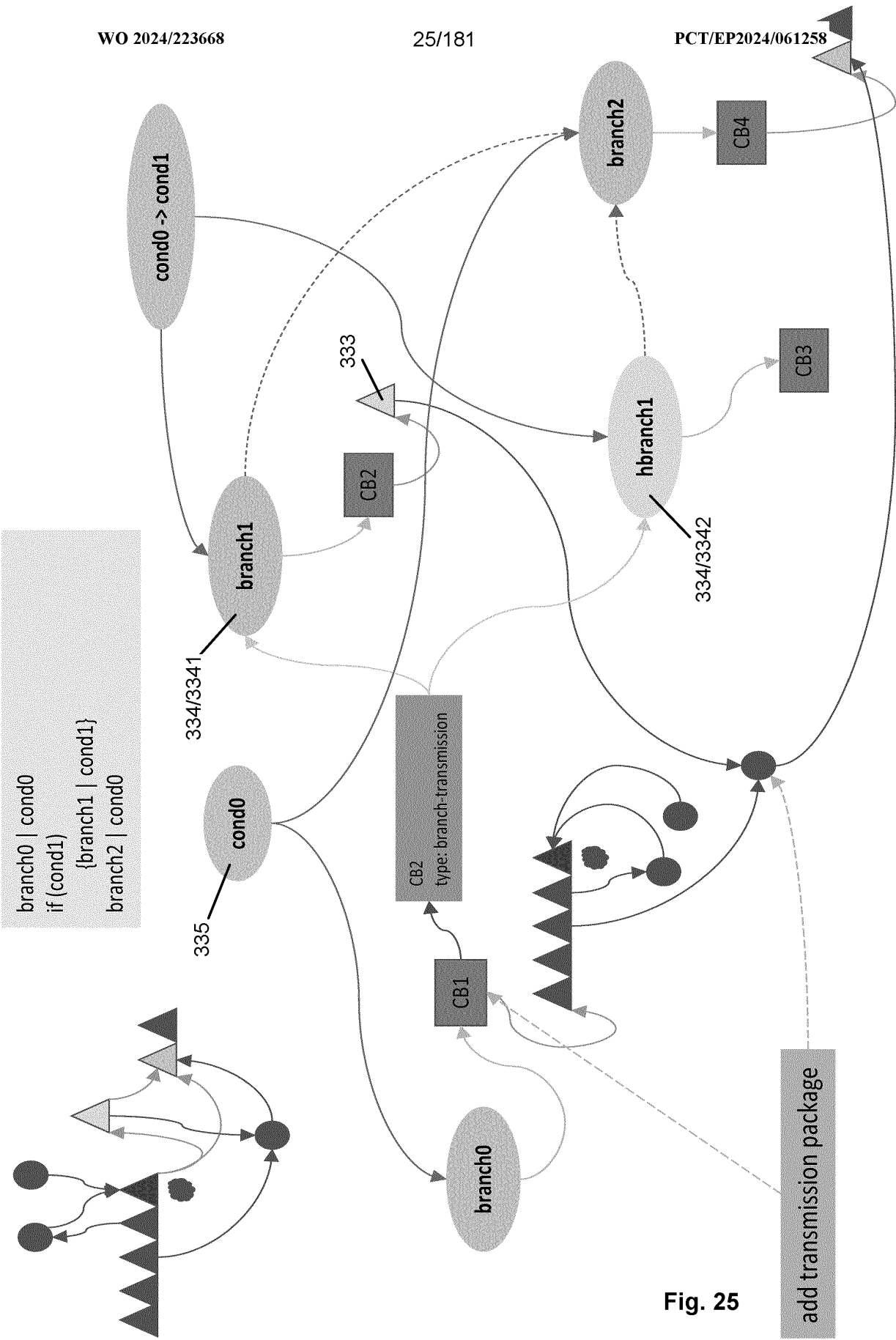


Fig. 25

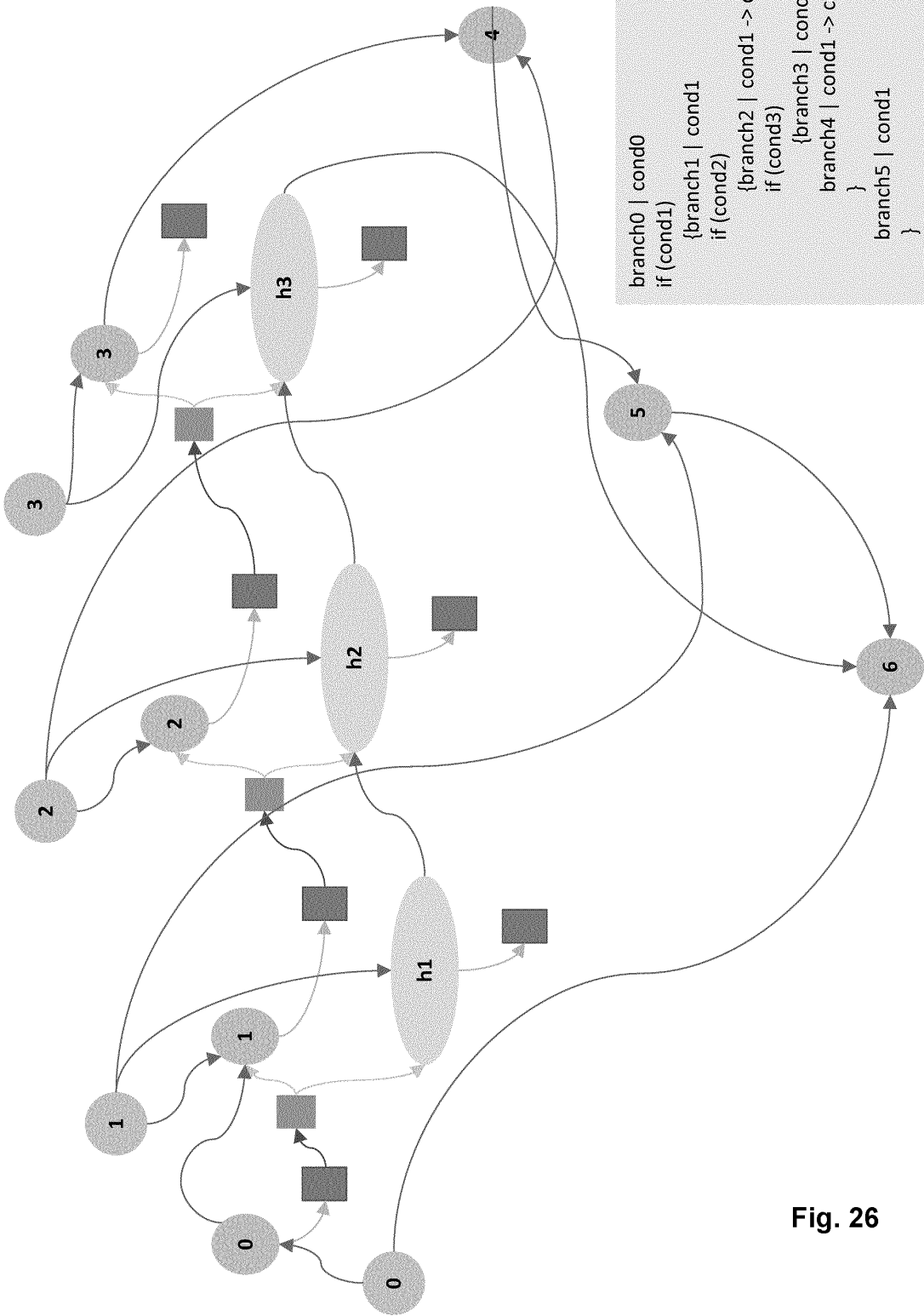


Fig. 26

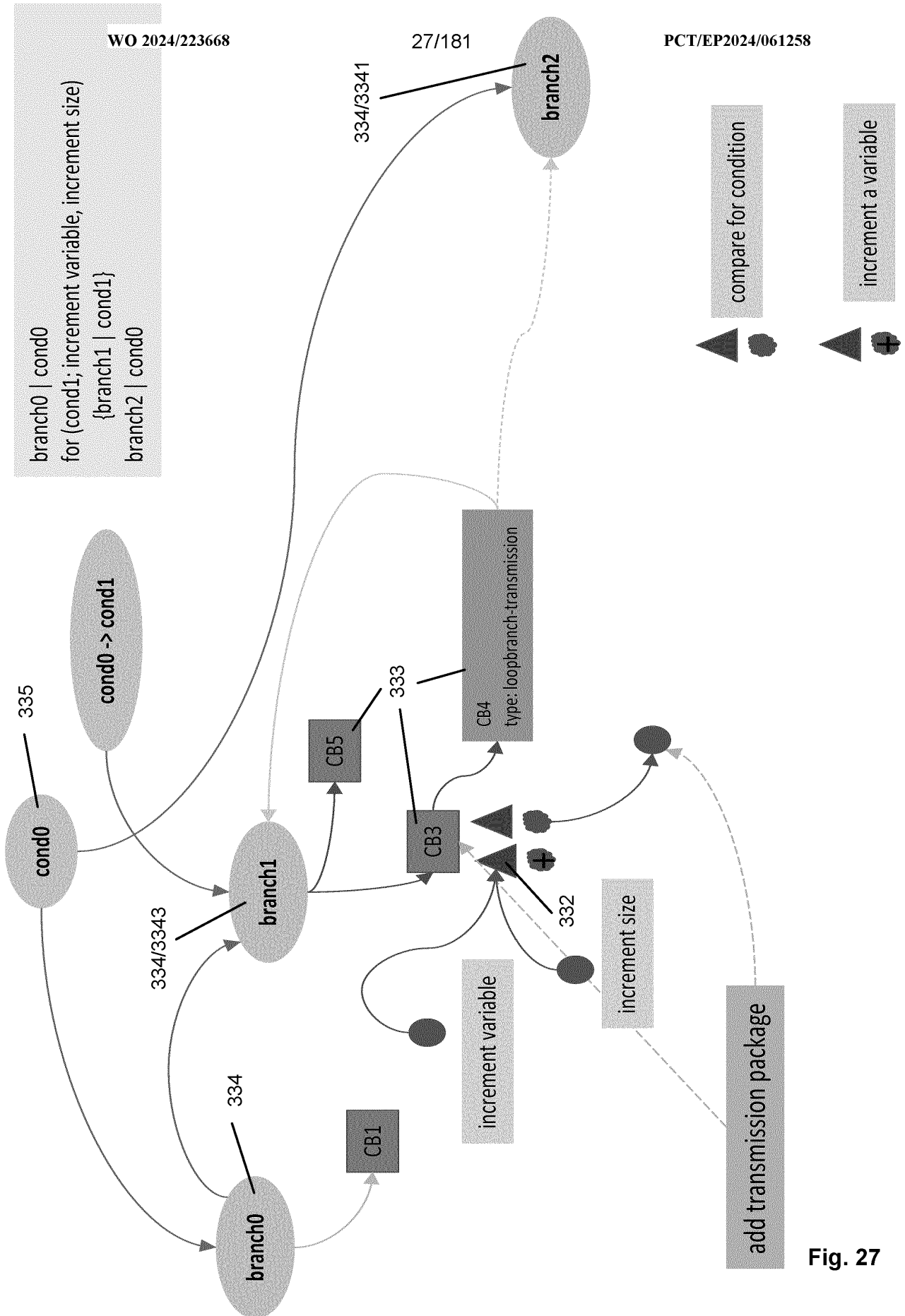


Fig. 27

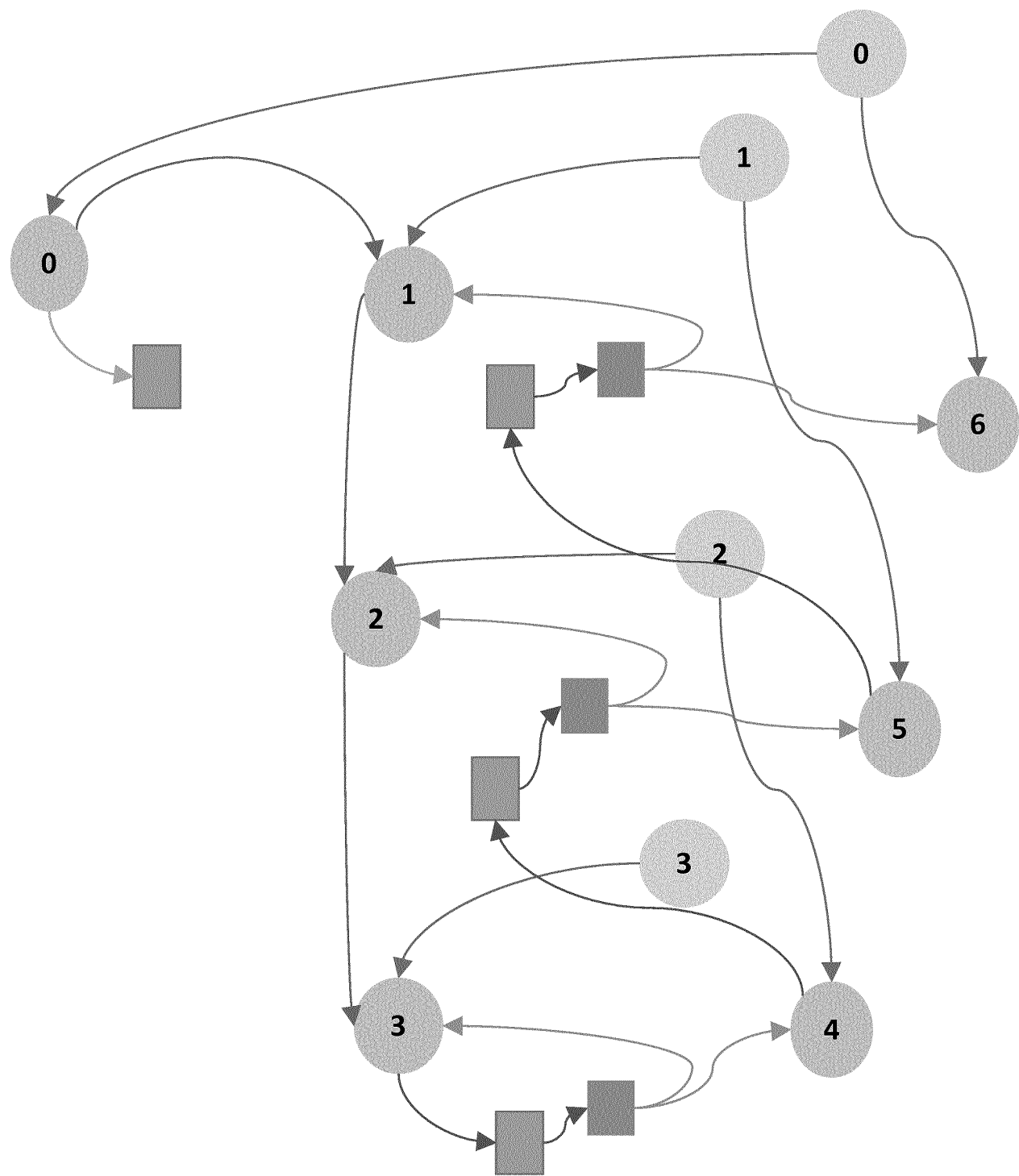


Fig. 28

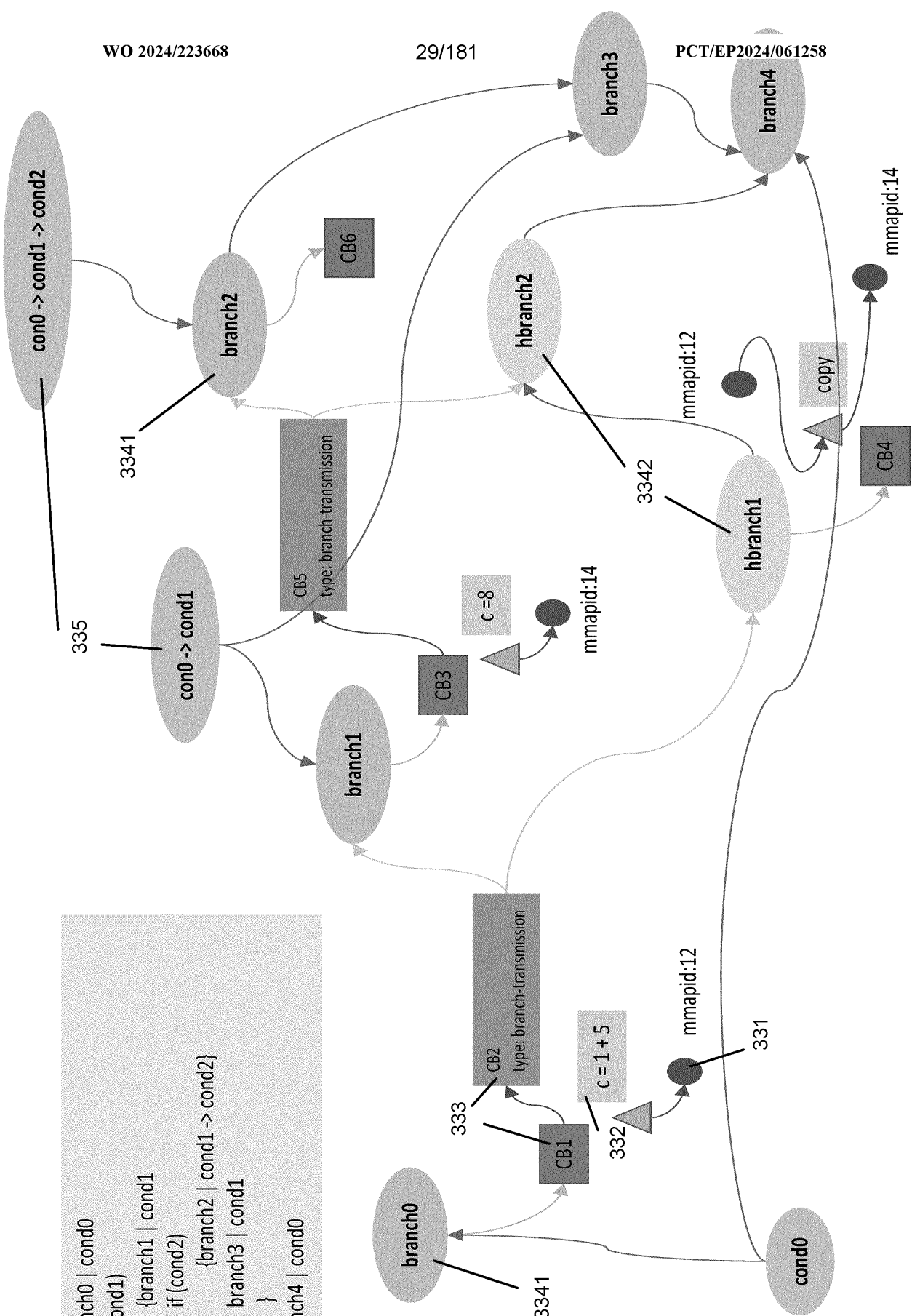


Fig. 29

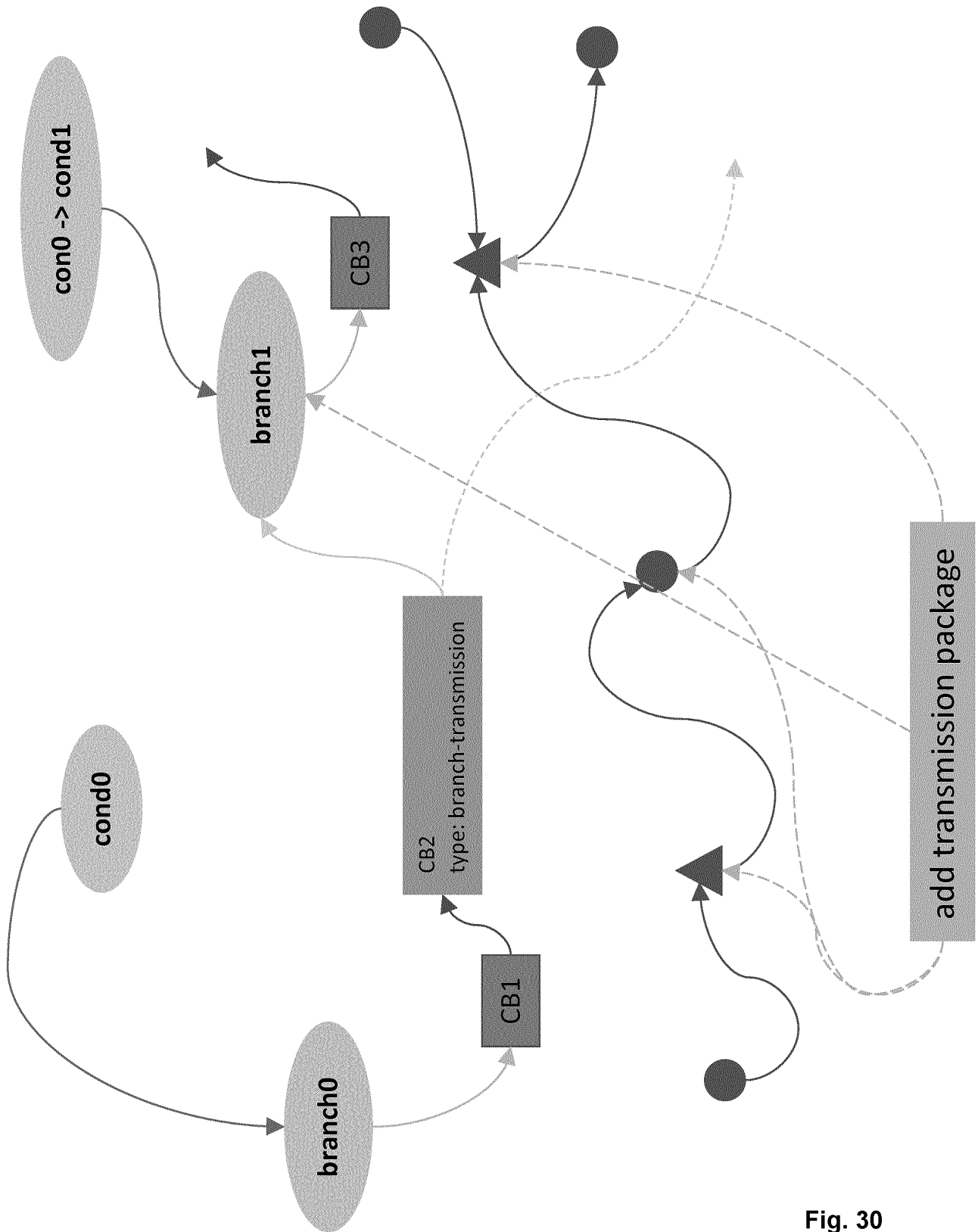


Fig. 30

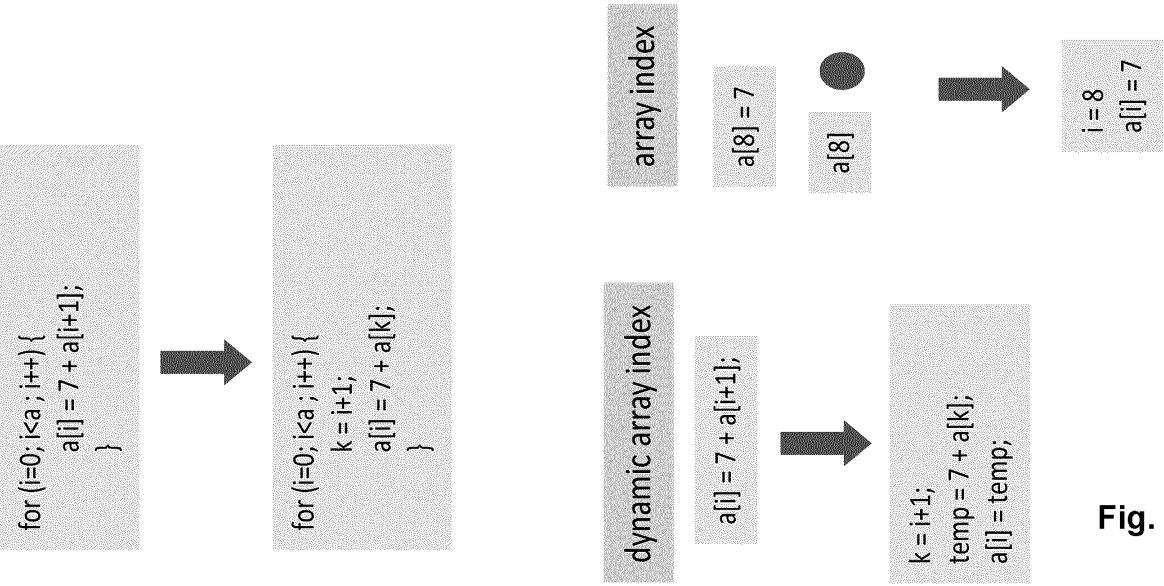
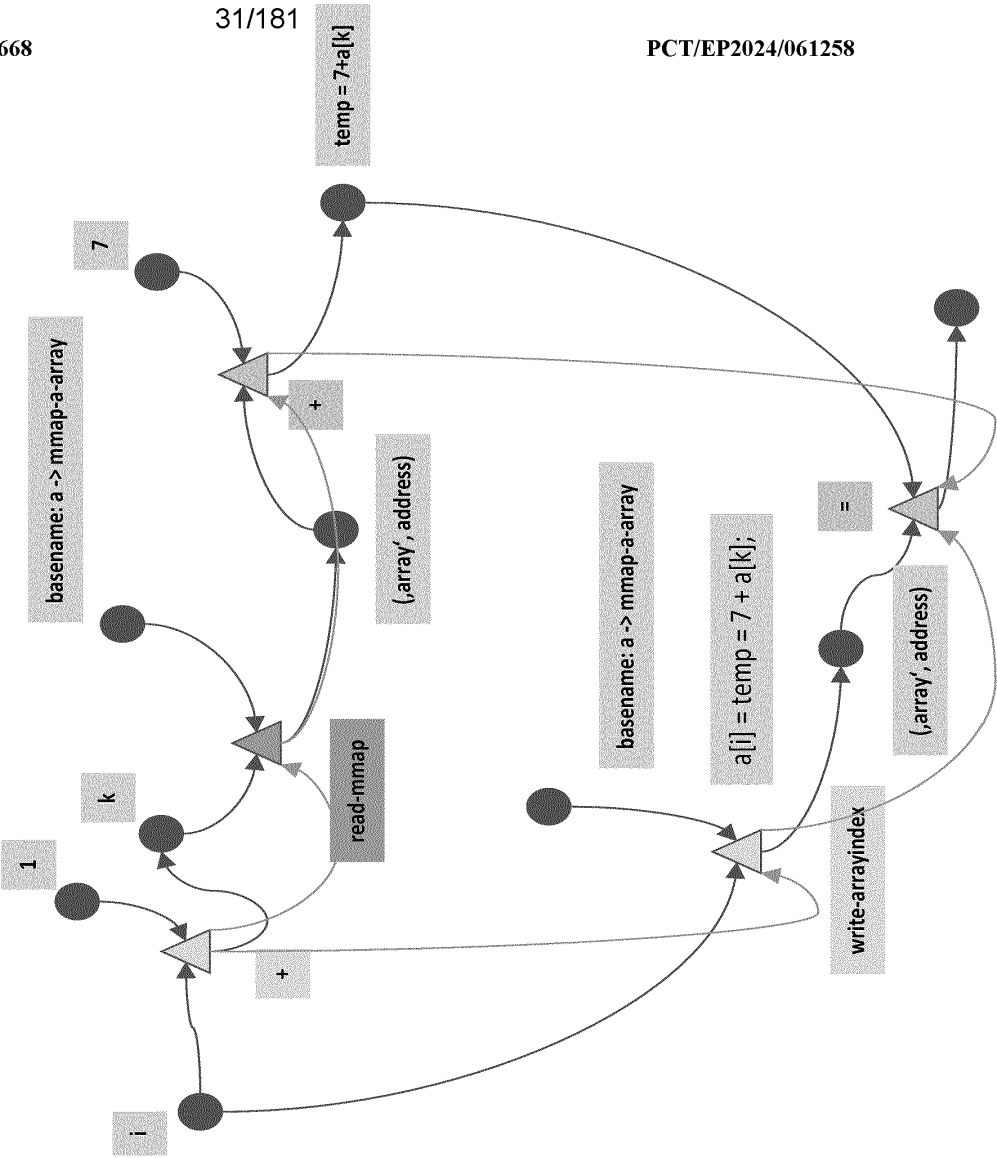


Fig. 31



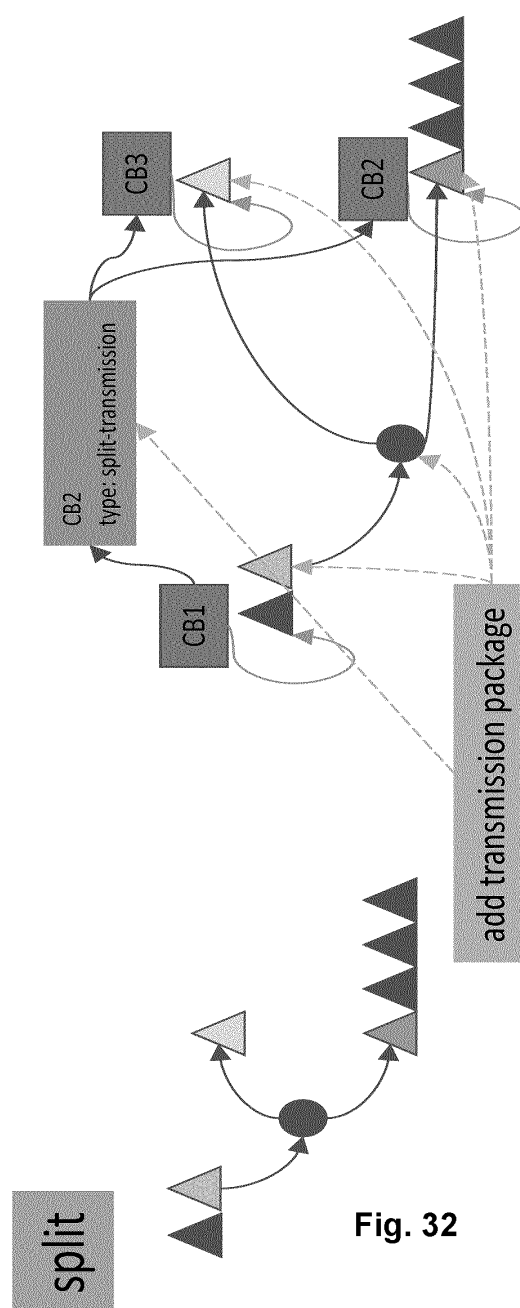


Fig. 32

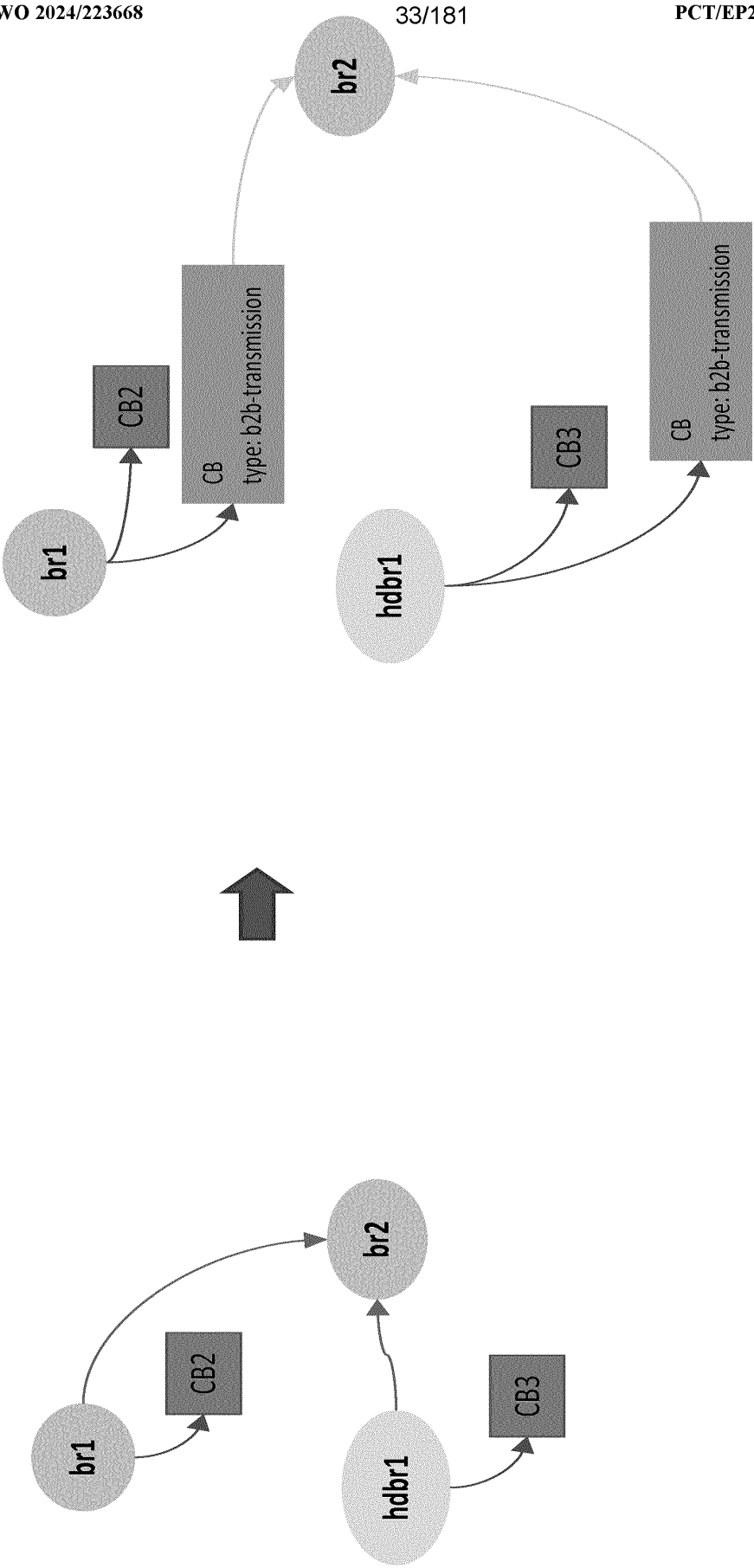


Fig. 33

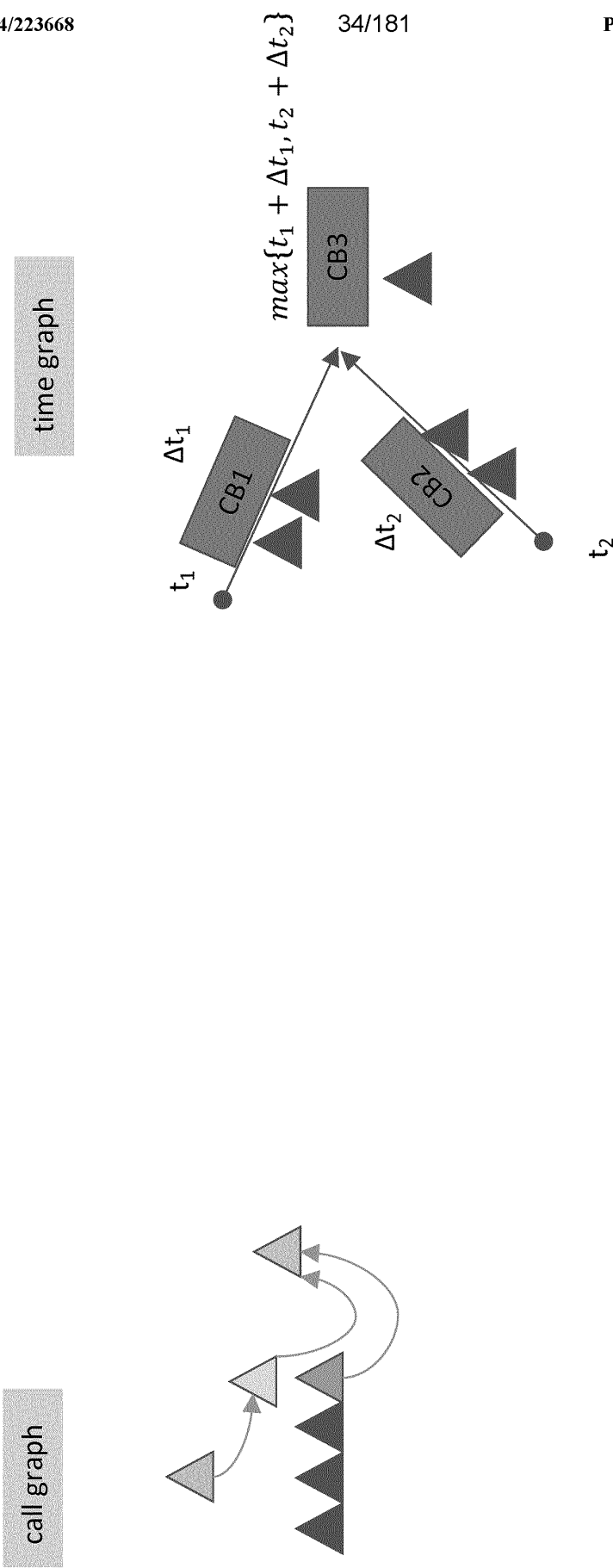
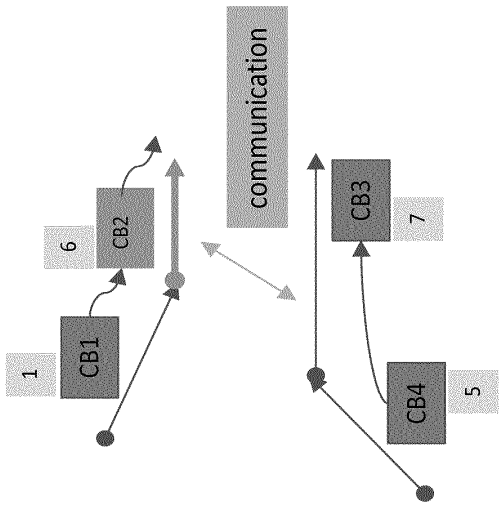
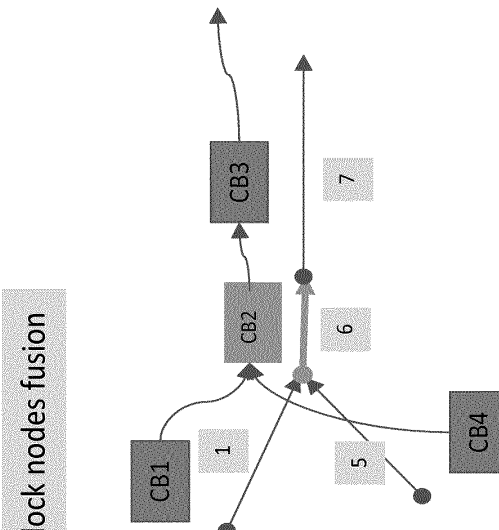
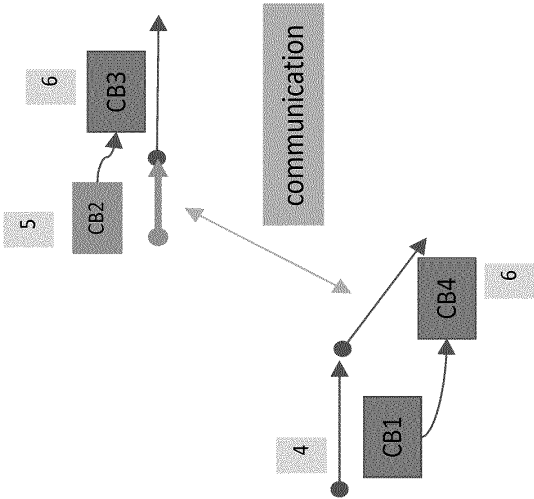


Fig. 34

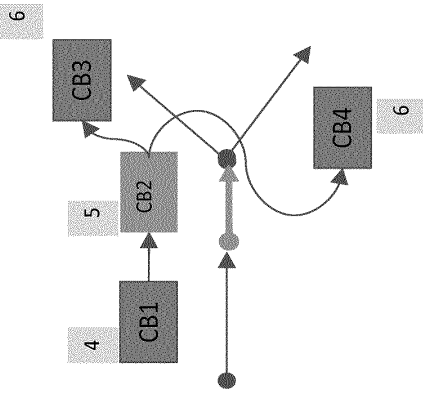


example block numbers

1



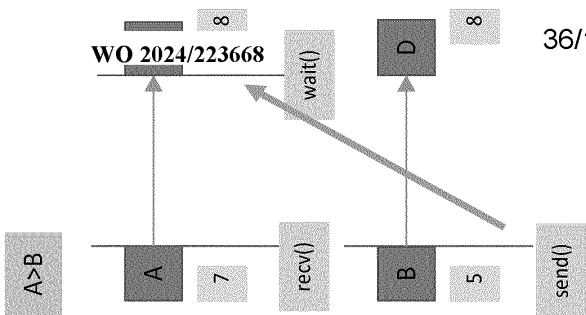
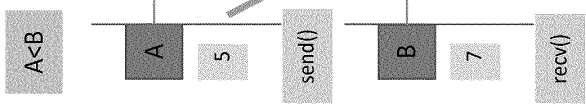
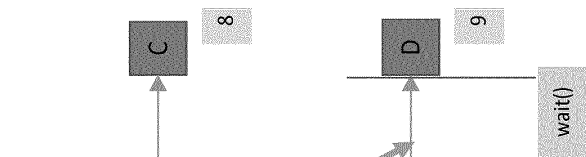
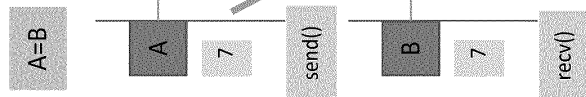
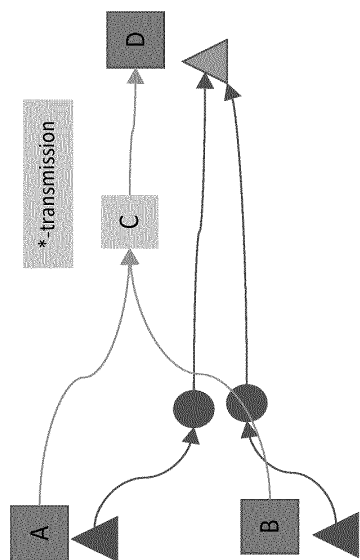
computation block nodes fusion



computation block nodes split

Fig. 35

computation block nodes fusion



computation block nodes split

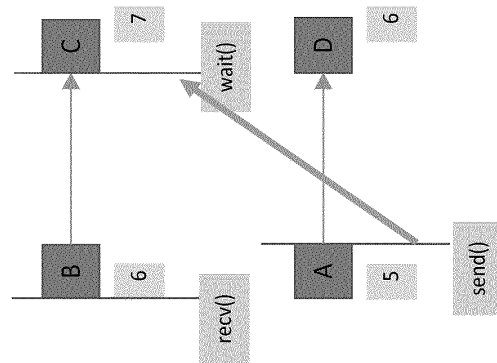
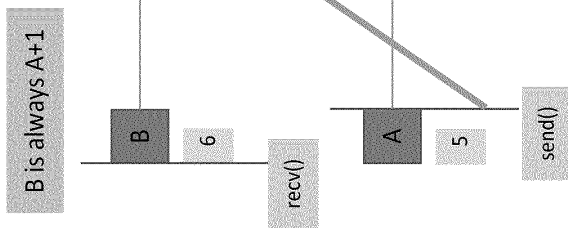
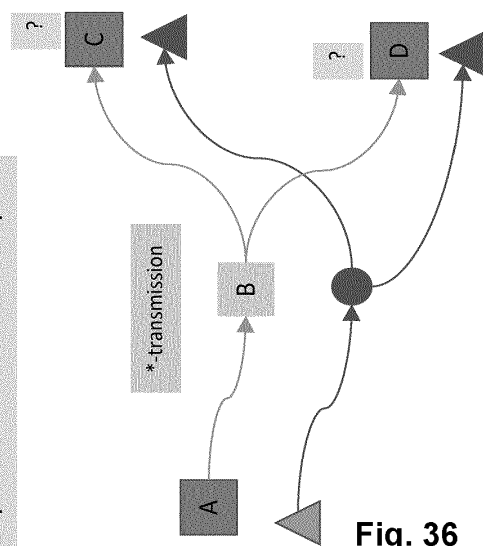
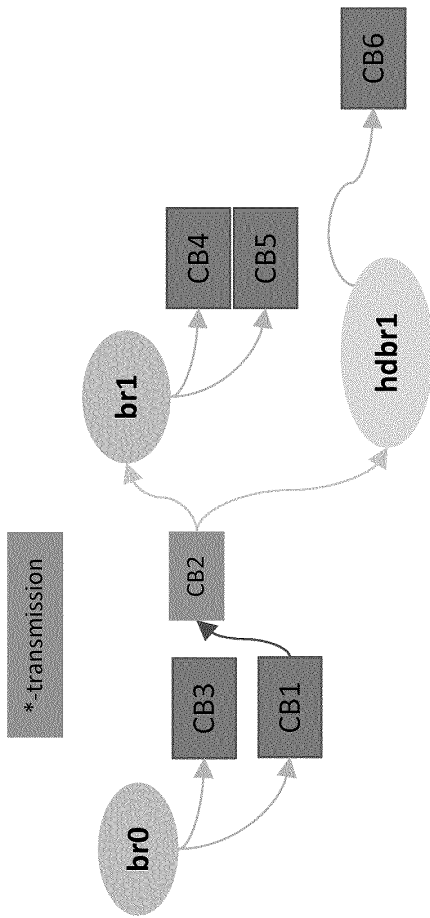


Fig. 36

branch transmissions (if- and loop-branching)



condition transmissions

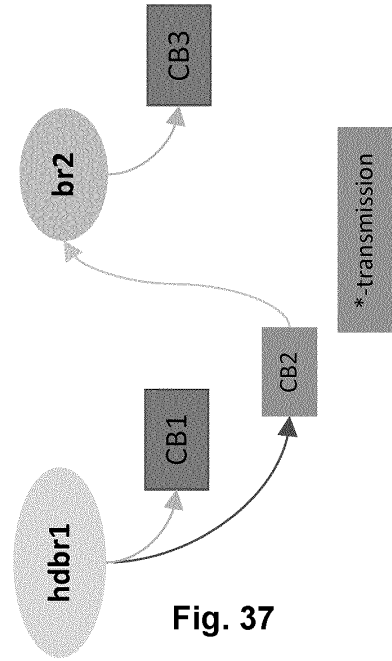
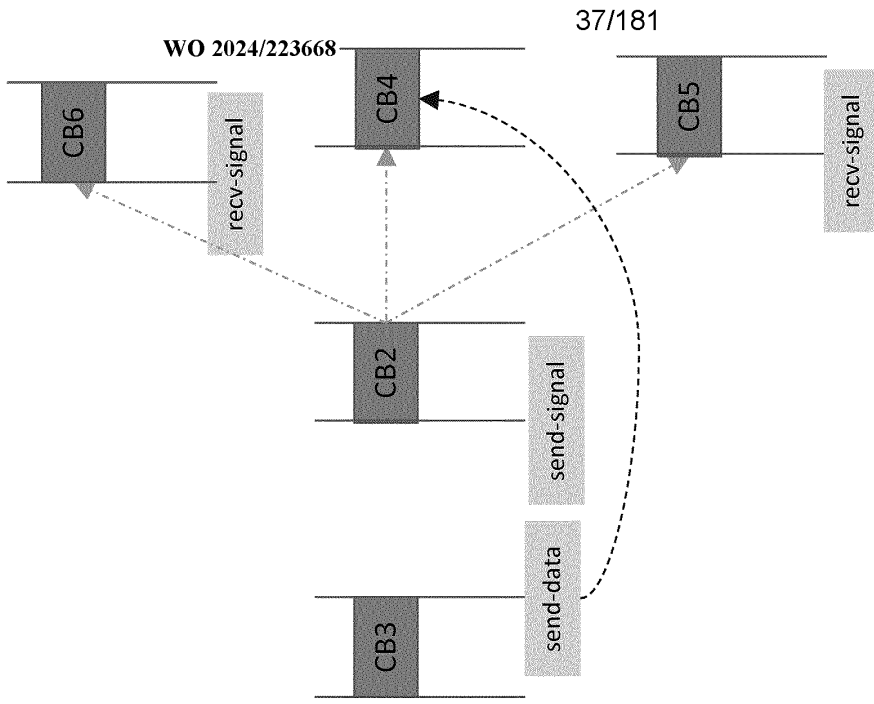
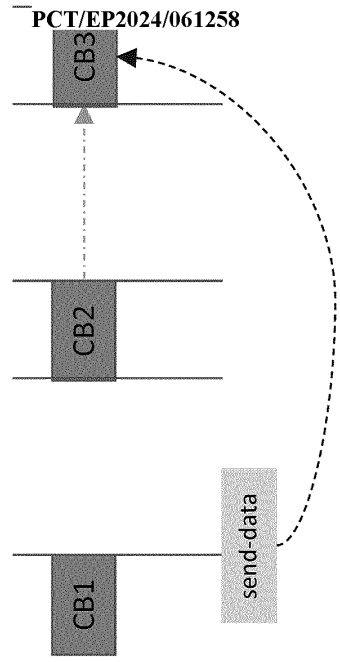


Fig. 37



PCT/EP2024/061258



37/181

WO 2024/223668

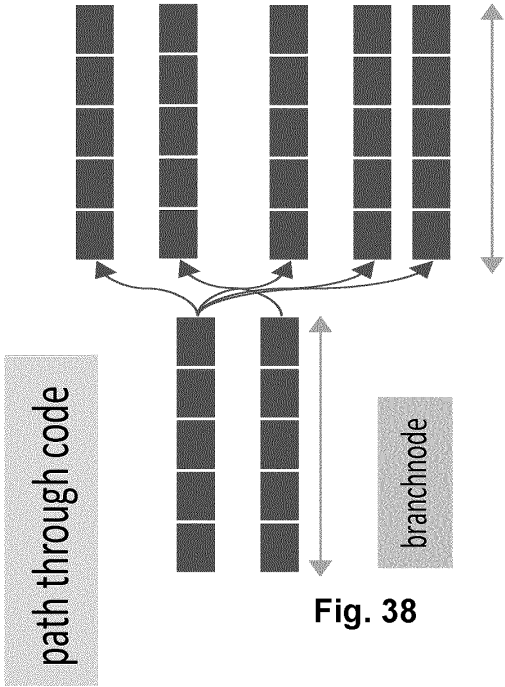
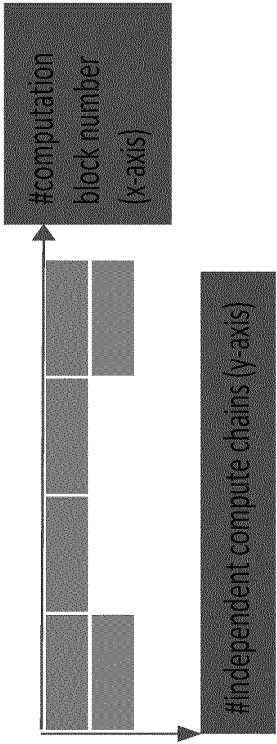
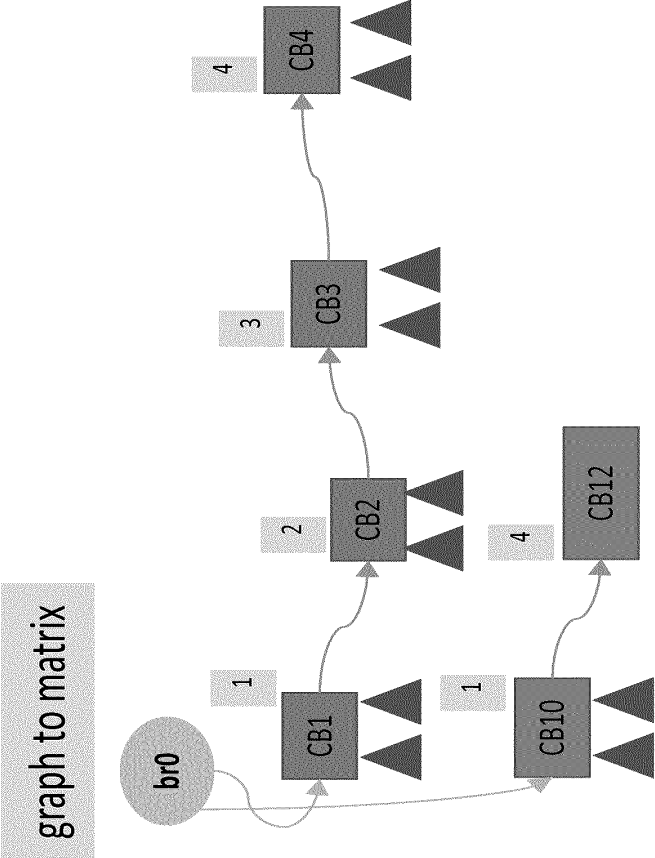


Fig. 38

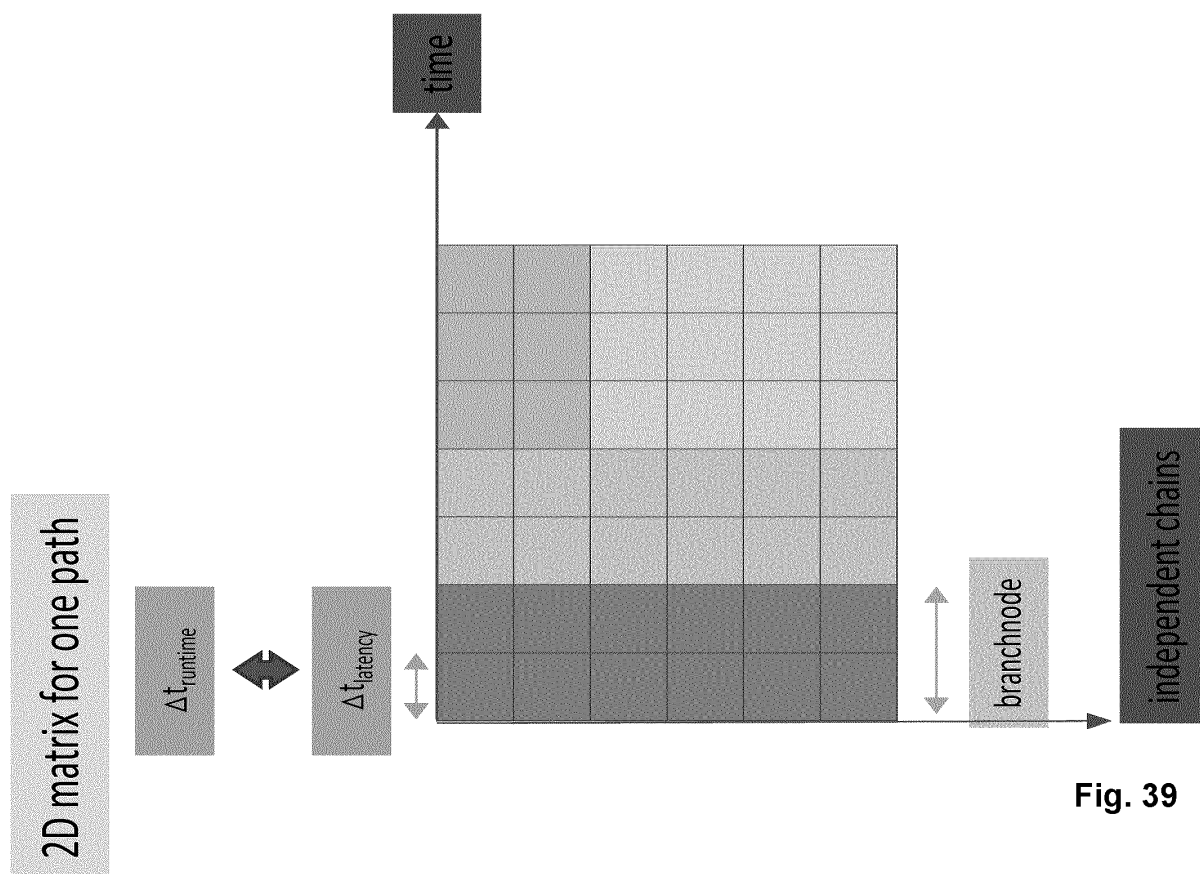
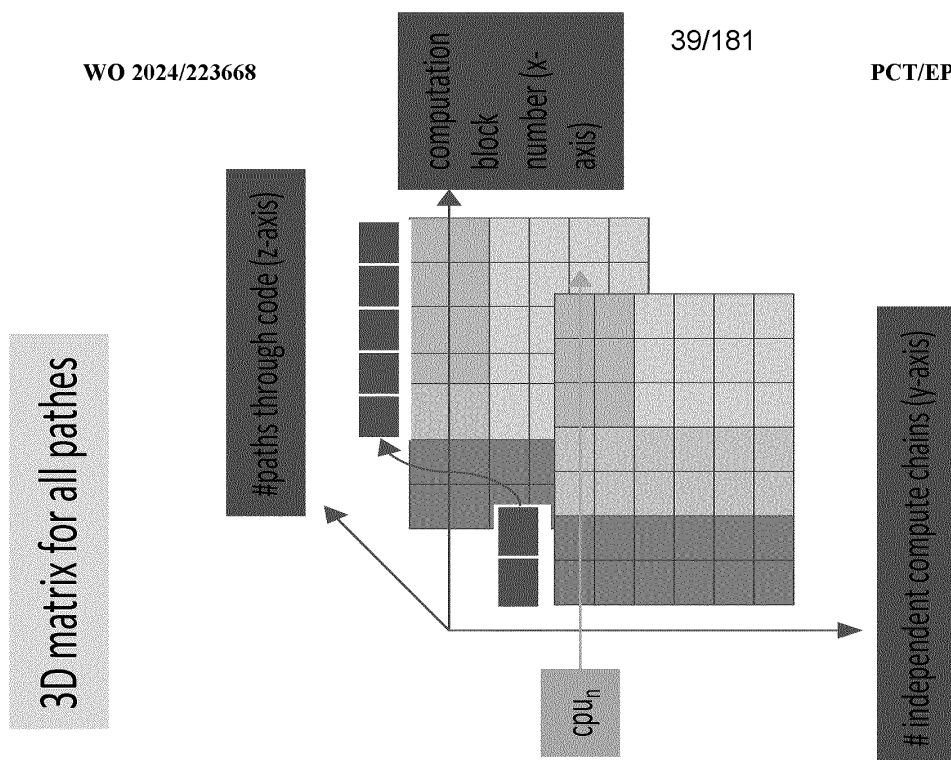


Fig. 39



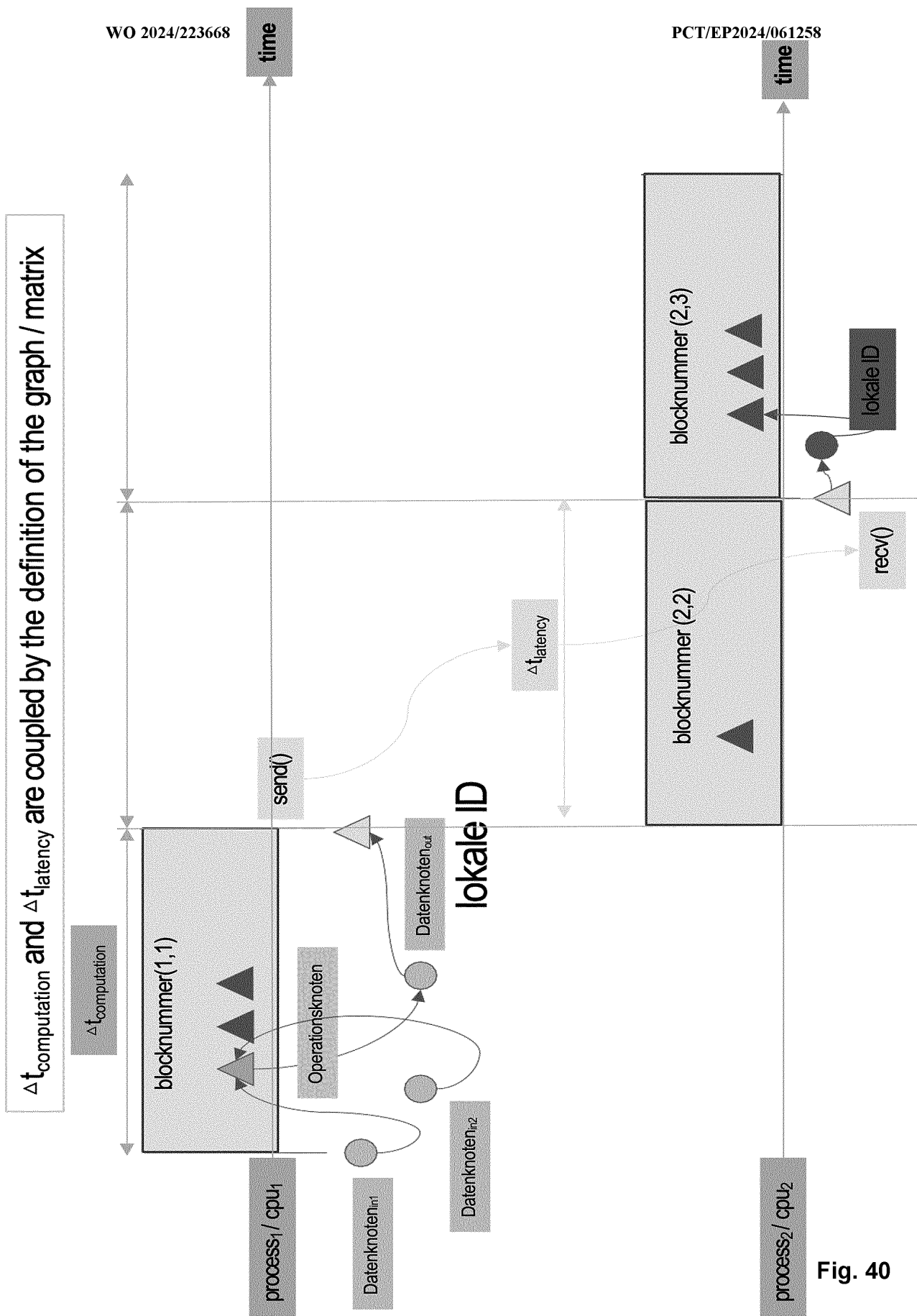
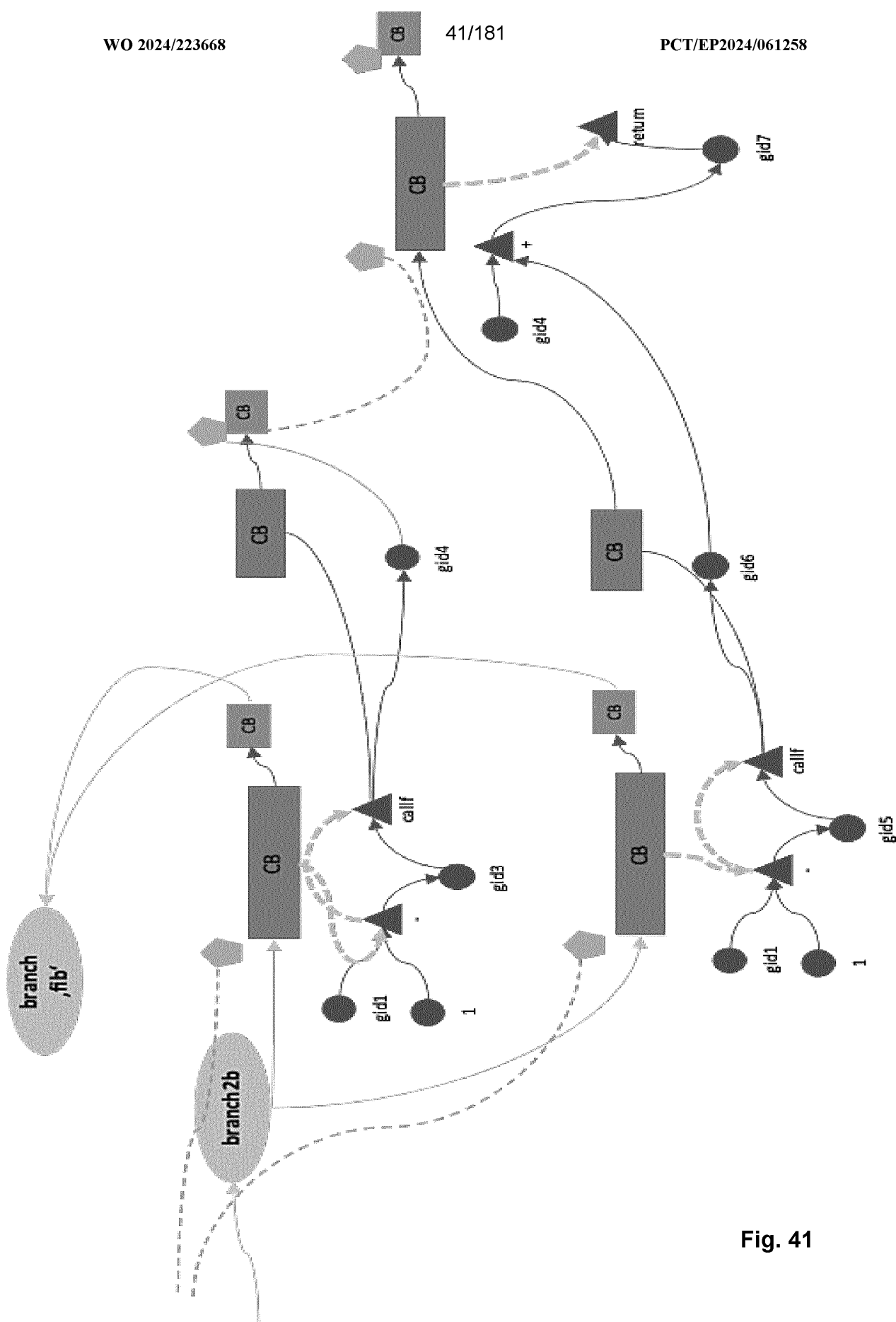


Fig. 40



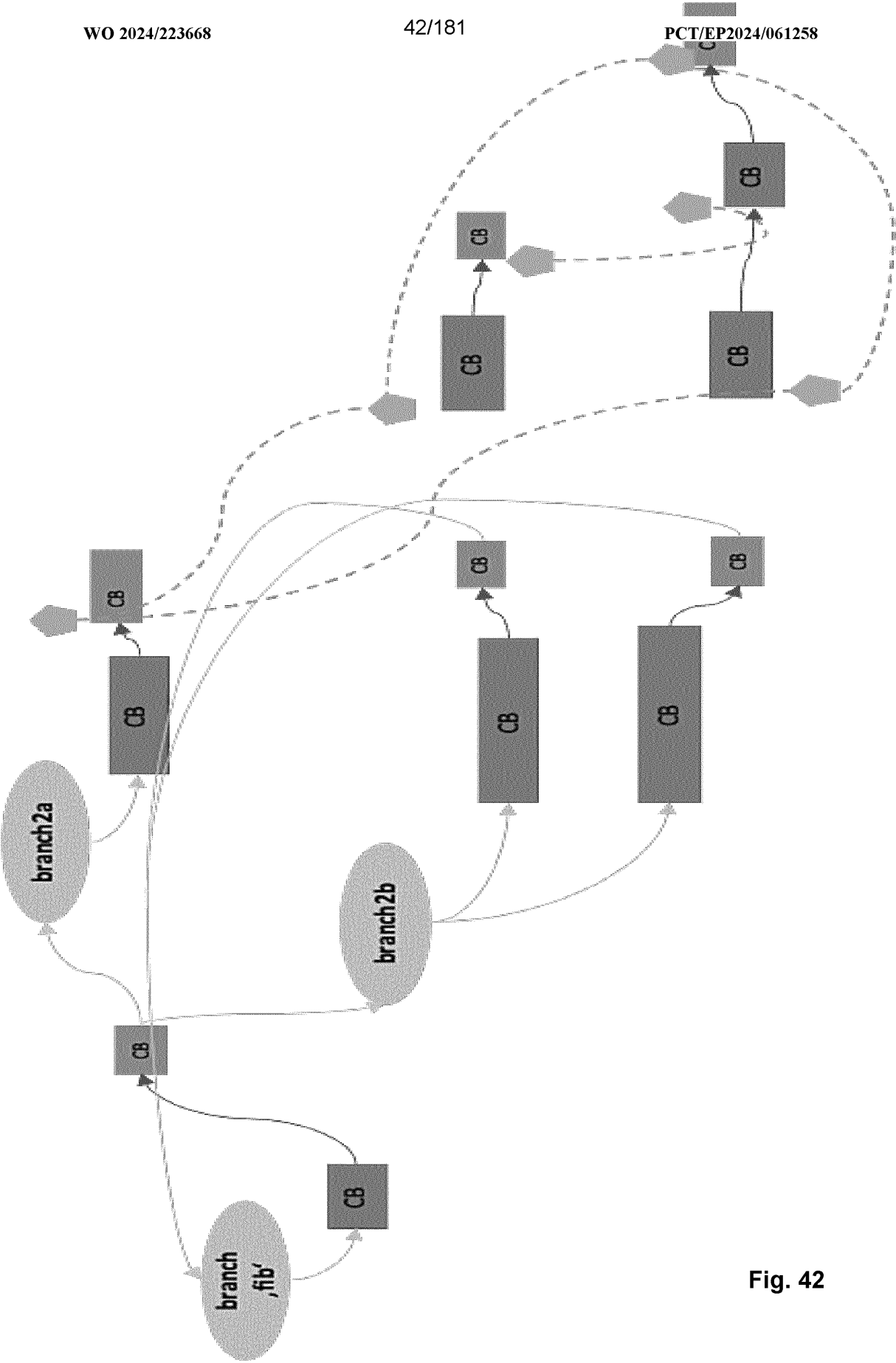


Fig. 42

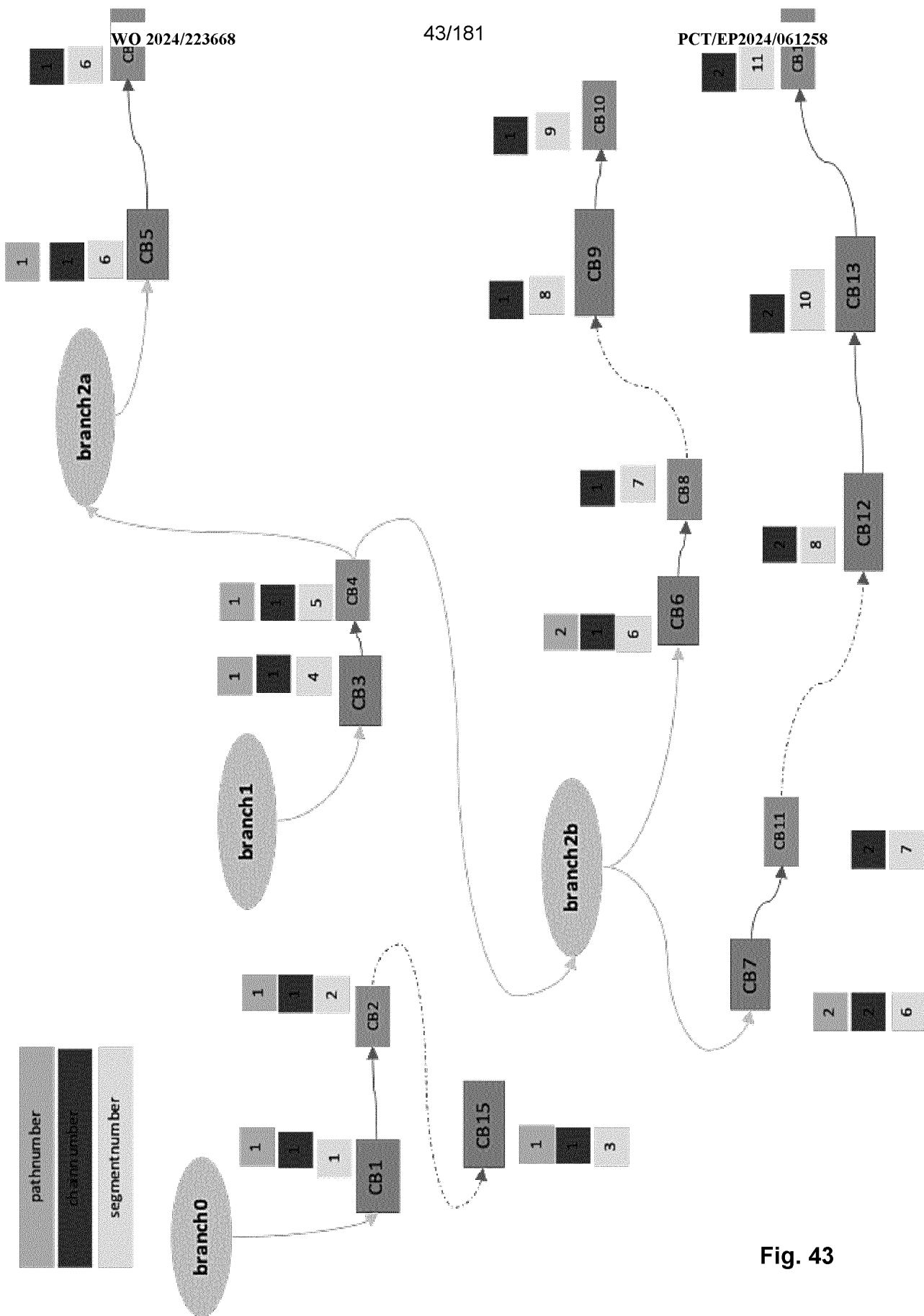


Fig. 43

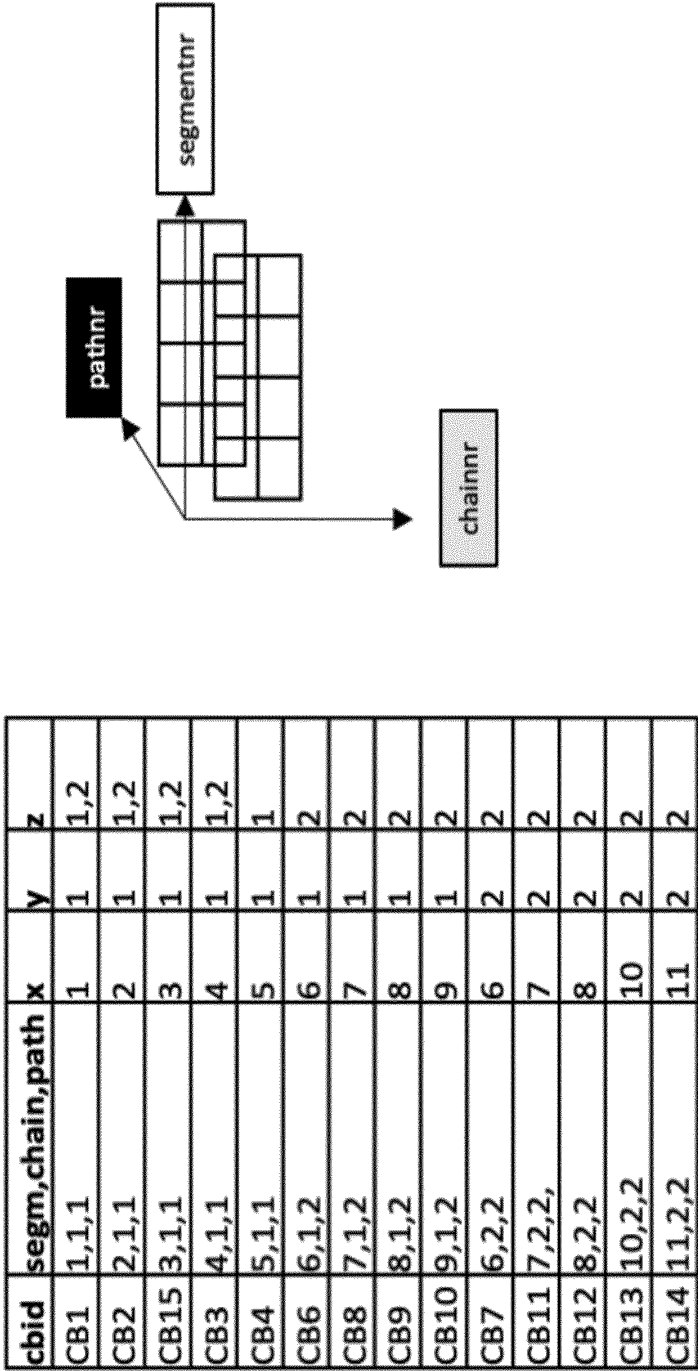
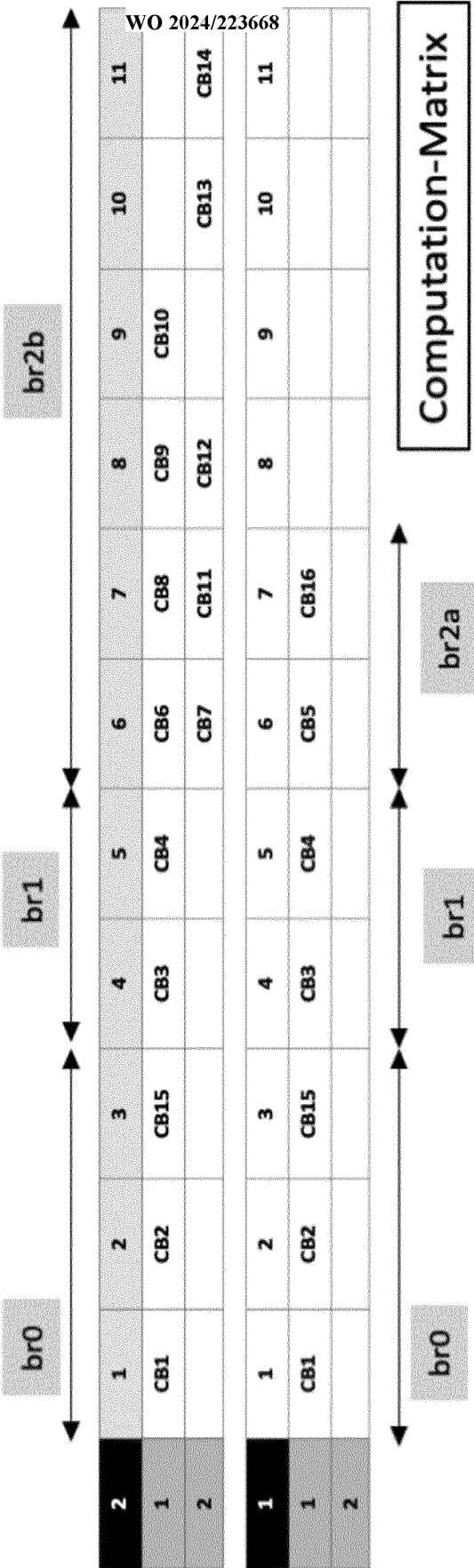


Fig. 44

Transfer-Matrix

2	1	2	3	4	5	6	7	8	9	10	11
1	e: COM1->		s: ->COM10	e: SIG1-> COM2-> s: ->COM1 ->COM4 ->COM5		s: ->SIG1 ->COM2 e: COM4->		s: ->COM8	s: COM6->		WO 2024/223668
2						s: ->SIG1 ->COM2 e: COM5->		s: ->COM9		s: ->COM6	s: COM7->

1	2	3	4	5	6	7	8	9	10	11
1	e: COM1->	s: ->COM10	e: SIG1-> COM2-> s: ->COM1 ->COM4 ->COM5		s: ->SIG1 ->COM2 e: COM3->					
2										

Fig. 45

	source cbid		target cbid		direction
	start	end	start	end	
COM1	CB1 - (1,1,1)		CB3 - (4,1,1)		forward
SIG1	CB3 - (4,1,1)		CB5 - (6,1,1), CB6 - (6,1,2), CB7 - (6,2,2)	gid8, addr CB15	forward
COM2	CB3 - (4,1,1)		CB5 - (6,1,1), CB6 - (6,1,2), CB7 - (6,2,2)	gid2	forward
COM3	CB5 - (6,1,1)		CB15 / CB9 / CB12	gid1	forward
COM4	CB6 - (6,1,2)		CB3 - (4,1,1)	gid1	backward
COM5	CB7 - (6,2,2)		CB3 - (4,1,1)	gid3, addr CB9	backward
COM6	CB9 - (9,1,2)		CB10 - (10,2,2)	gid5, addr CB12	backward
COM7	CB13 - (10,2,2)		CB15 / CB9 / CB12	gid4	forward
COM8	LL		CB9 - (8,1,2)	gid7	
COM9	LL		CB12 - (8,2,2)	gid4	
COM10	LL		CB15 - (3,1,1)	gid6	

1	2
---	---

2 procs

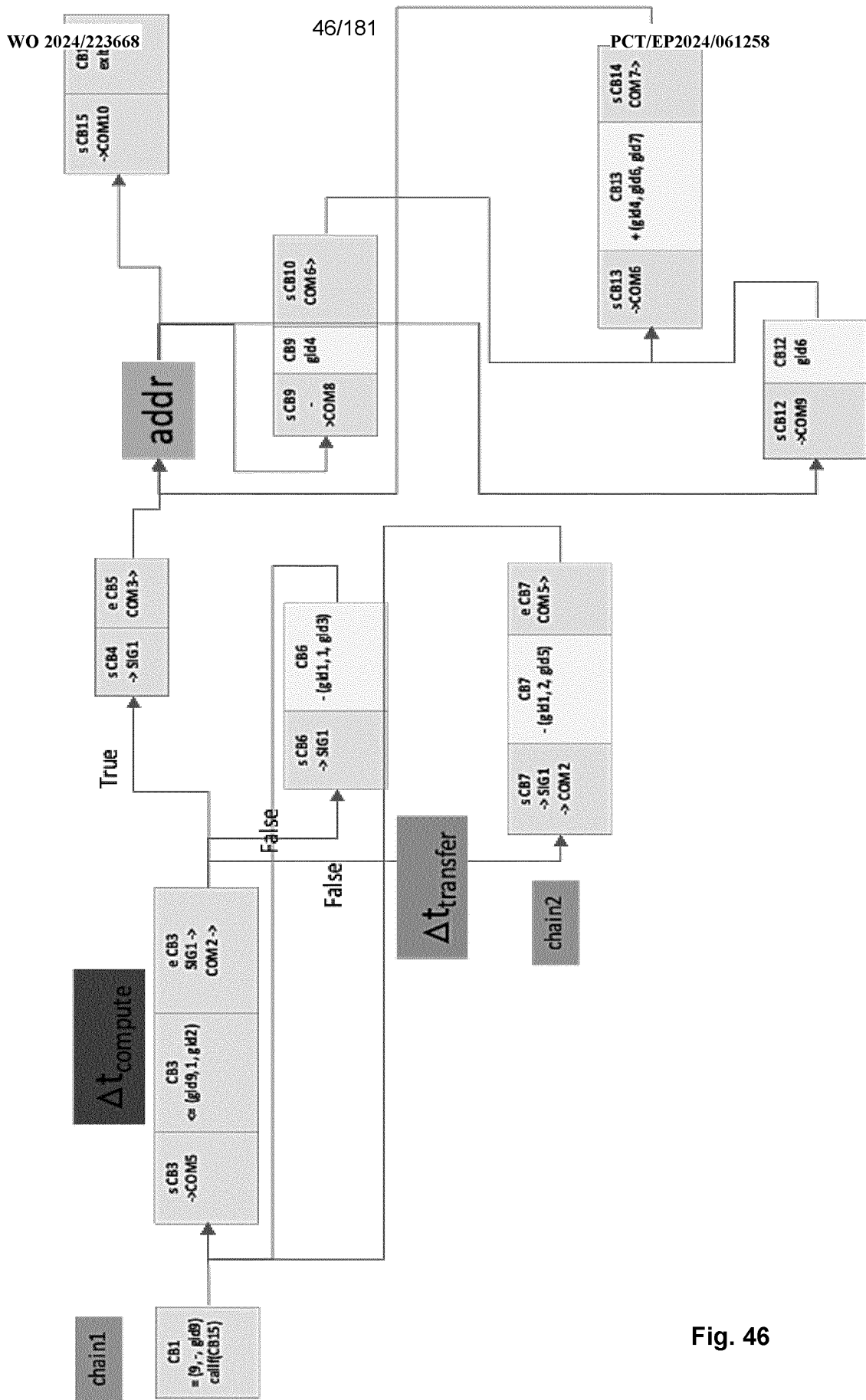


Fig. 46

callf (in1, targetcb, out)

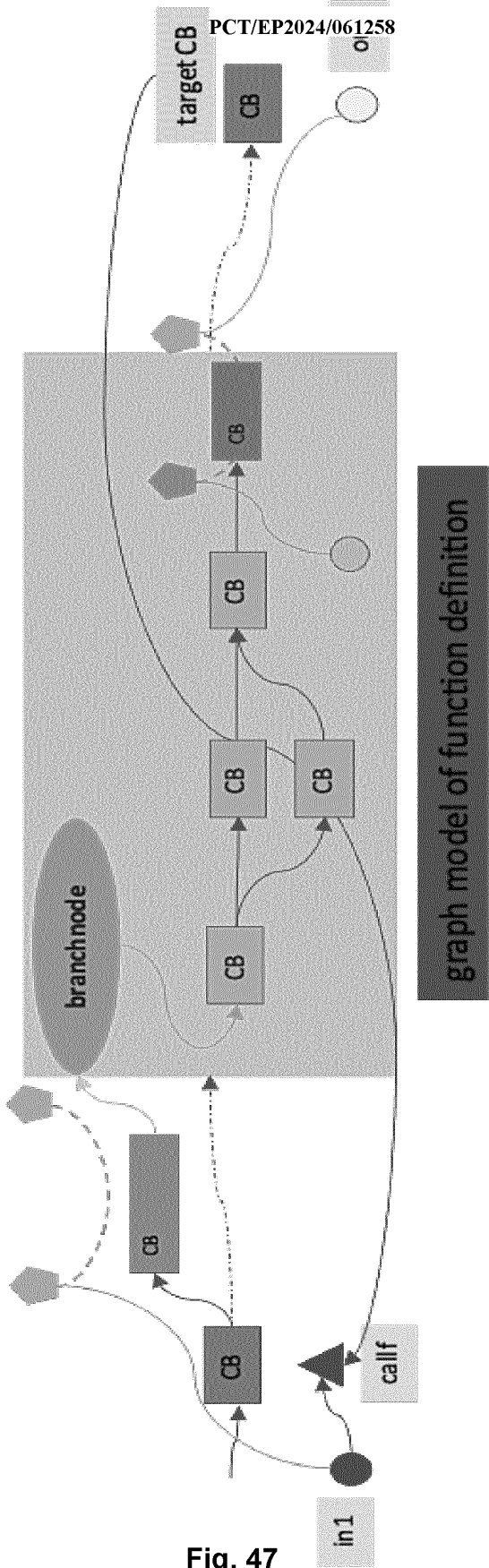
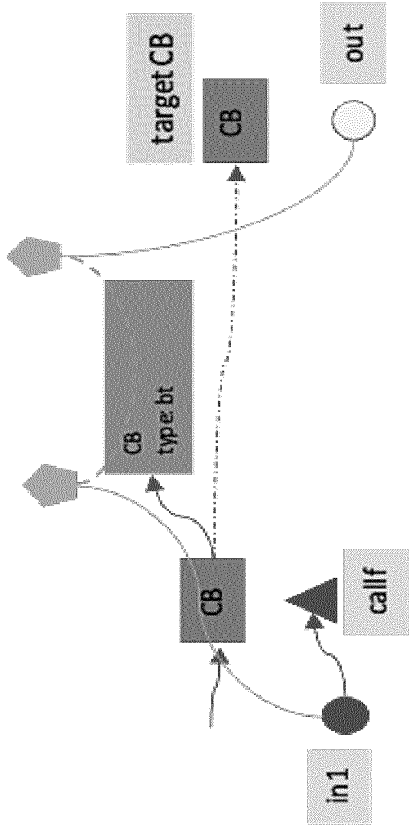


Fig. 47

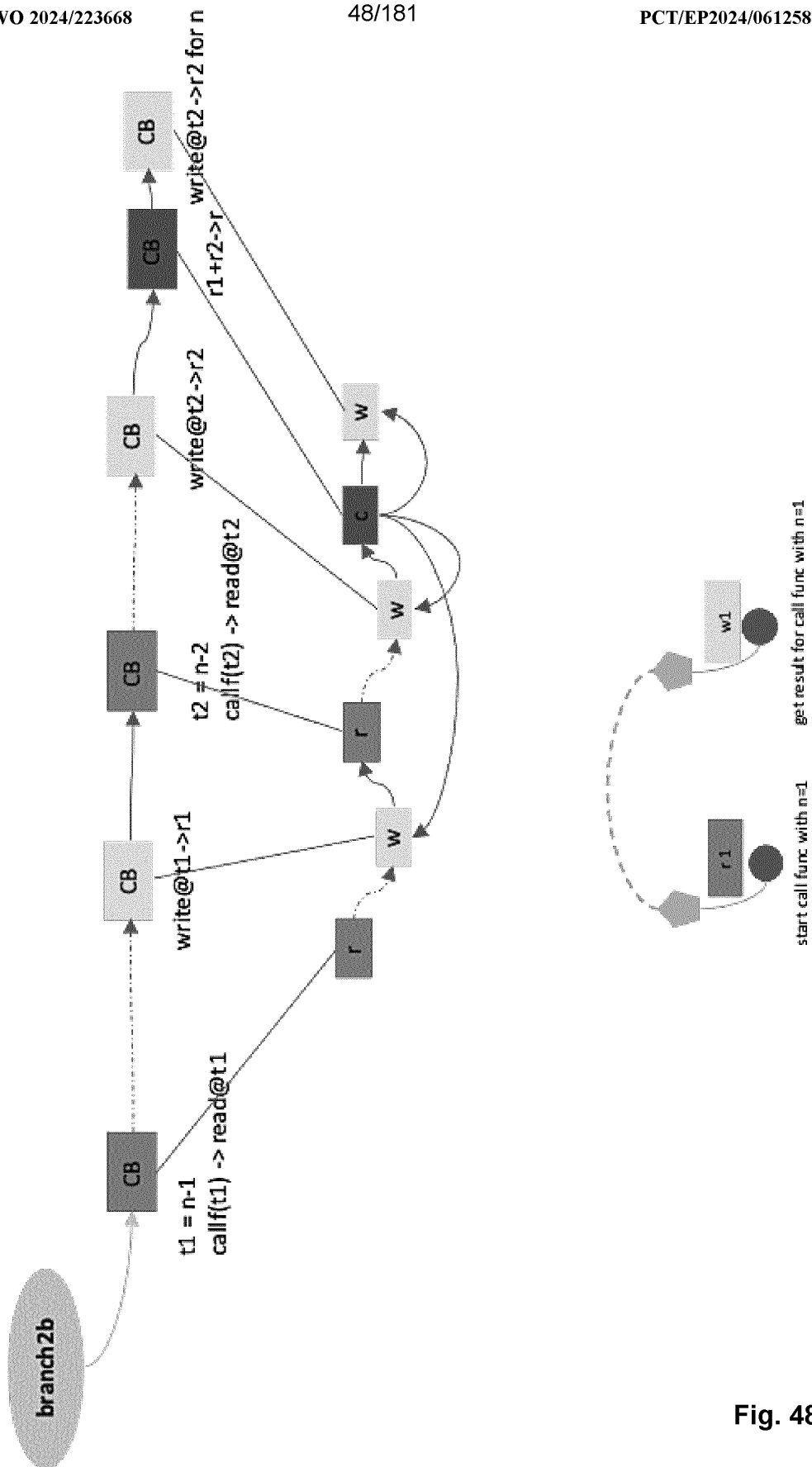


Fig. 48

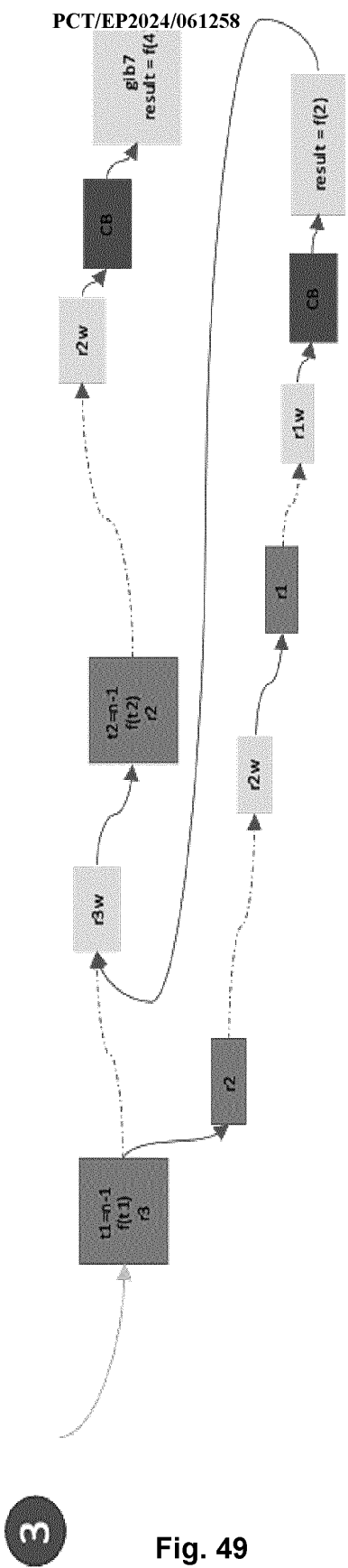
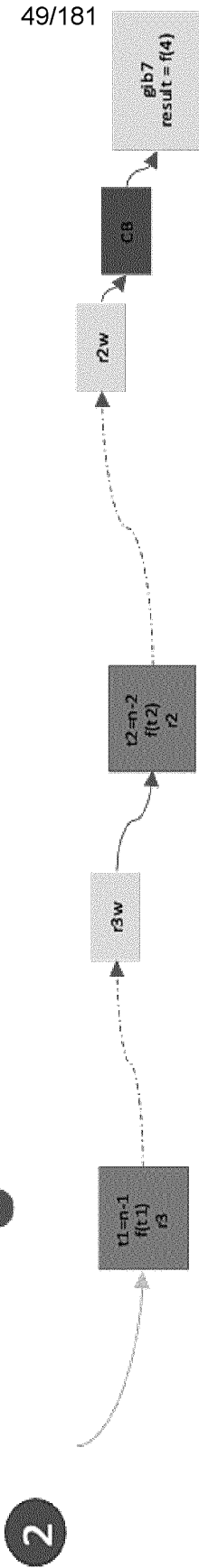
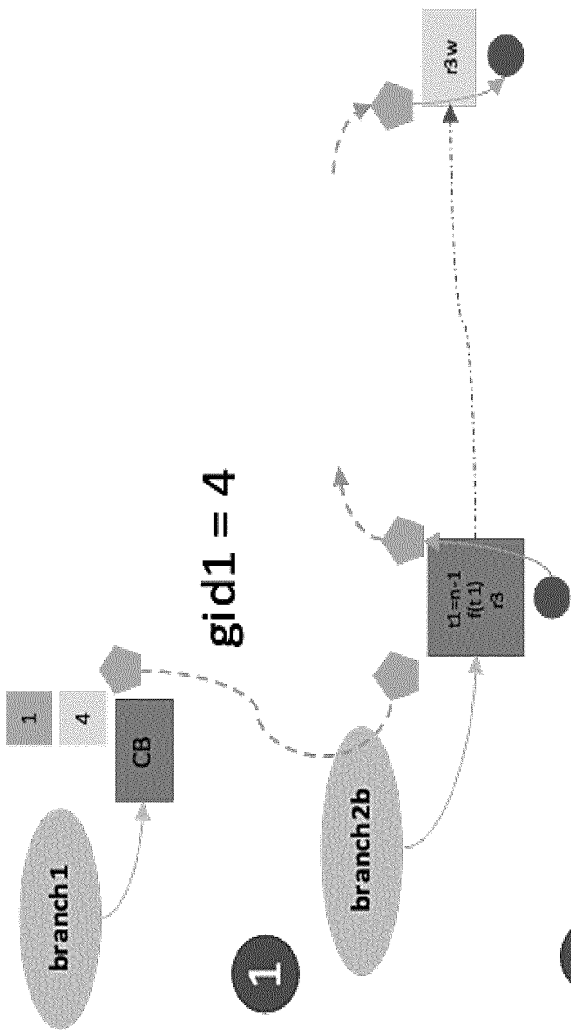


Fig. 49

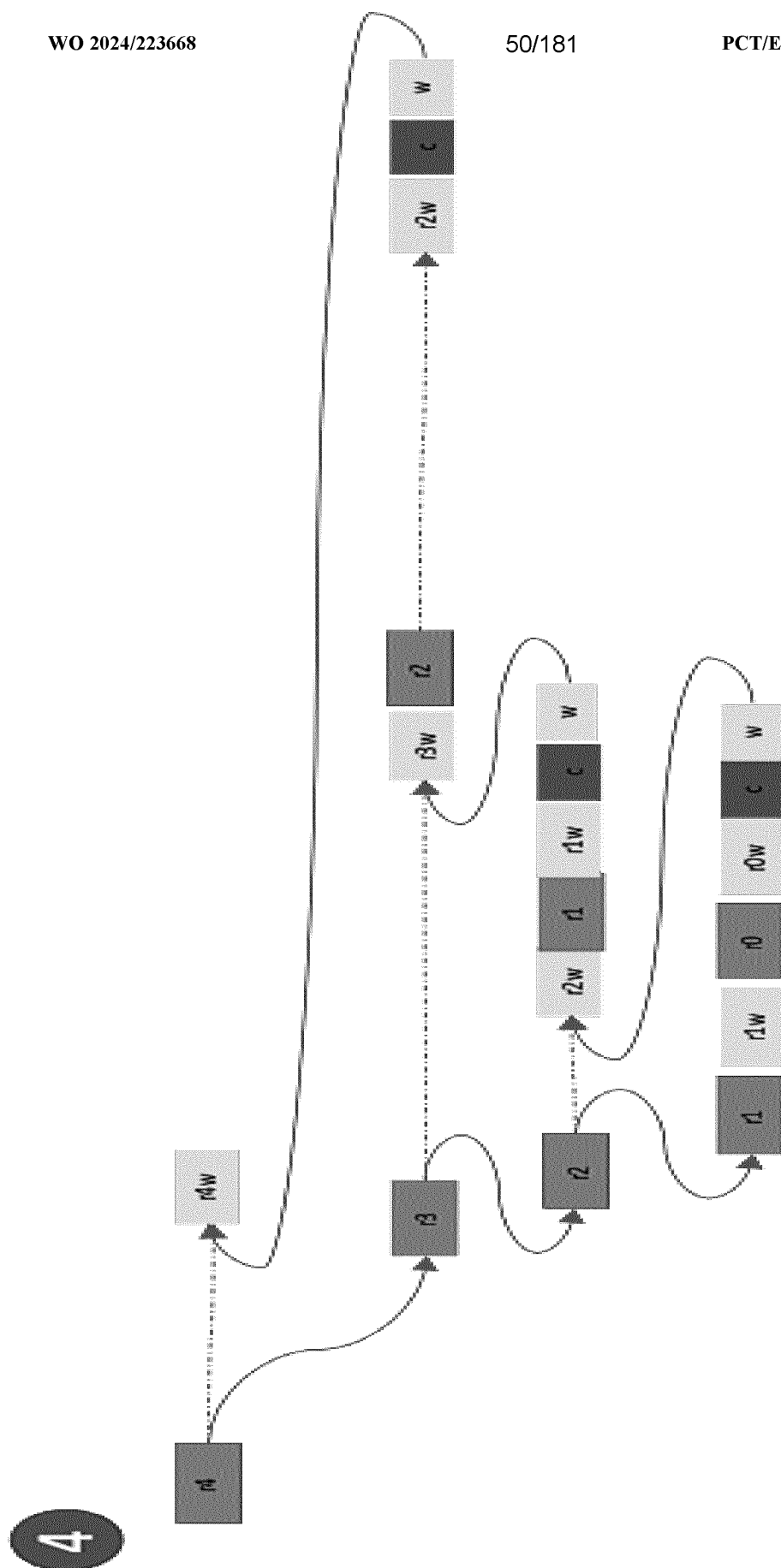


Fig. 50

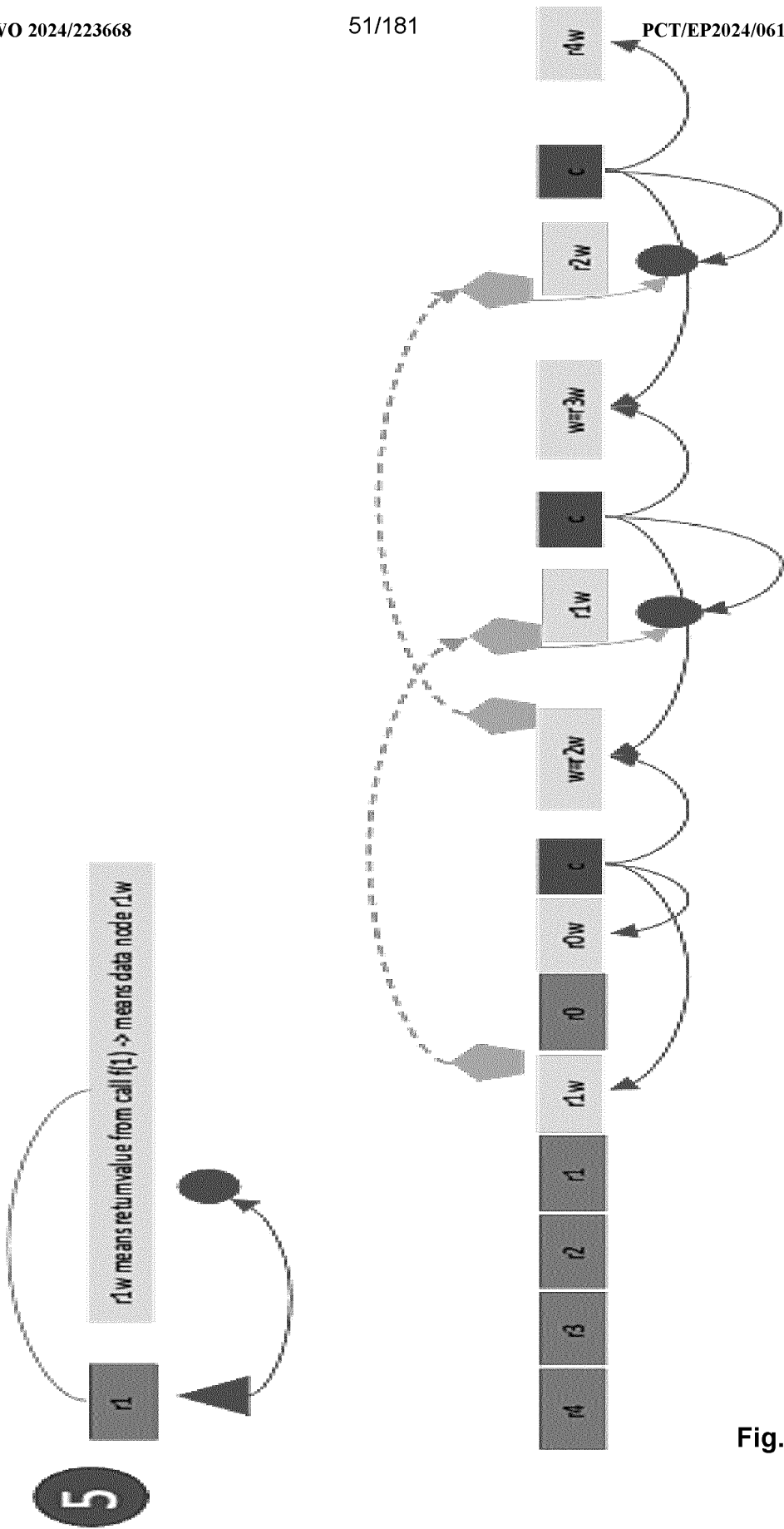


Fig. 51

6

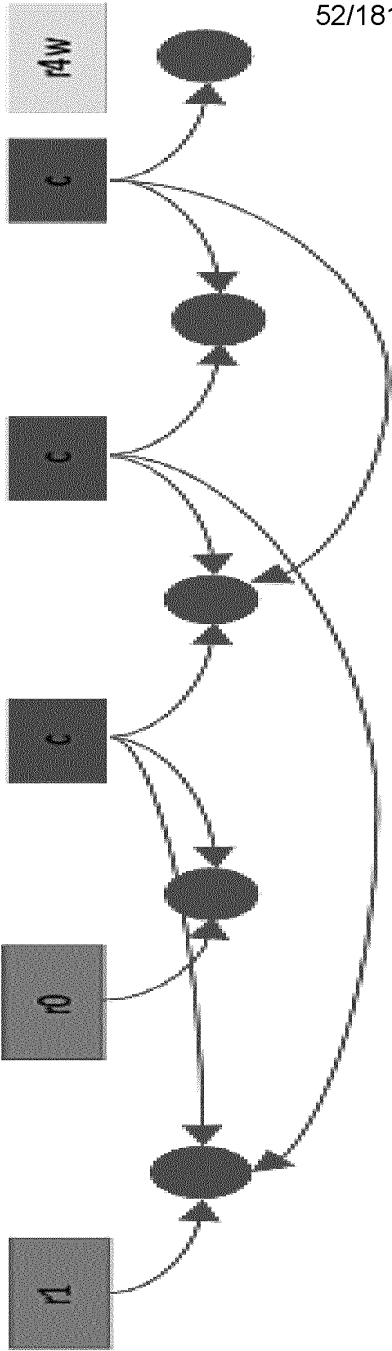


Fig. 52

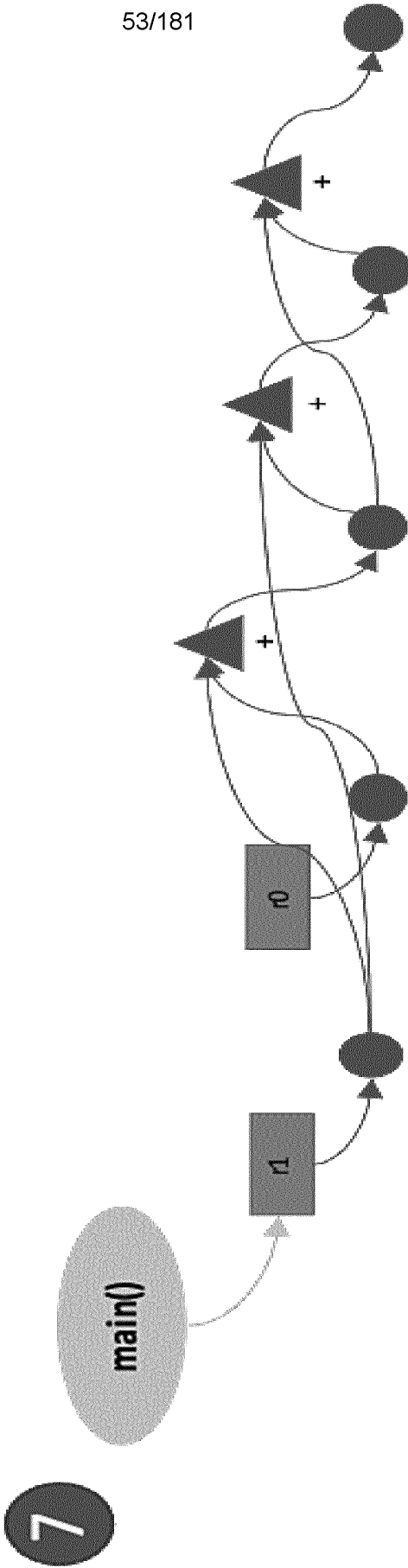


Fig. 53

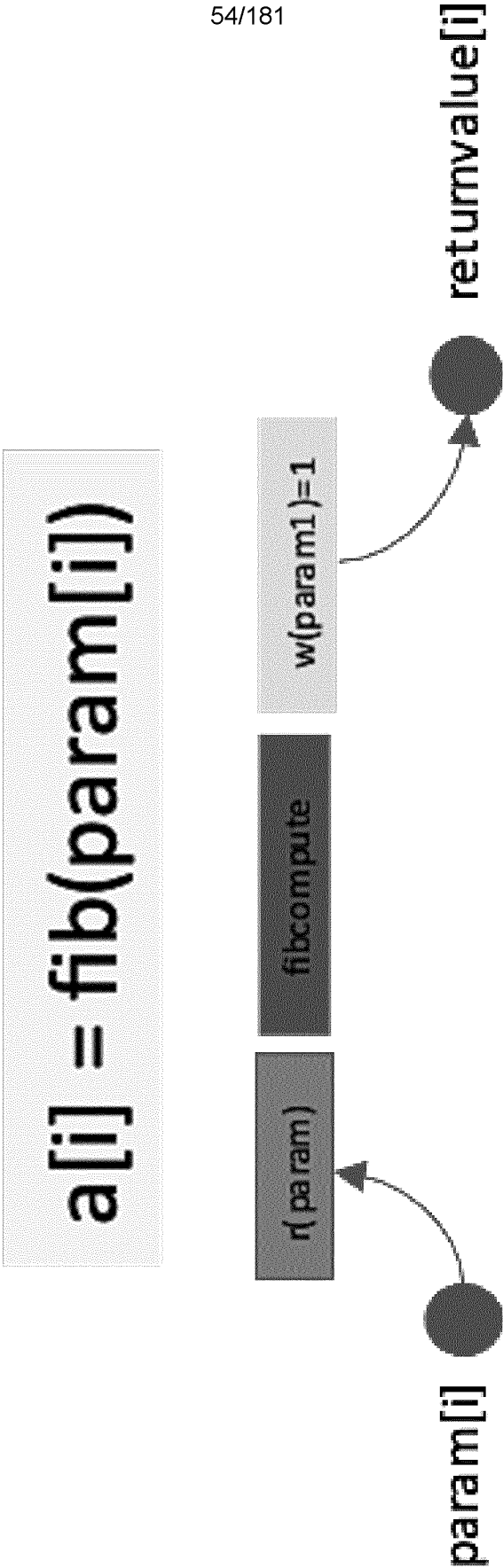


Fig. 54

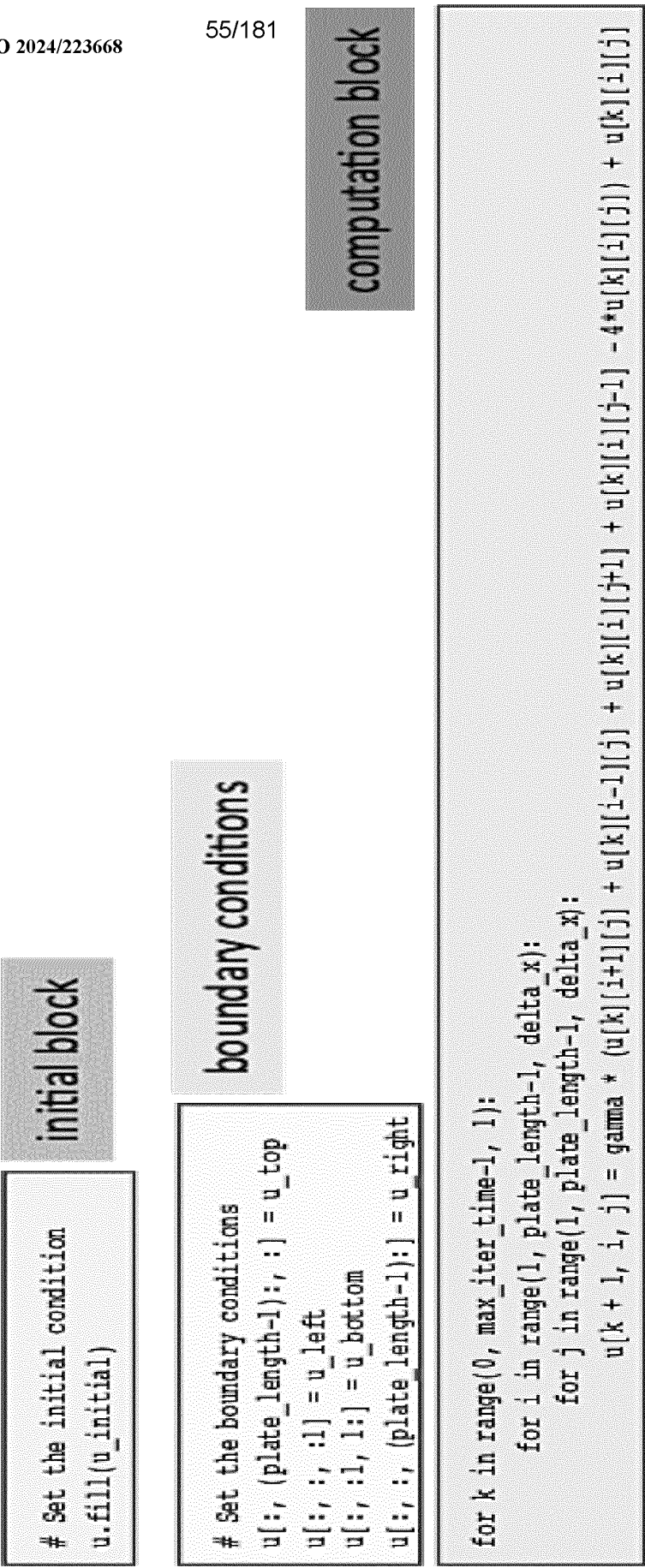
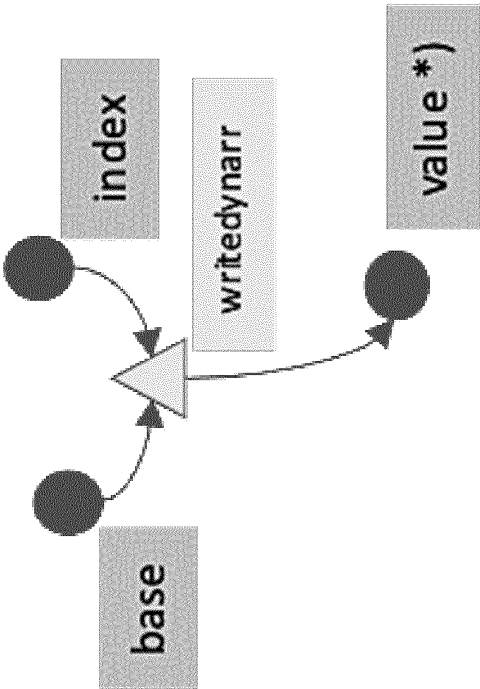


Fig. 55

`base[index] = *base+index = value`
`writedynarr(base, index) = value`



`value = base[index] = *base+index`
`value = readdynarr(base, index)`

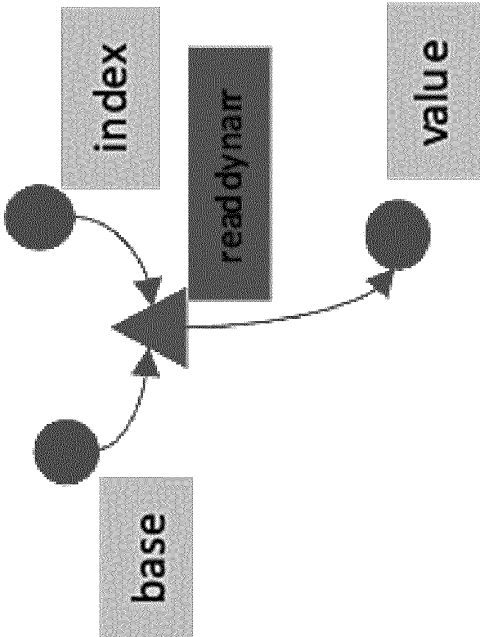
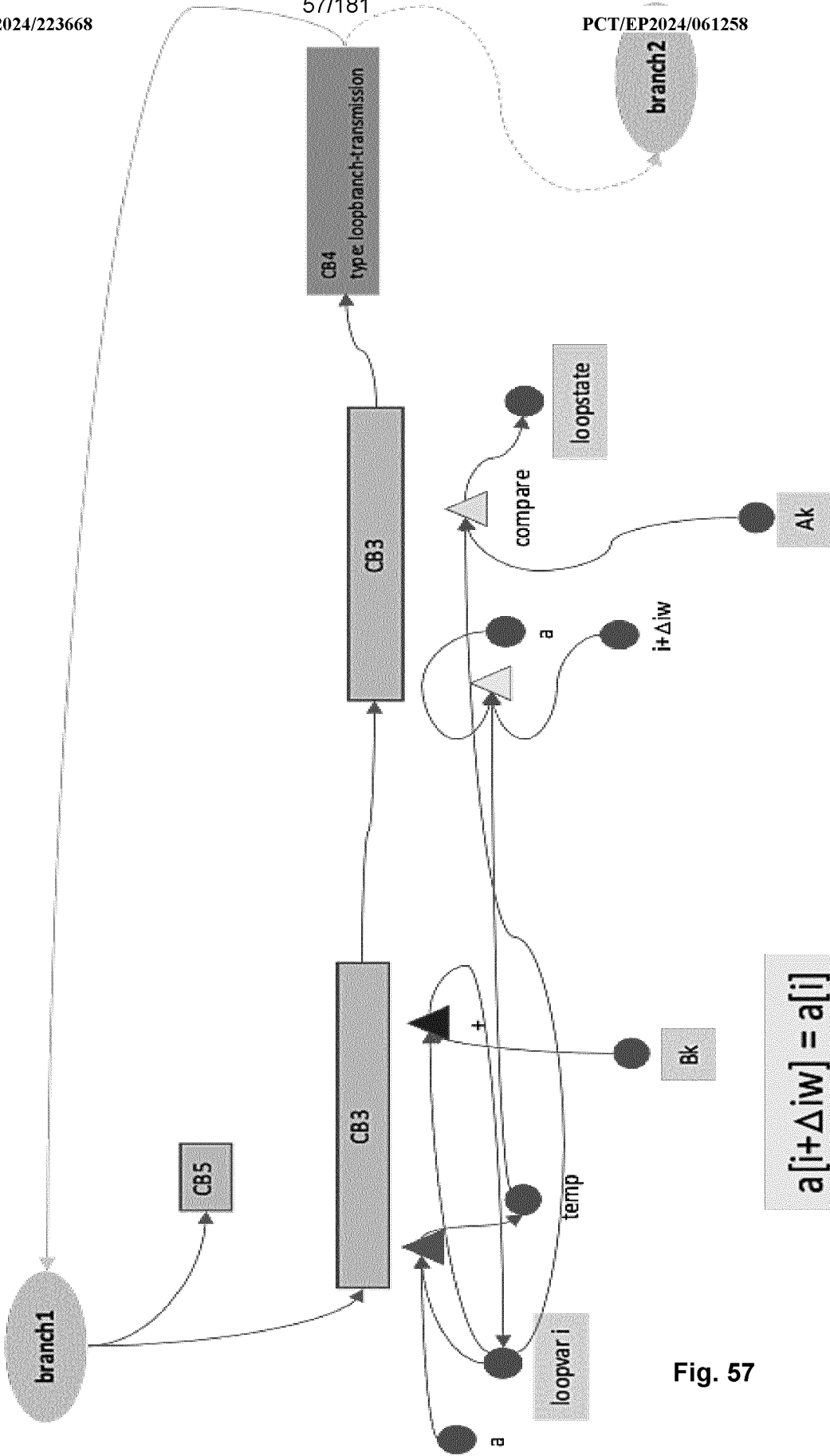


Fig. 56

for k in range(k0, Ak, Bk):

$$n_k = \left(\frac{A_k - k_0}{B_k}\right) + 1$$



$$a[i+\Delta iw] = a[i]$$

Fig. 57

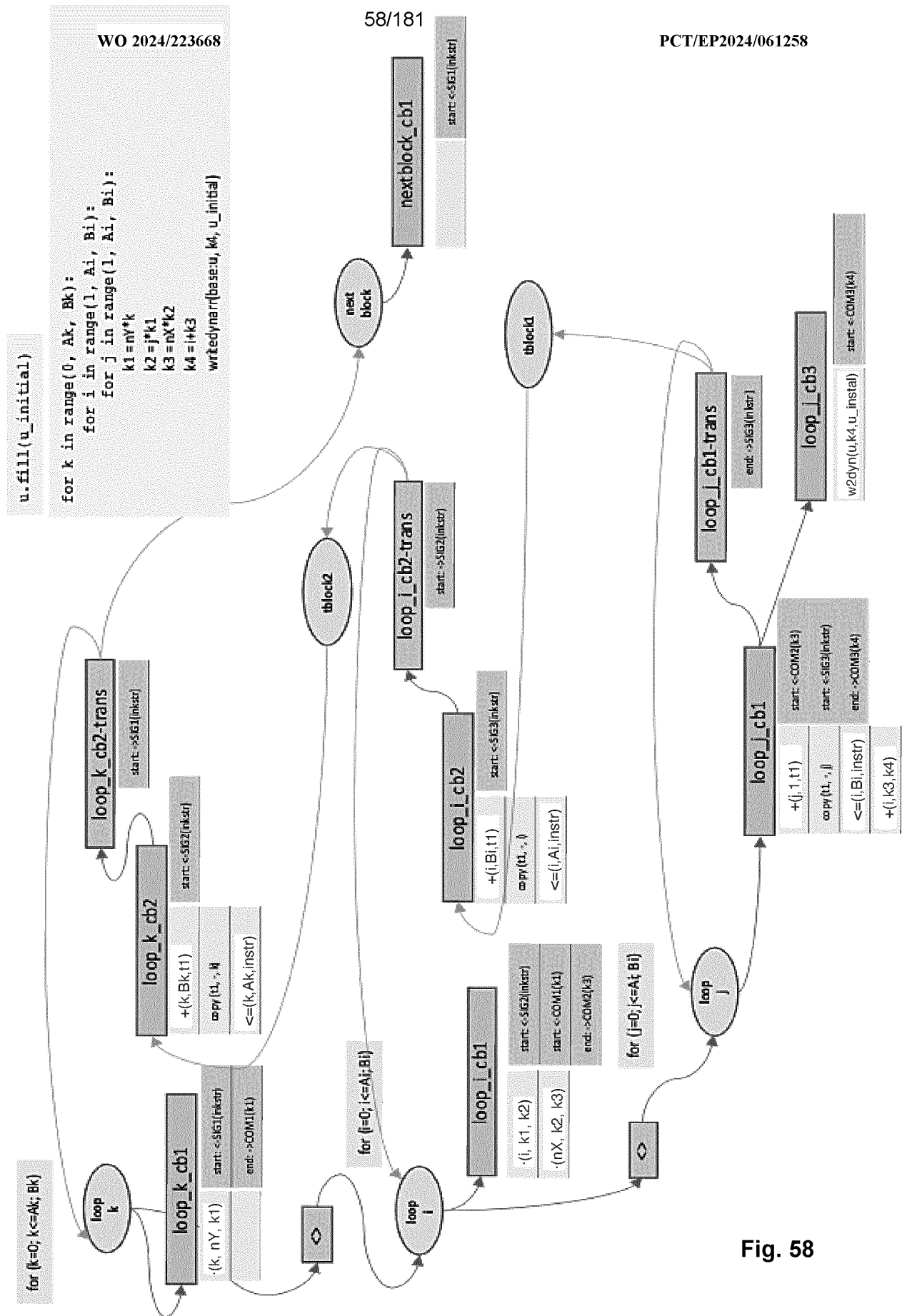


Fig. 58

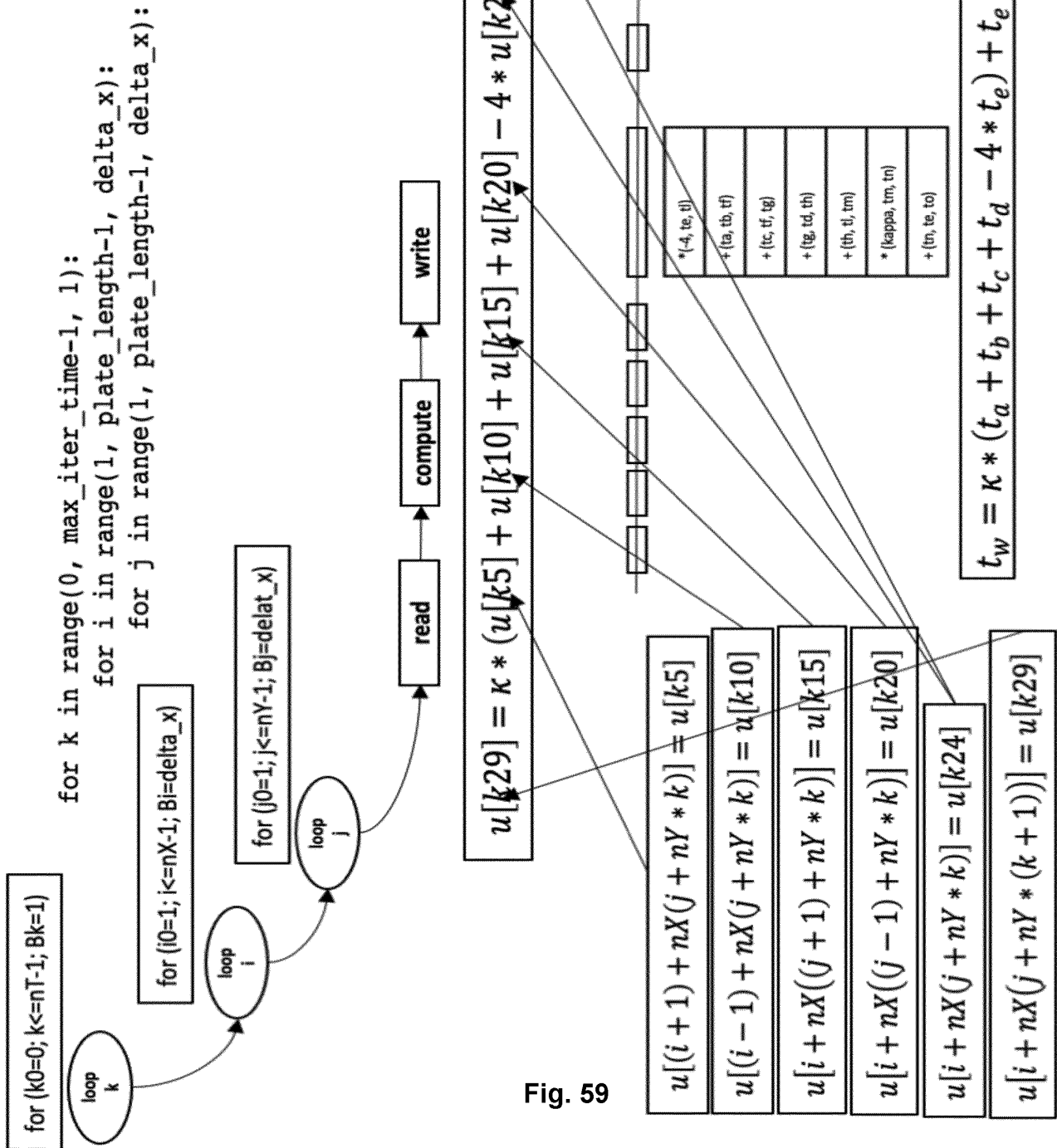


Fig. 59

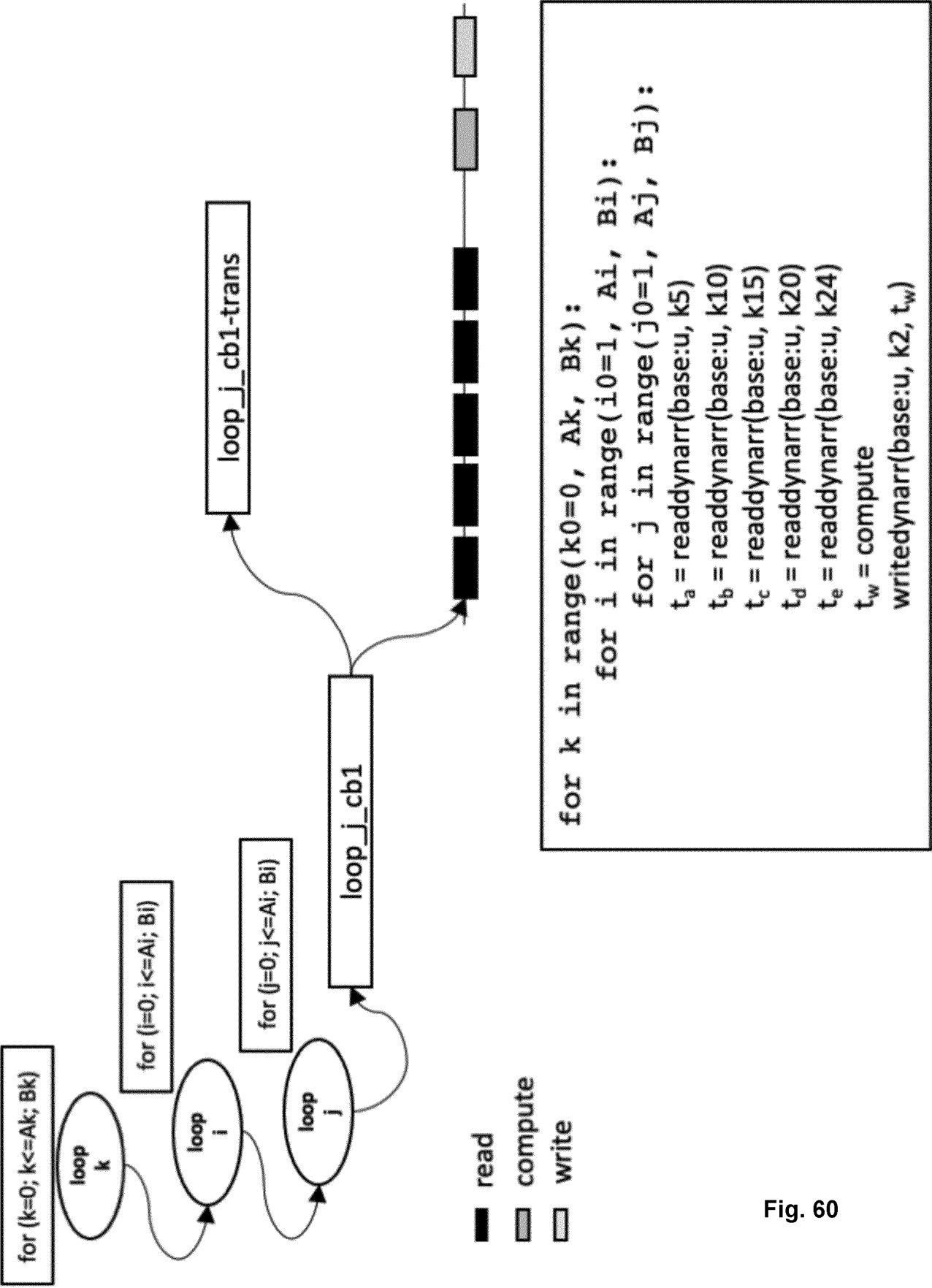


Fig. 60

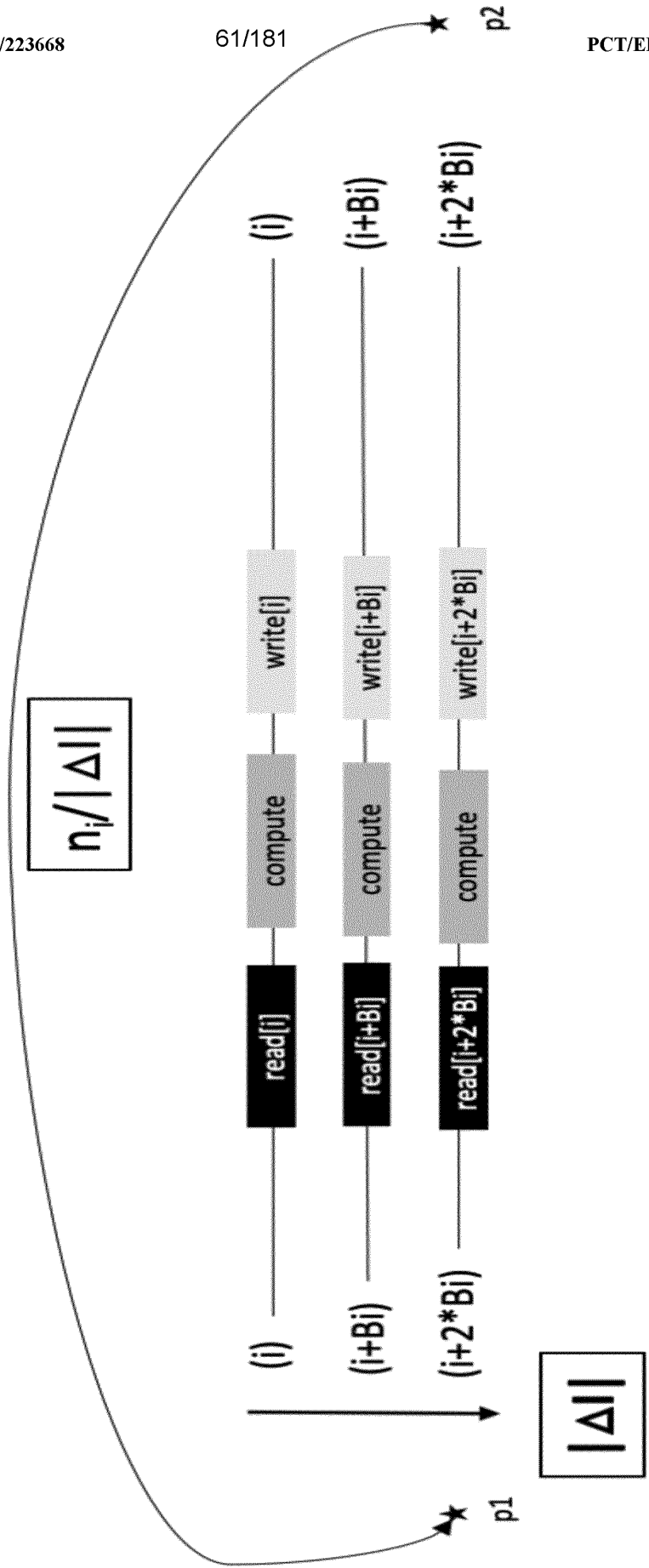


Fig. 61

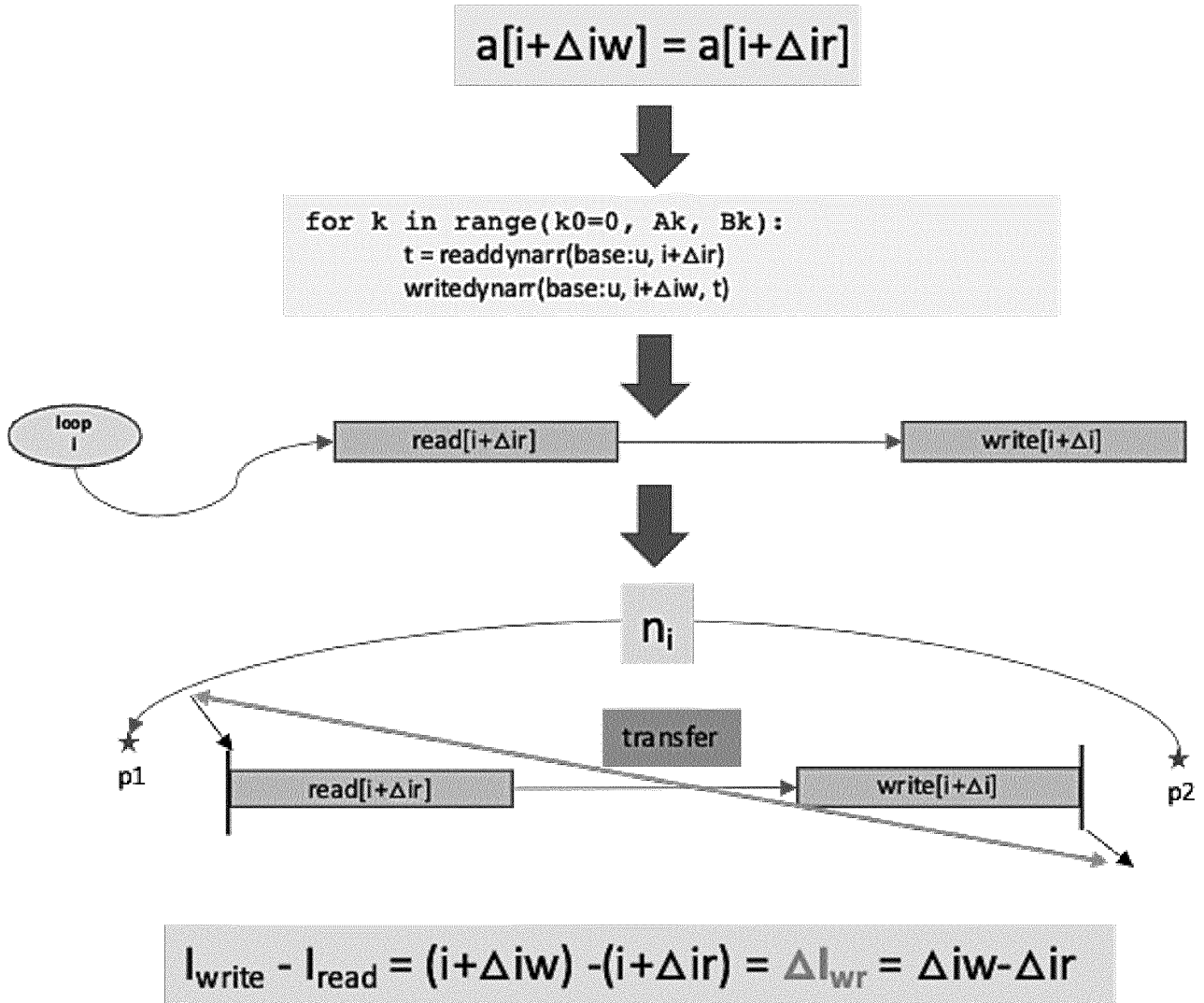


Fig. 62

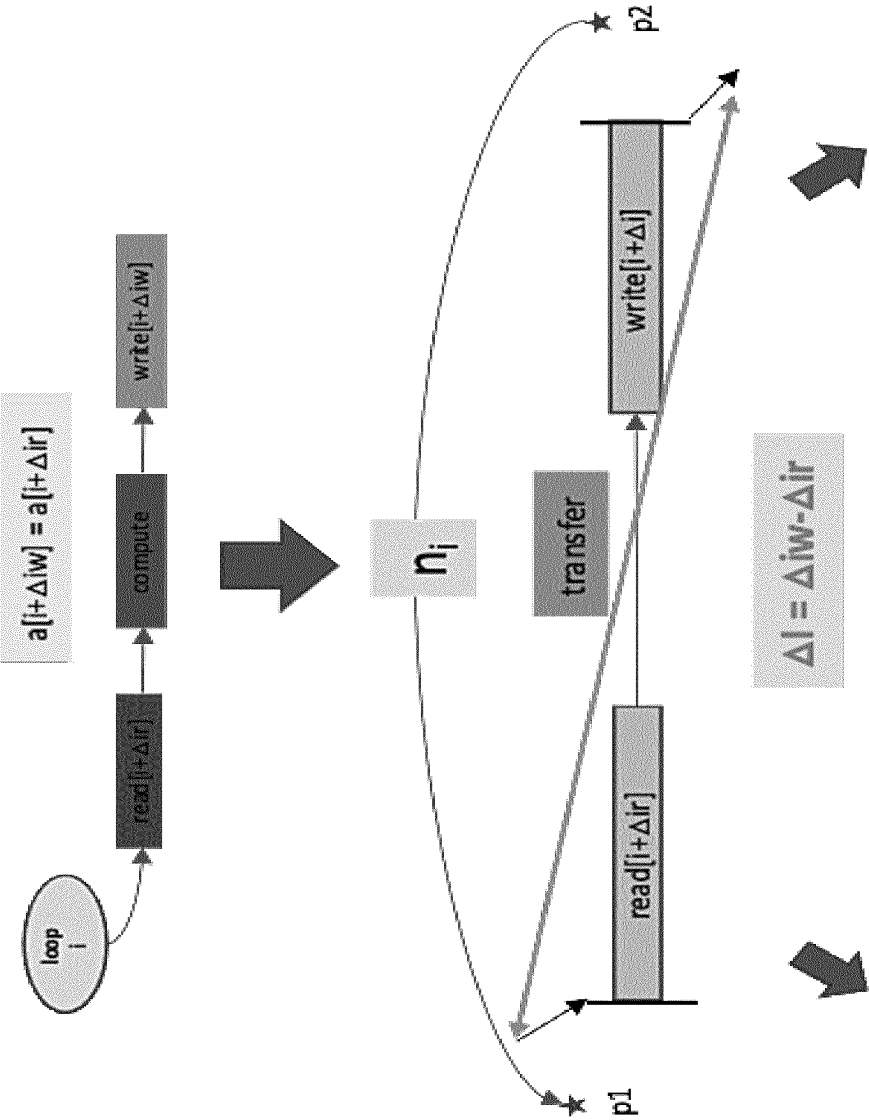


Fig. 63

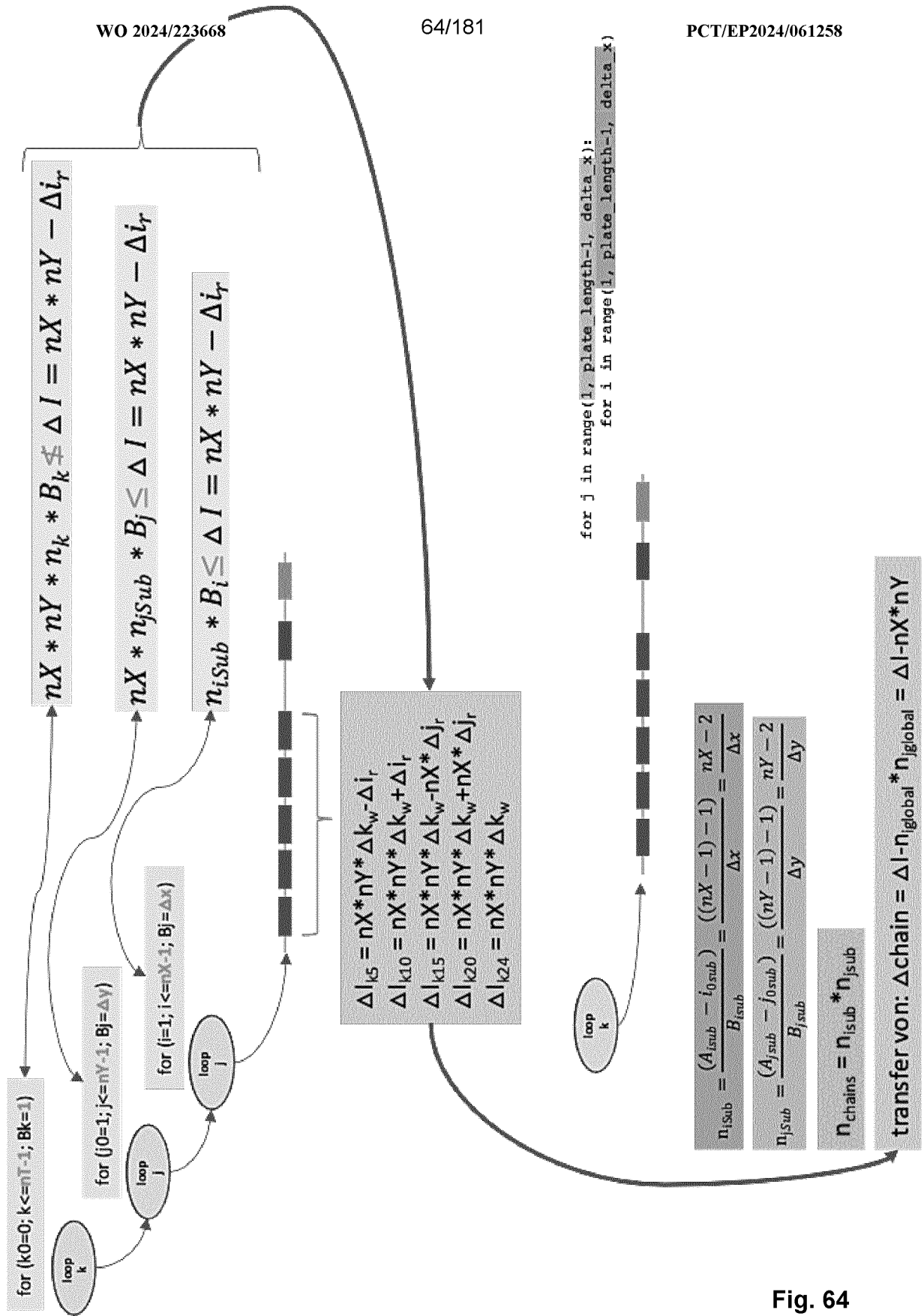


Fig. 64

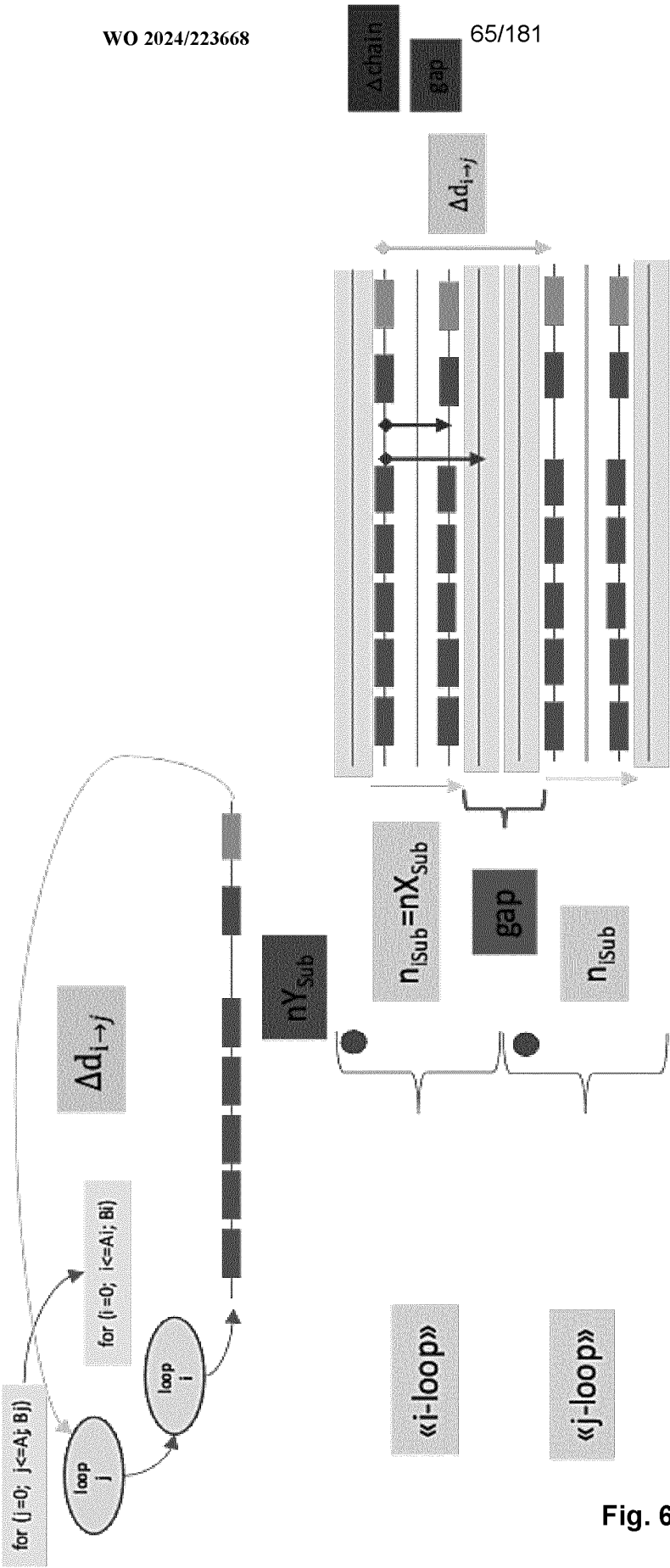


Fig. 65

$$\Delta s_i = \frac{(l_{0sub} - l_0)}{B_{1sub}} = \frac{1}{\Delta x}$$

$$\Delta e_i = \frac{(A_i - A_{1sub})}{B_{1sub}} = \frac{1}{\Delta x}$$

$$\Delta d_{i \rightarrow j} = n_{1sub} * B_{1sub} + \Delta e_i + \Delta s_i = \left(\frac{(A_{1sub} - l_{0sub})}{B_{1sub}} \right) * B_{1sub} + \frac{(A_i - A_{1sub})}{B_{1sub}} + \frac{(l_{0sub} - l_0)}{B_{1sub}} = \frac{nX - 2}{\Delta x} + \frac{2}{\Delta x}$$

gap start i-loop

gap end i-loop

«symmetry-distance»

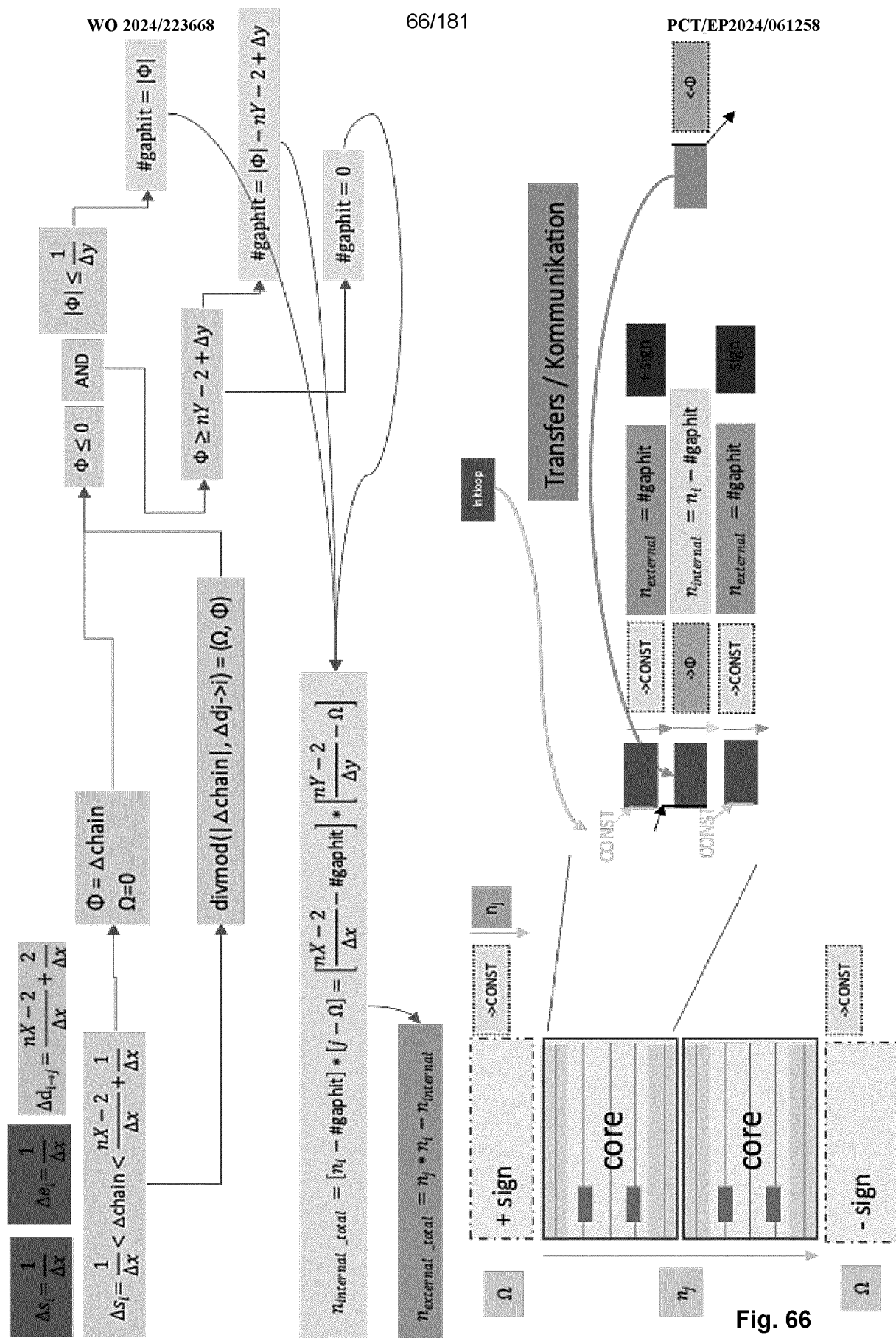


Fig. 66

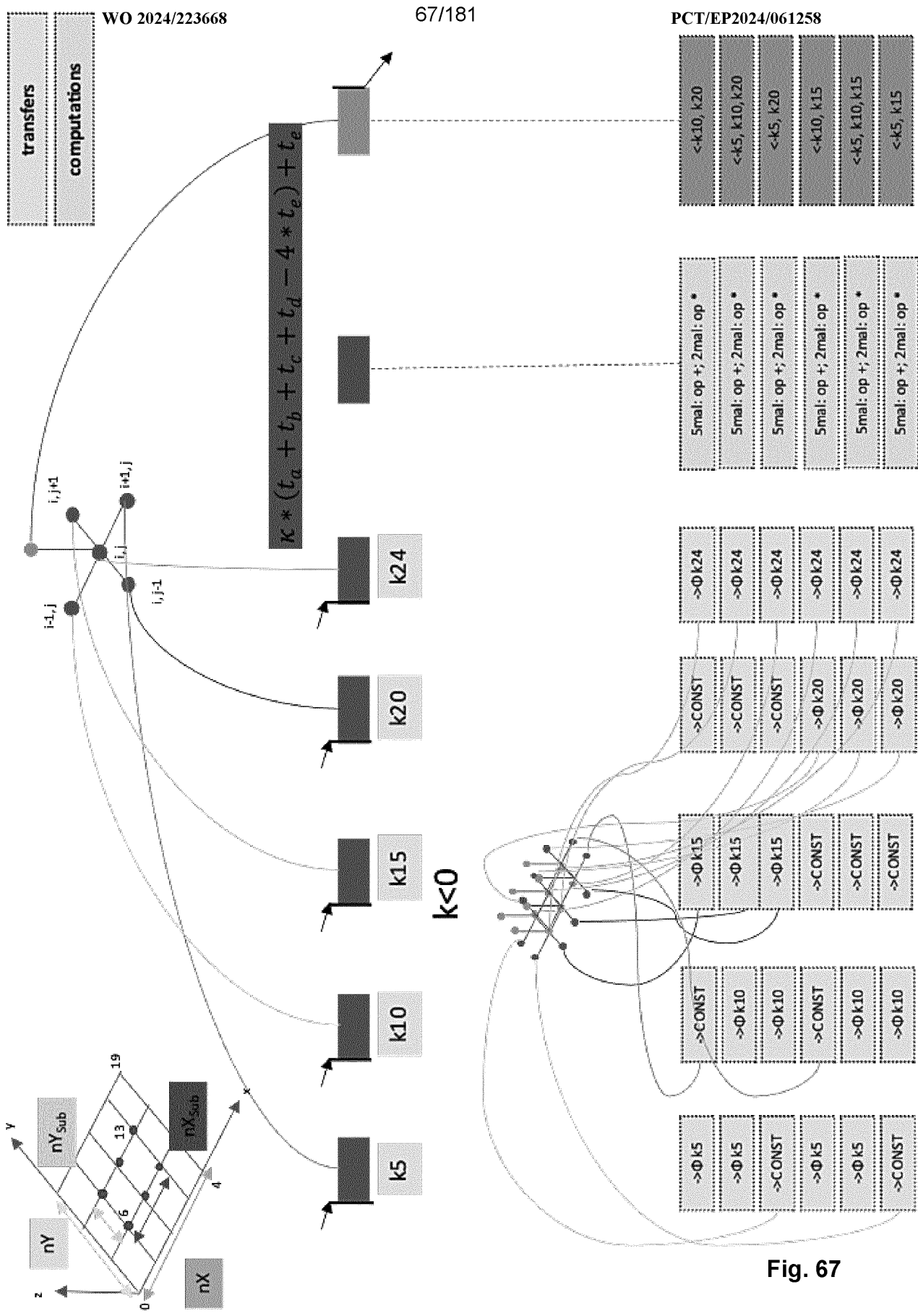


Fig. 67

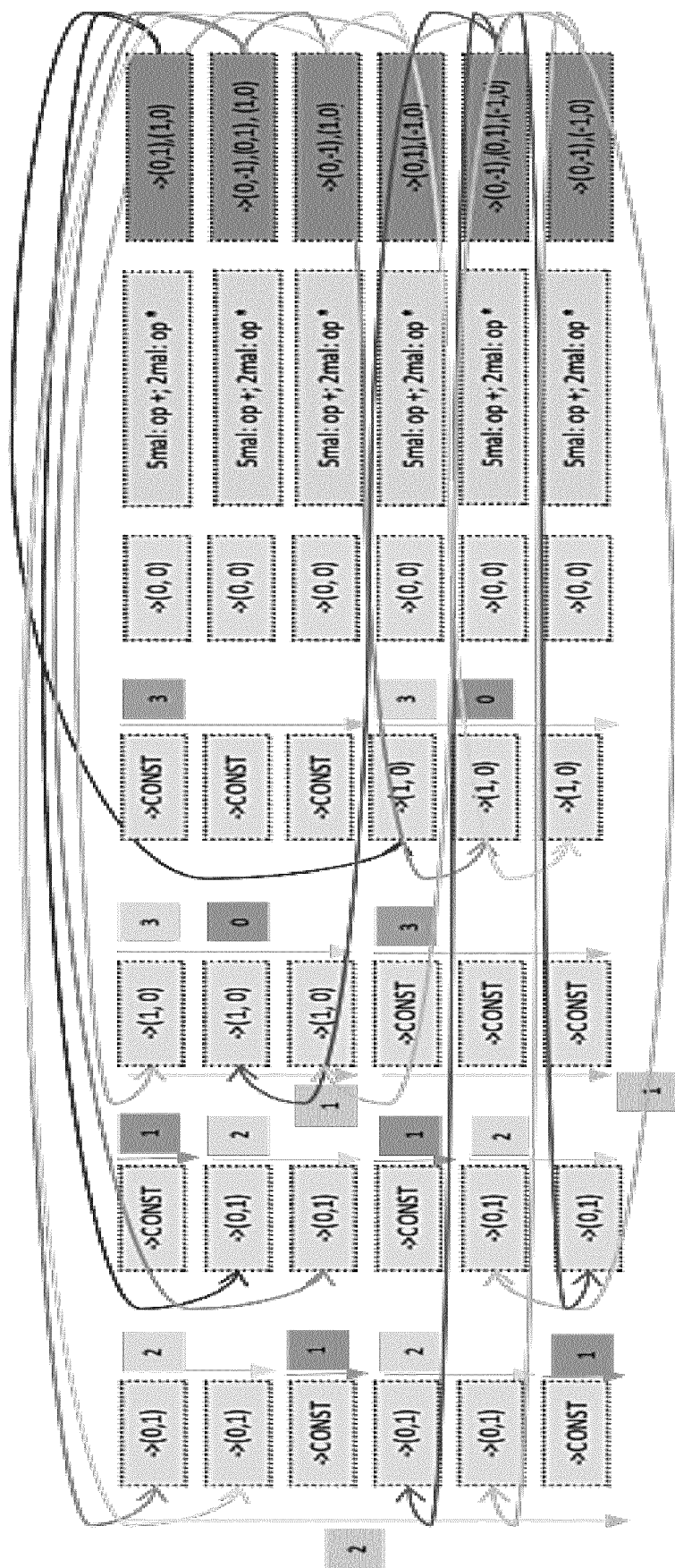


Fig. 68

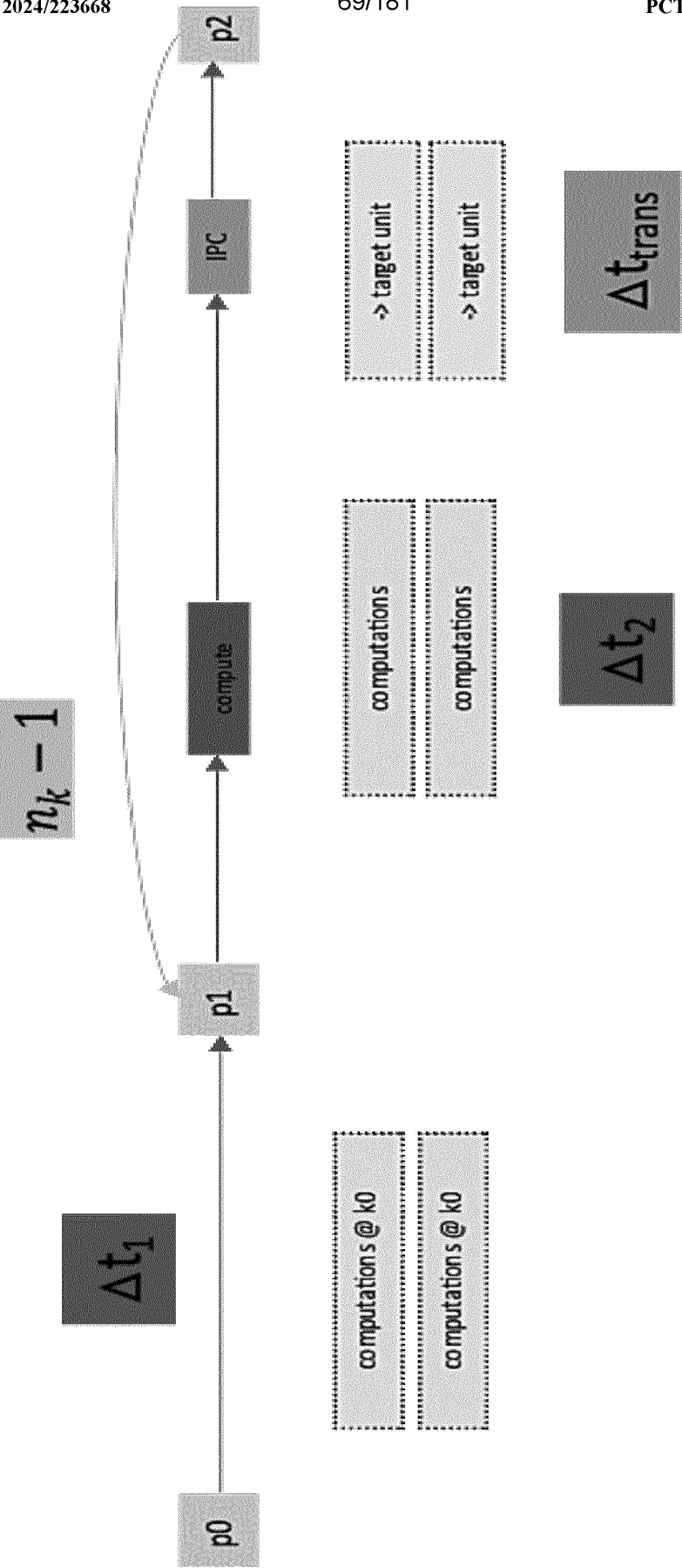


Fig. 69

2>1	0	4>1	0	1>1	5mal: op +; 2mal: op *	send: >2, >4
3>2	1>2	5>2	0	2>2	5mal: op +; 2mal: op *	send: >1, >3, >5
0	2>3	6>3	0	3>3	5mal: op +; 2mal: op *	send: >2, >6
5>4	0	0	1>4	4>4	5mal: op +; 2mal: op *	send: >4, >1
6>5	4>5	0	2>5	5>5	5mal: op +; 2mal: op *	send: >4, >6, >2
0	5>6	0	3>6	6>6	5mal: op +; 2mal: op *	send: >5, >3

WO 2024/223668

70/181

PCT/EP2024/061258

$$\begin{pmatrix} 2 \rightarrow 1 \\ 3 \rightarrow 2 \\ 0 \\ 5 \rightarrow 4 \\ 6 \rightarrow 4 \\ 0 \end{pmatrix} \cdot \Delta t_{trans}$$

$$\begin{pmatrix} 4 \rightarrow 1 \\ 5 \rightarrow 2 \\ 6 \rightarrow 3 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot \Delta t_{trans}$$

$$\begin{pmatrix} 0 \\ 1 \rightarrow 2 \\ 2 \rightarrow 3 \\ 0 \\ 4 \rightarrow 5 \\ 5 \rightarrow 6 \end{pmatrix} \cdot \Delta t_{trans}$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \rightarrow 4 \\ 2 \rightarrow 5 \\ 3 \rightarrow 6 \end{pmatrix} \cdot \Delta t_{trans}$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot \Delta t_{trans}$$

$$\begin{pmatrix} \Gamma \\ \Gamma \\ \Gamma \\ \Gamma \\ \Gamma \\ \Gamma \end{pmatrix} \cdot \Delta t_{latency}$$

meta-values

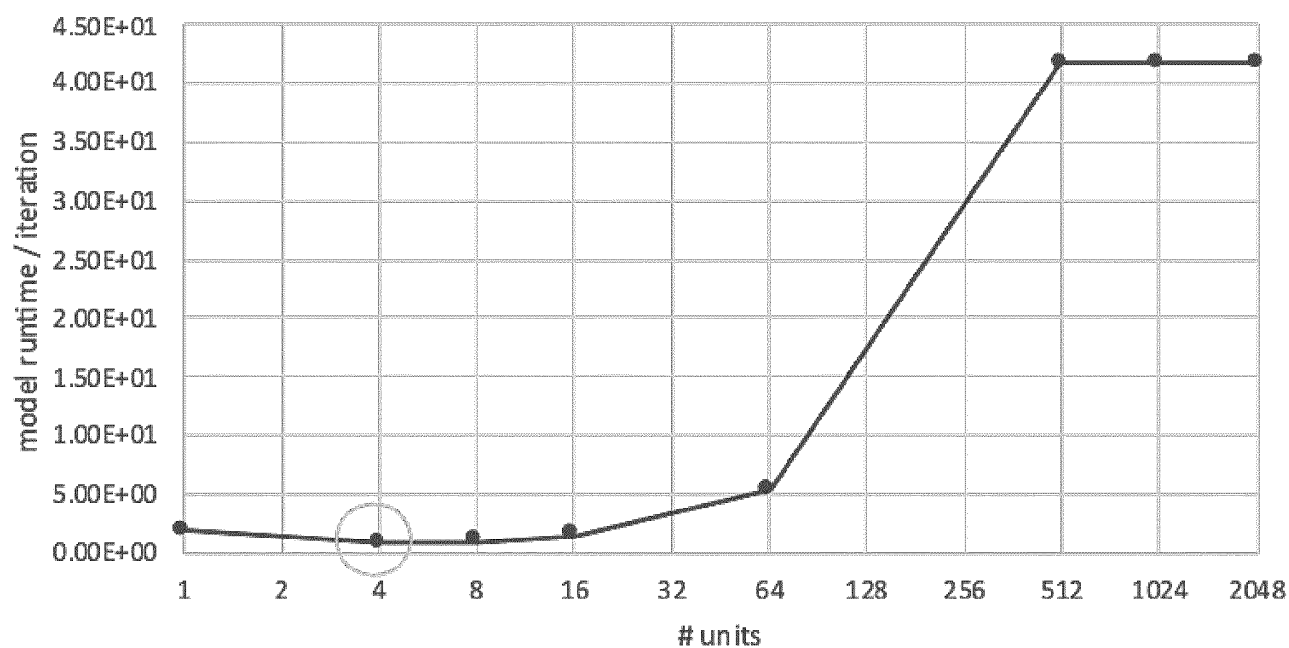
$$cache_size = \begin{pmatrix} 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ 6 \end{pmatrix} float_size$$

$$transfer_size = \begin{pmatrix} 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ 6 \end{pmatrix} float_size$$

$$\Gamma = 5 + 2 *$$

Fig. 70

Intel P5 Infiniband



Intel Haswell L3-cache

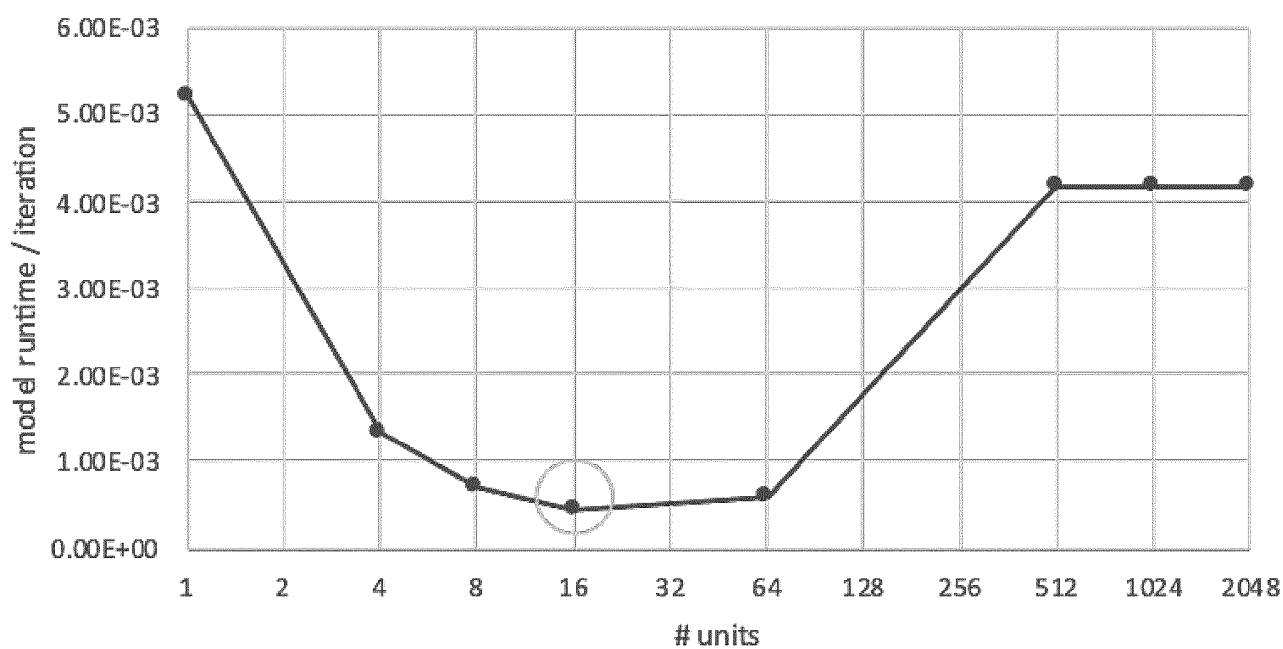


Fig. 72

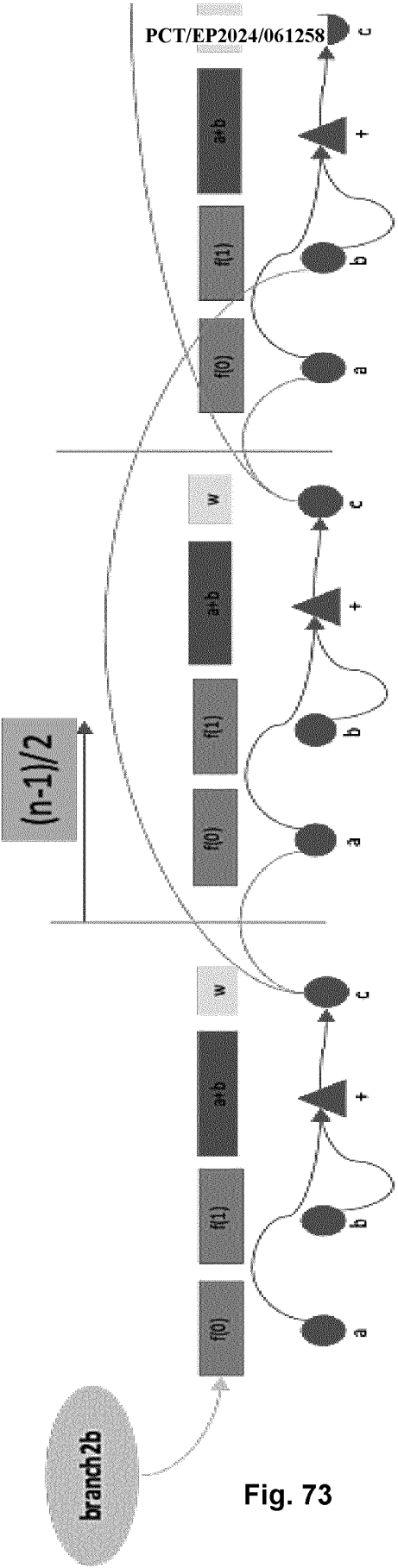
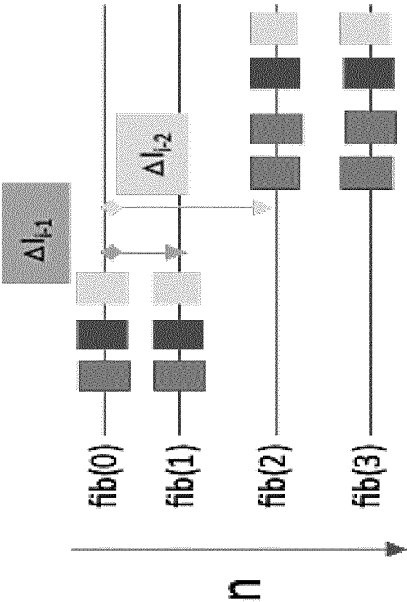
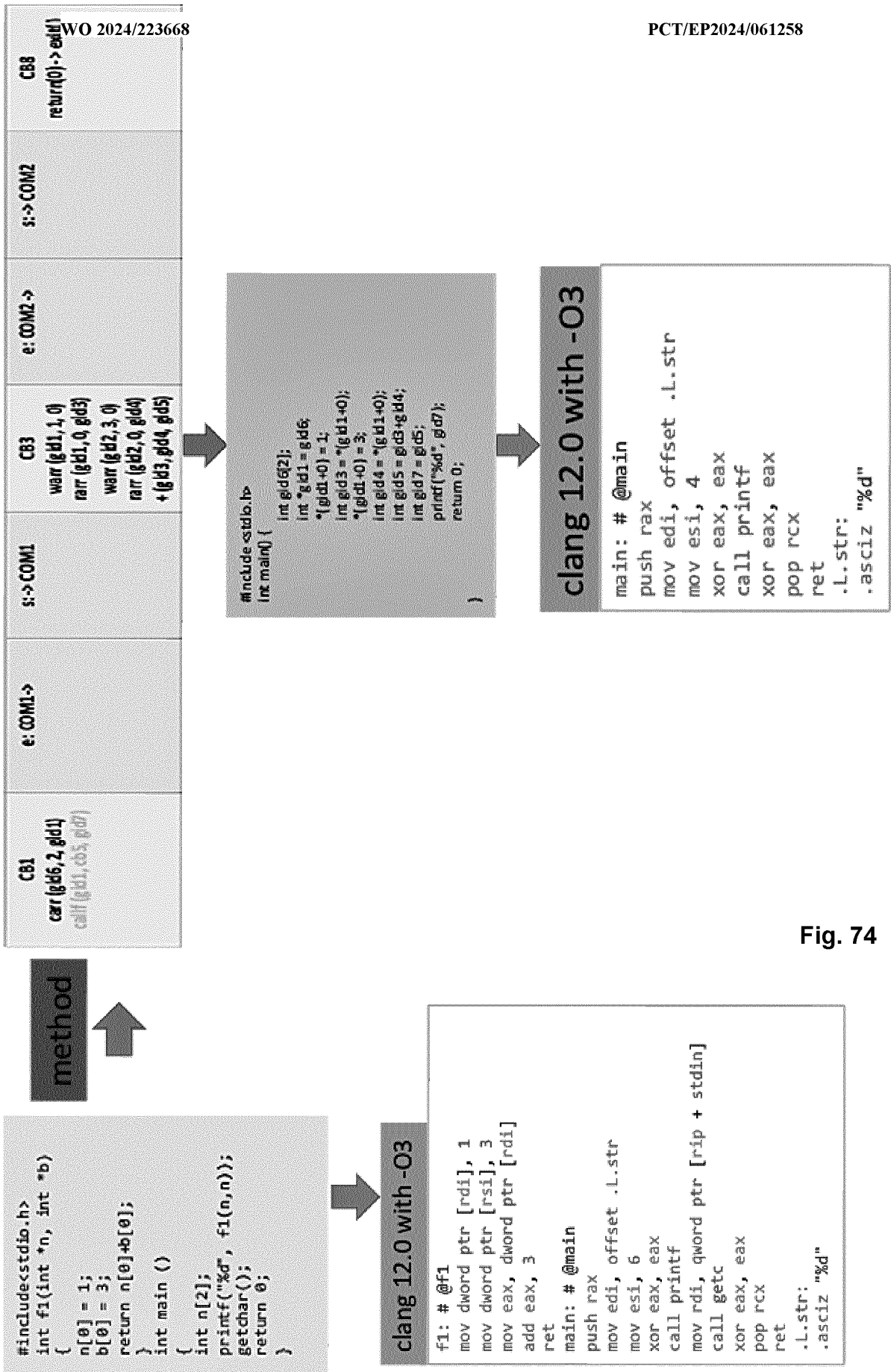


Fig. 73



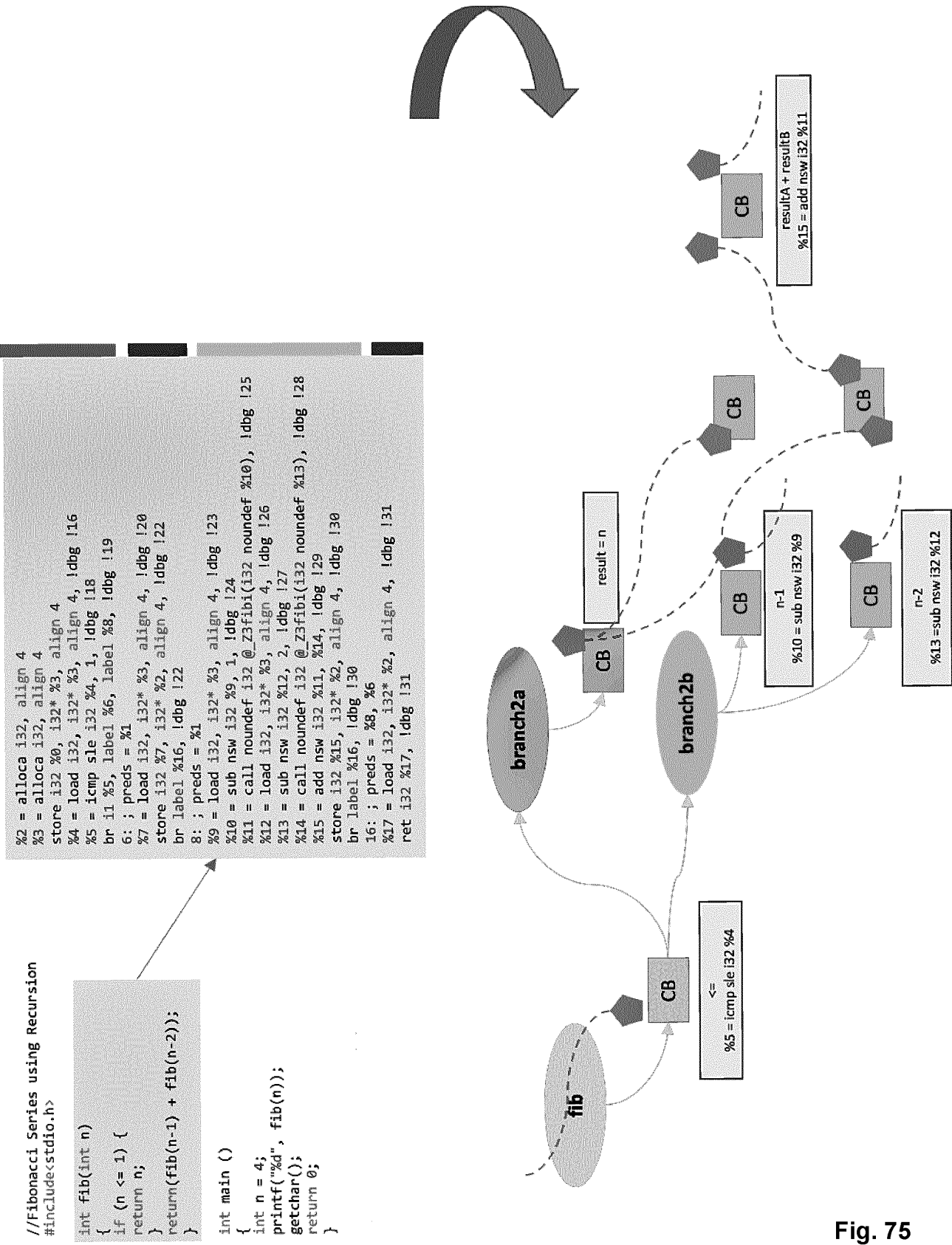


Fig. 75

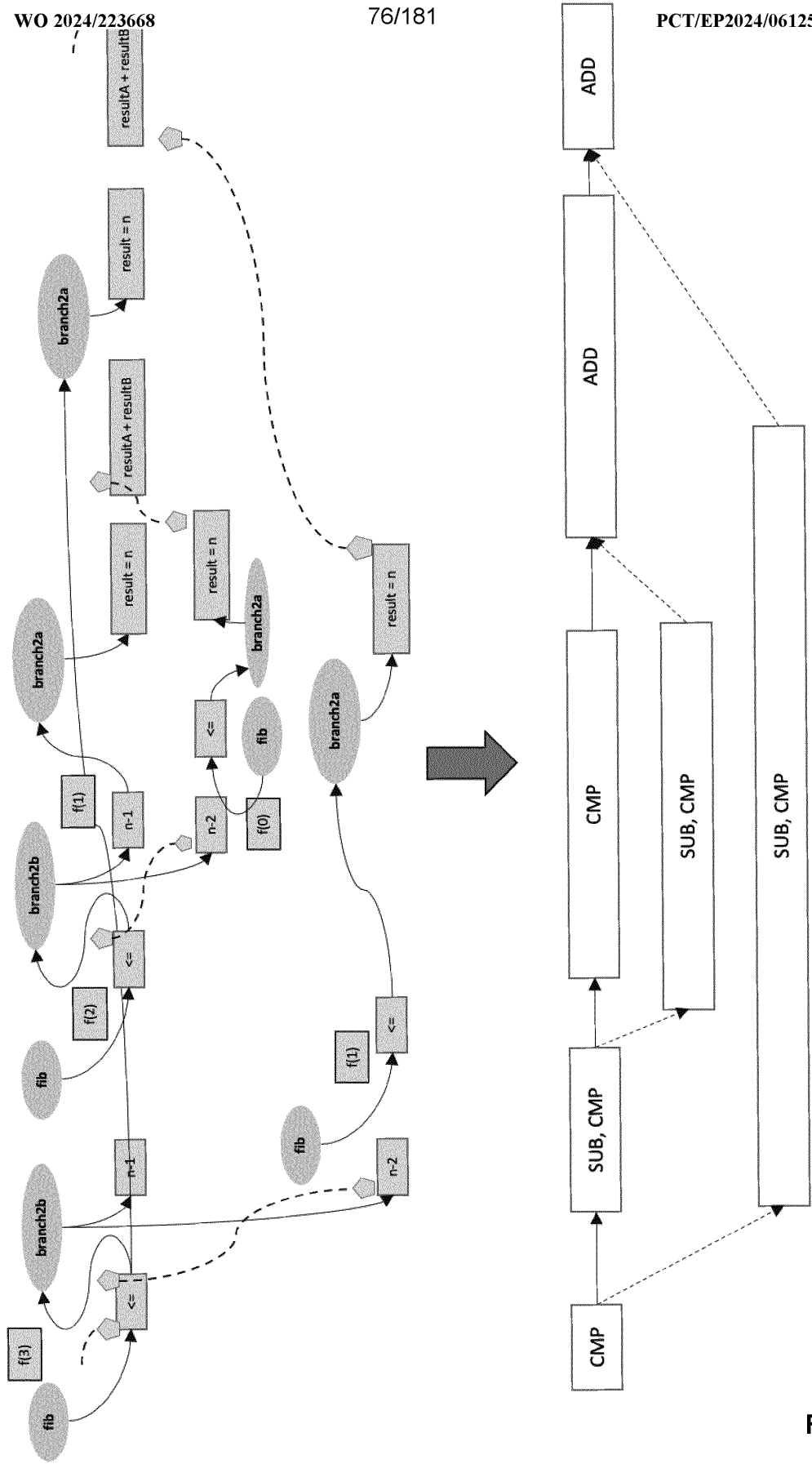


Fig. 76

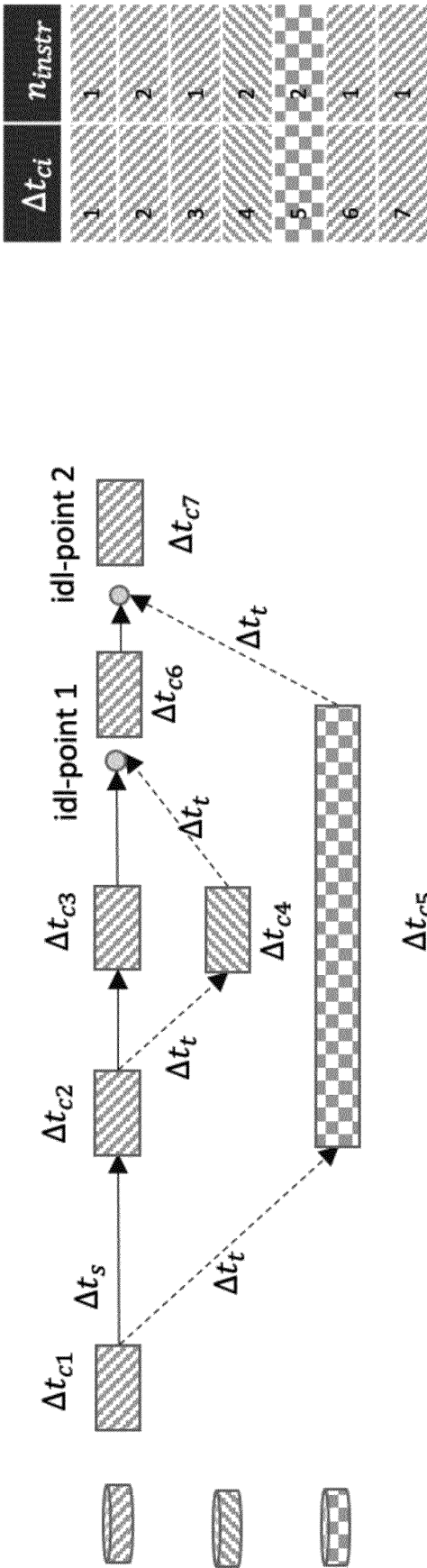
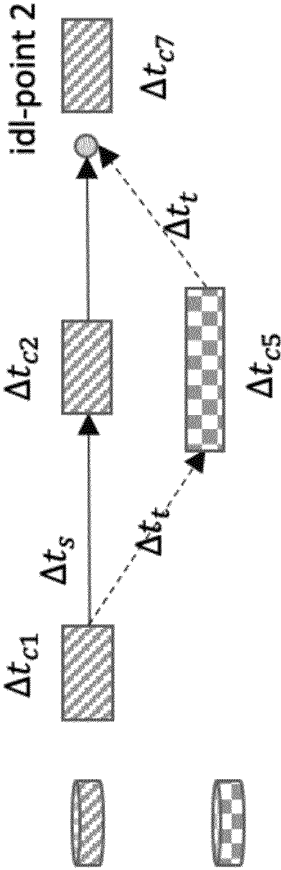
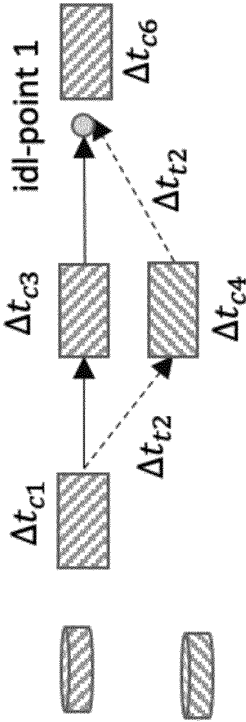


Fig. 77

Δt_{ci}	n_{instr}
1	5
3	1
4	2
6	2

Δt_{ci}	n_{instr}
1	1
2	6
5	2
7	1



c1

c2

Fig. 78



Fig. 79

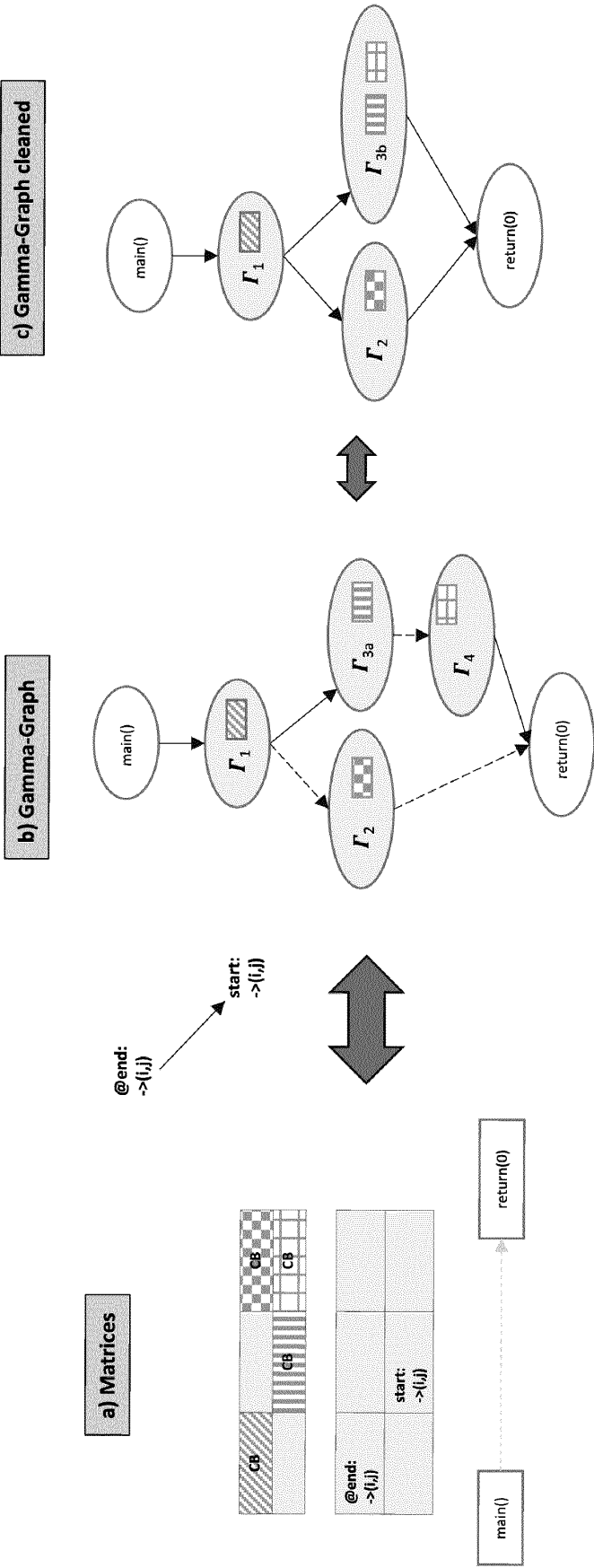


Fig. 80

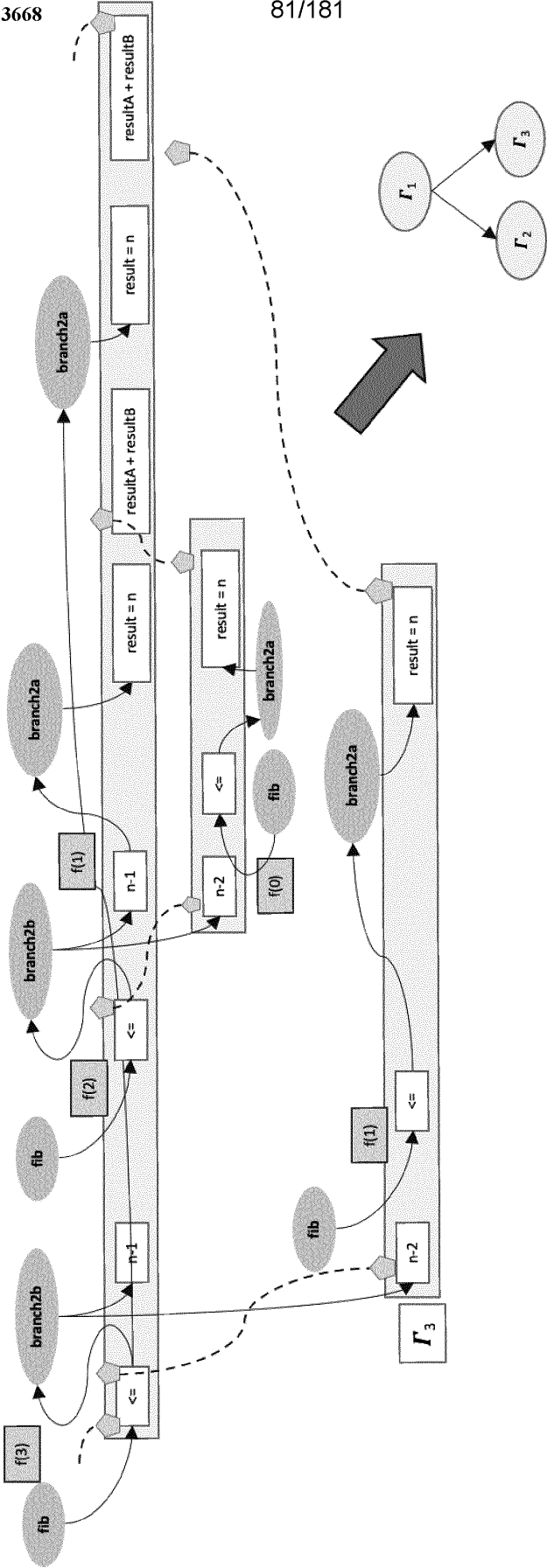


Fig. 81

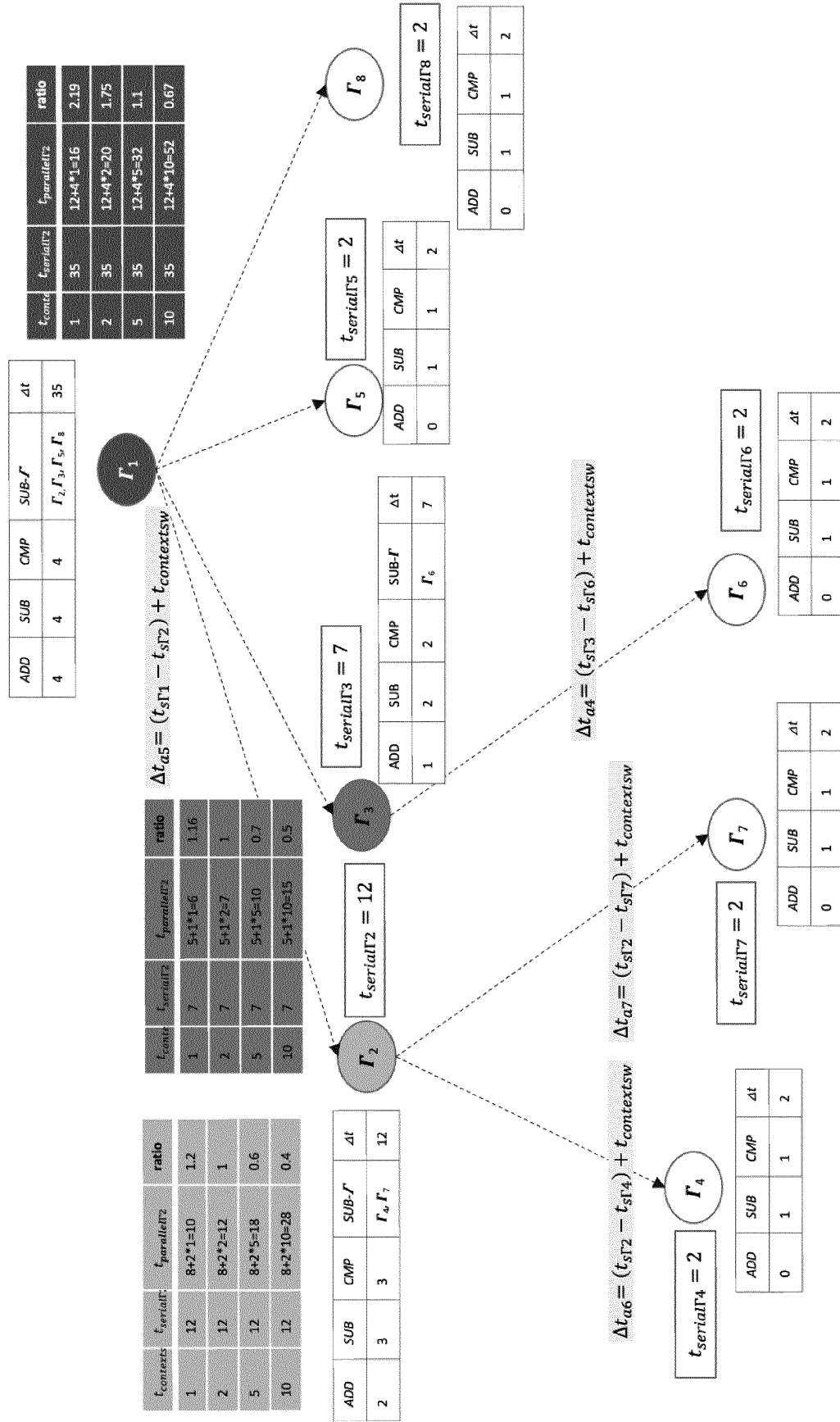


Fig. 82

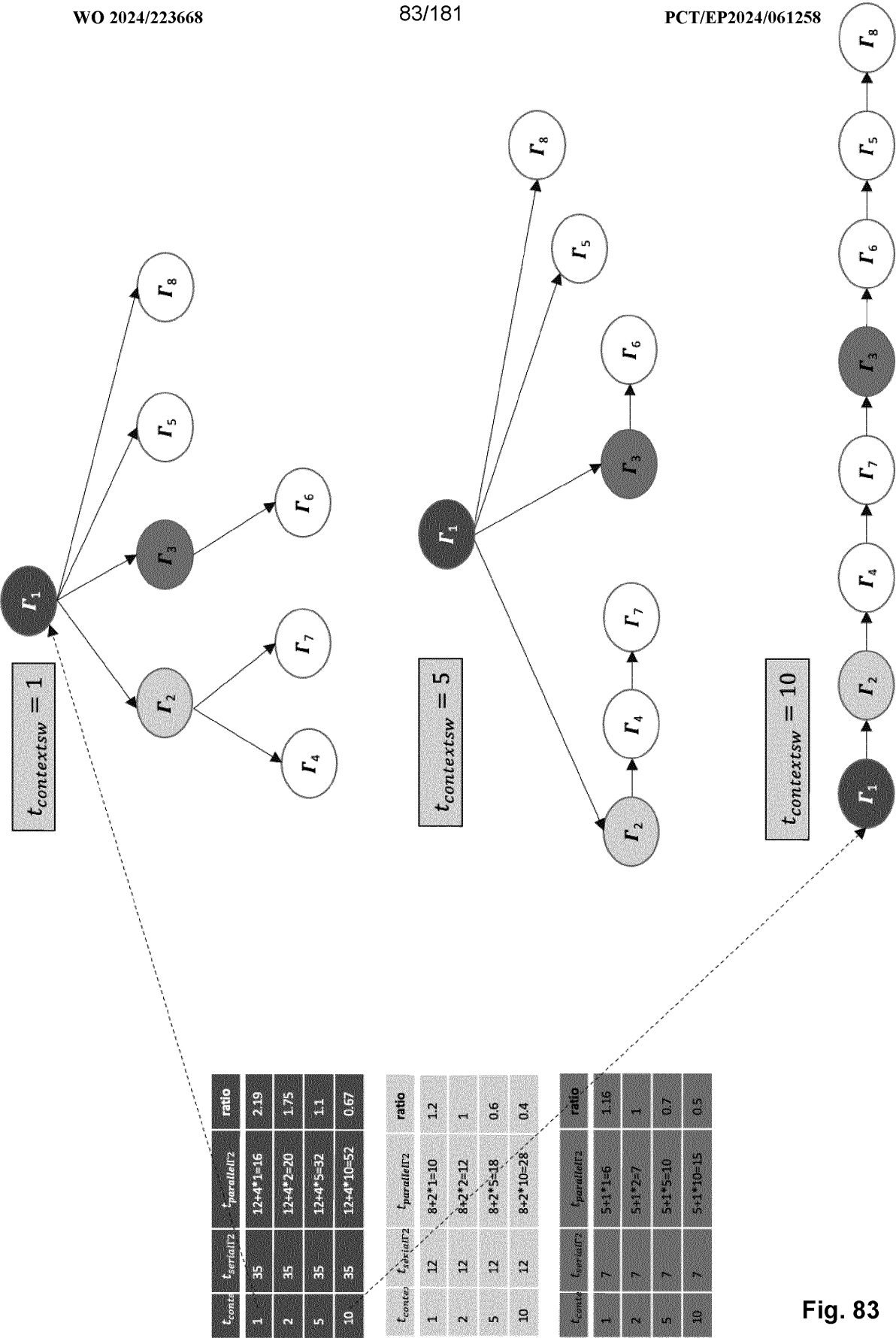


Fig. 83

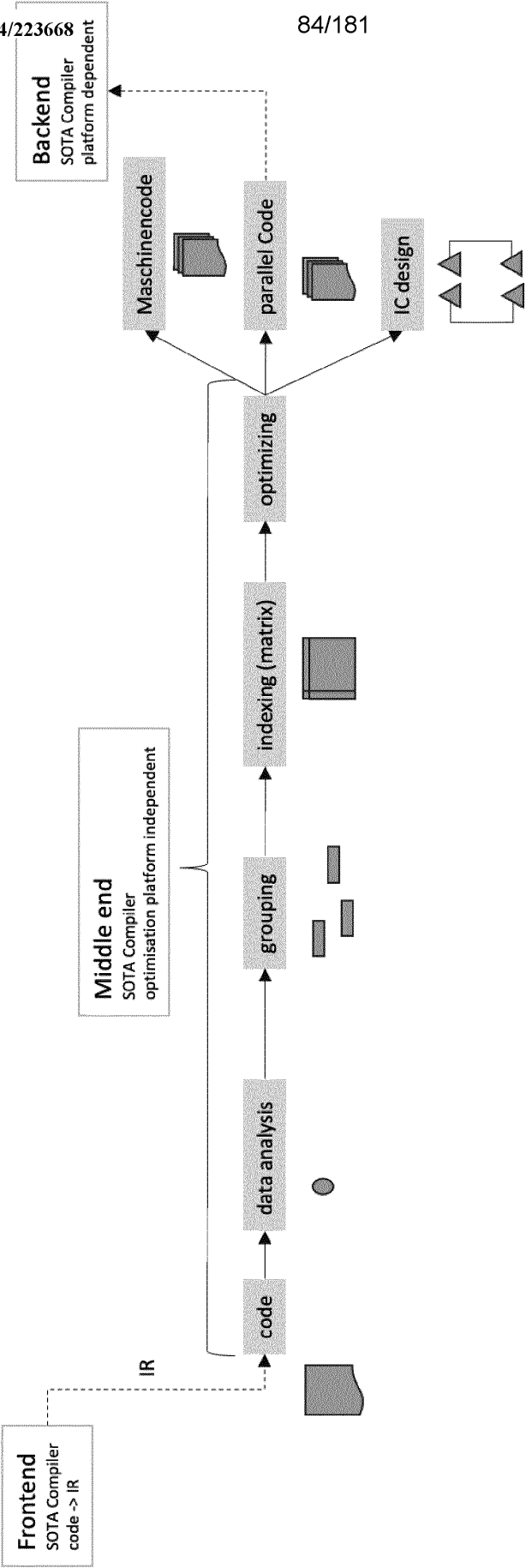


Fig. 84

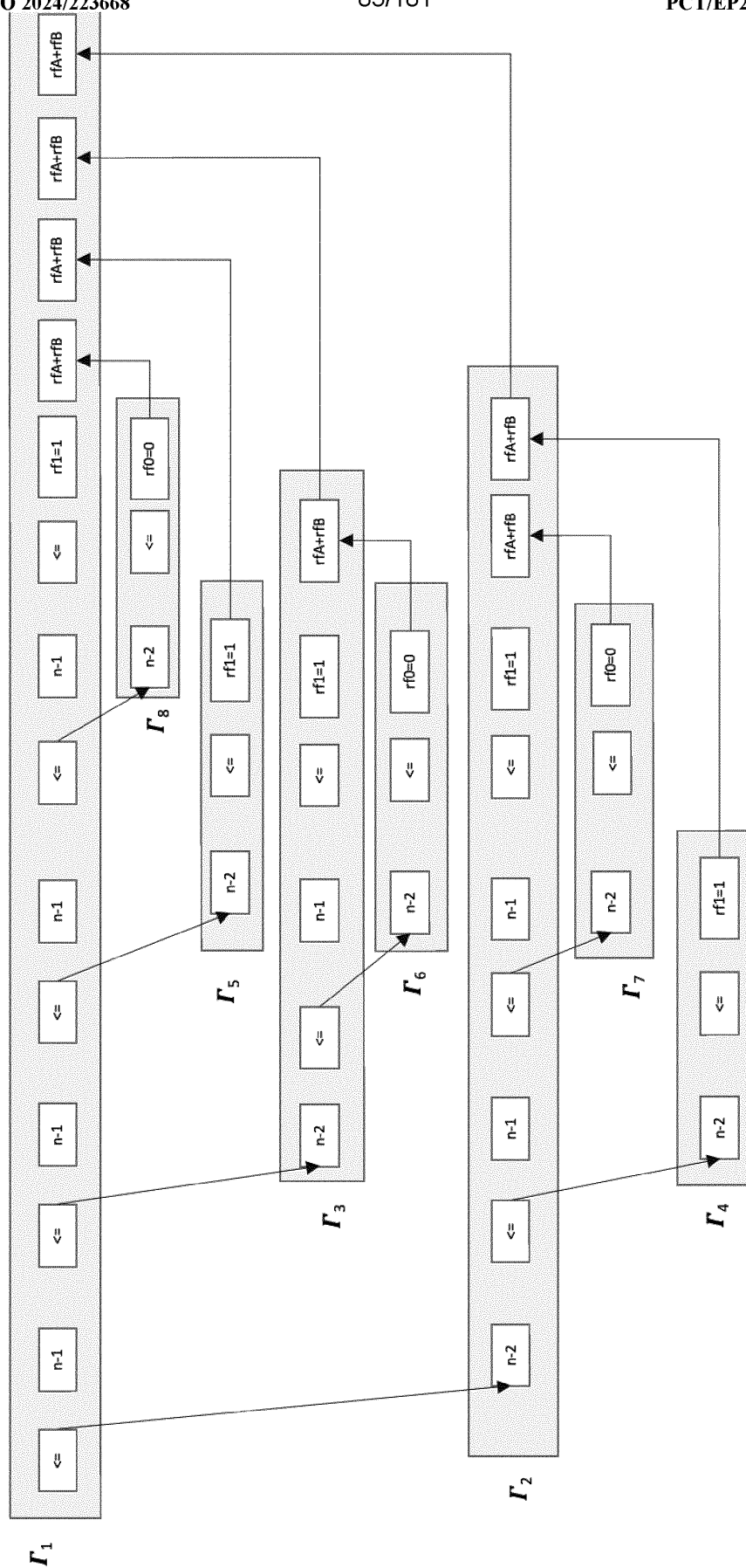
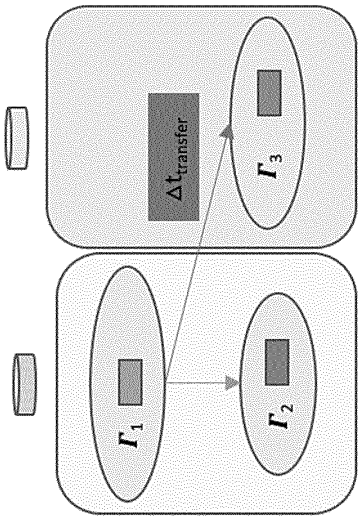
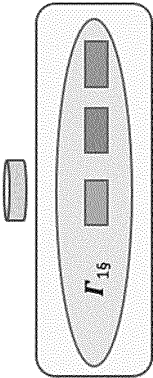


Fig. 85



$$t_{end1} = \max(\Delta t_1 + \Delta t_{transfer}, \Delta t_2) + \Delta t_3$$



$$t_{end2} = \Delta t_1$$

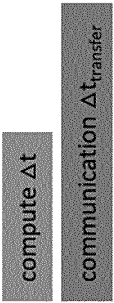
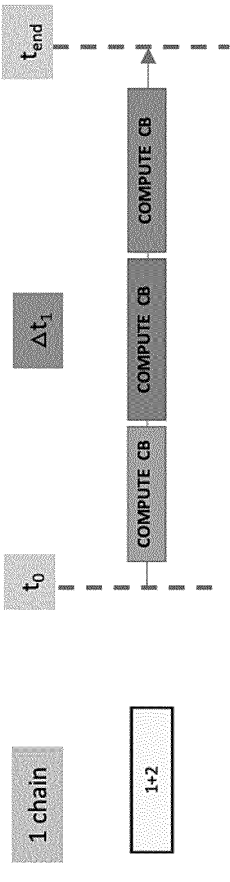
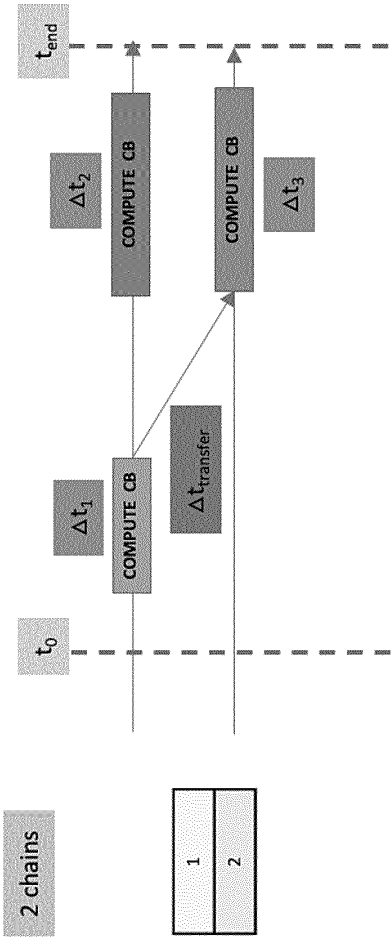


Fig. 86

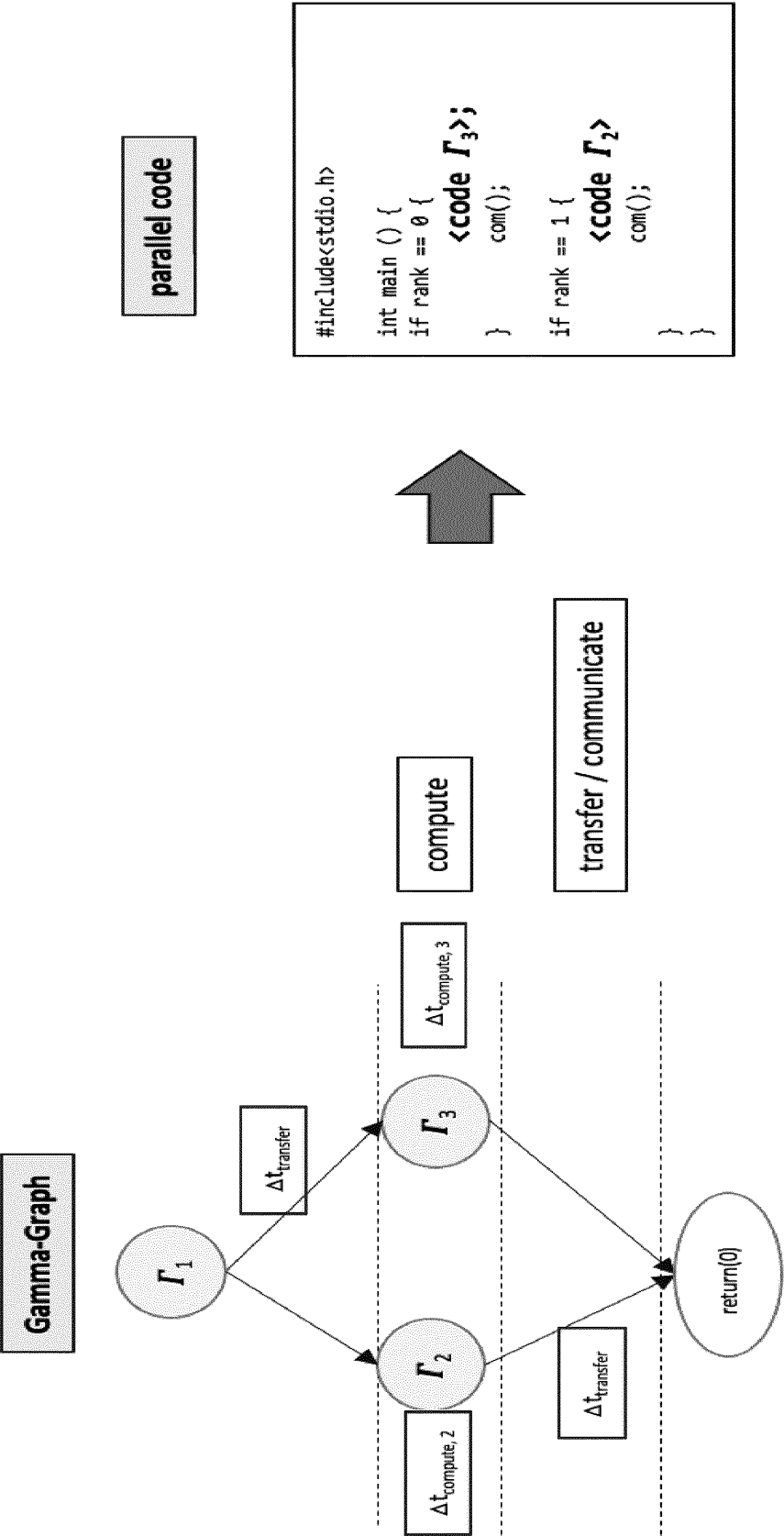


Fig. 87

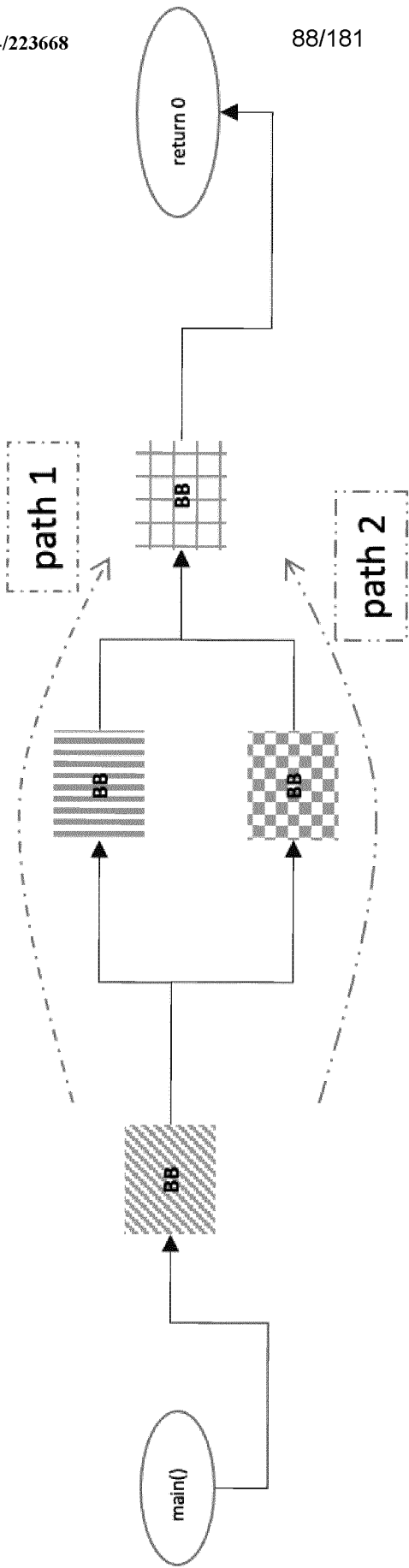


Fig. 88

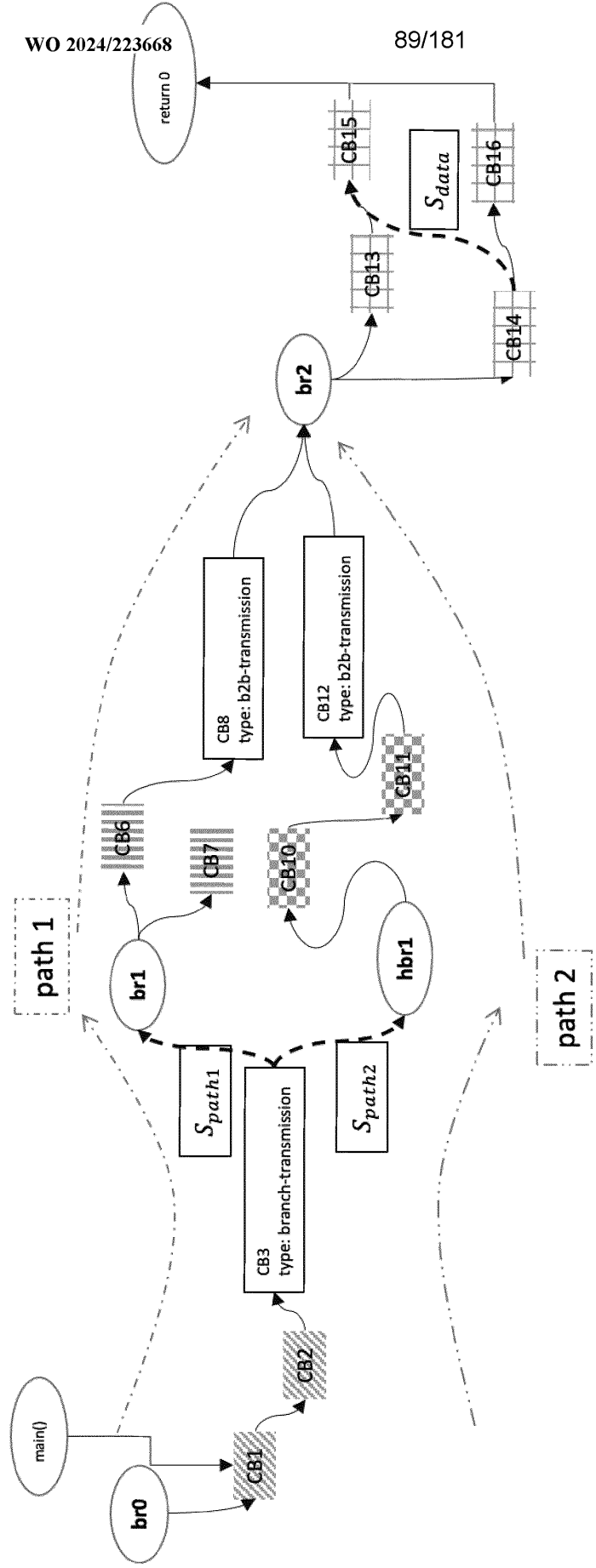


Fig. 89



Fig. 90

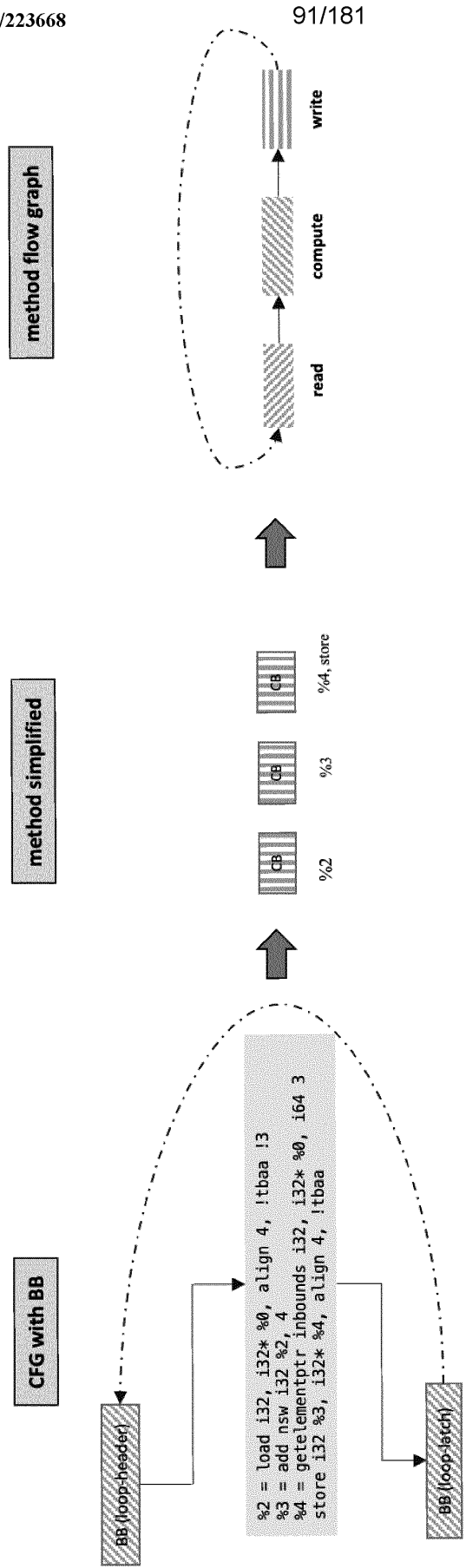


Fig. 91

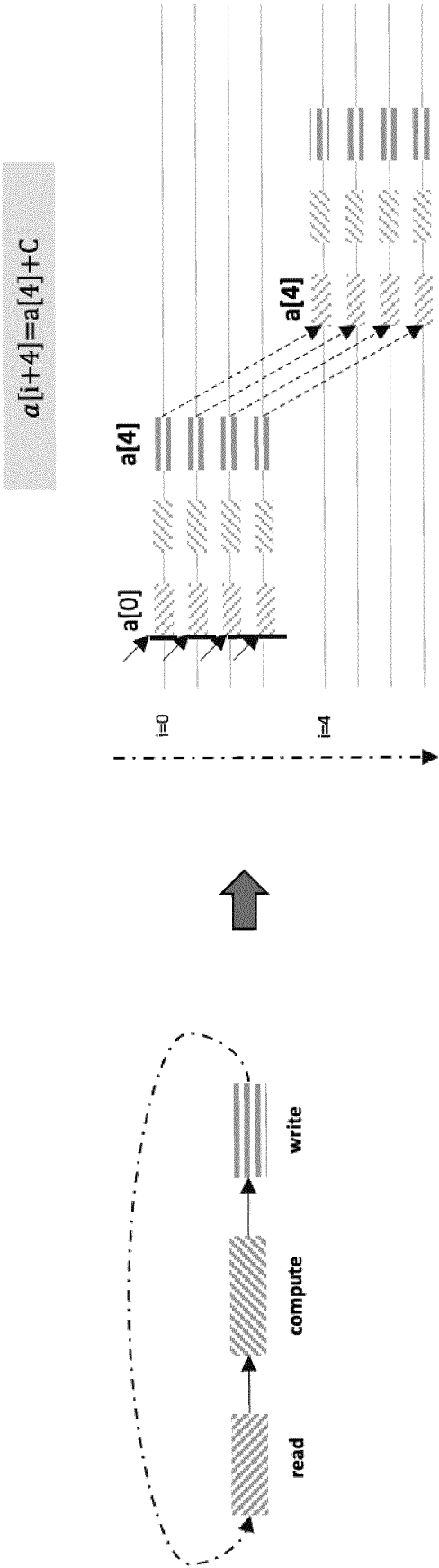


Fig. 92

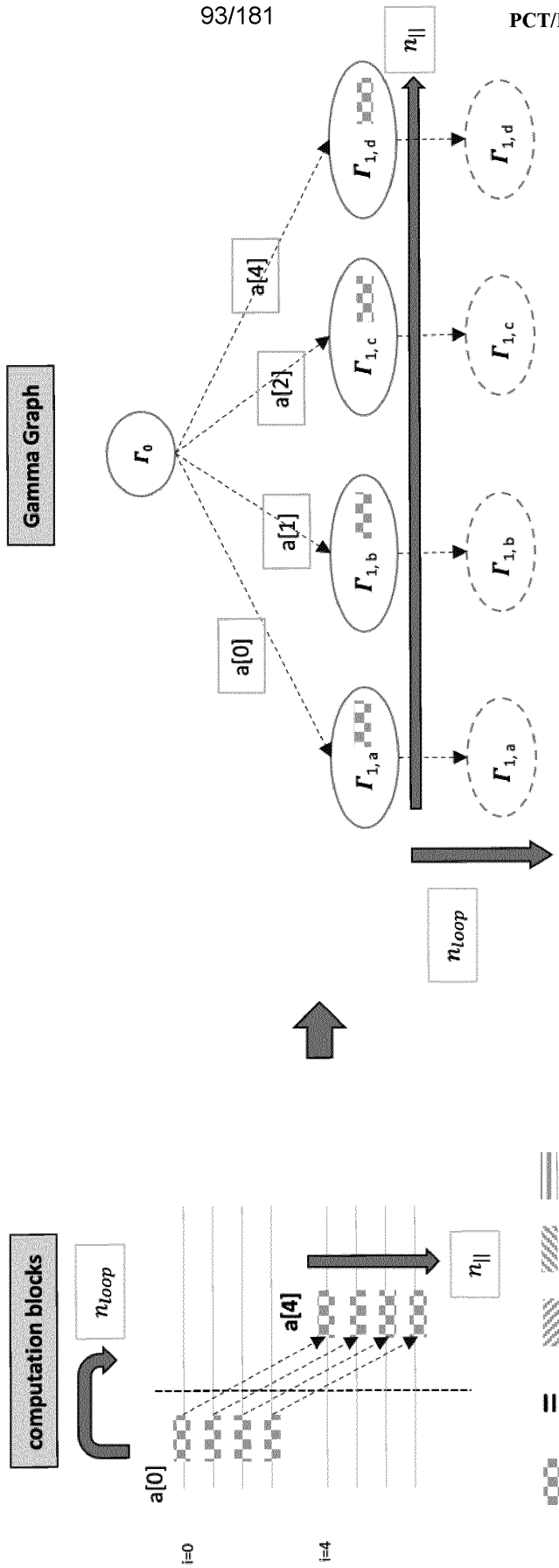


Fig. 93

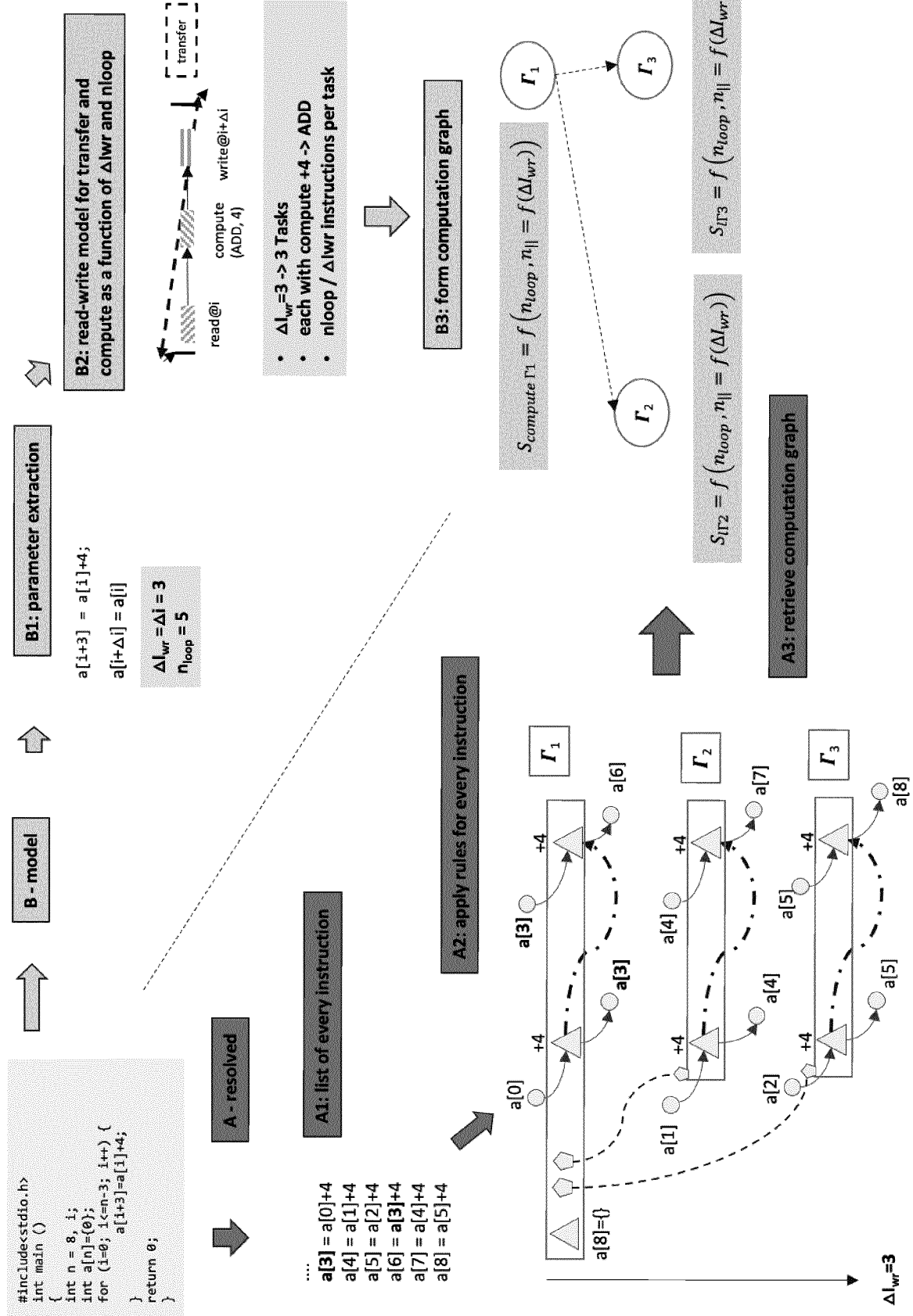


Fig. 94

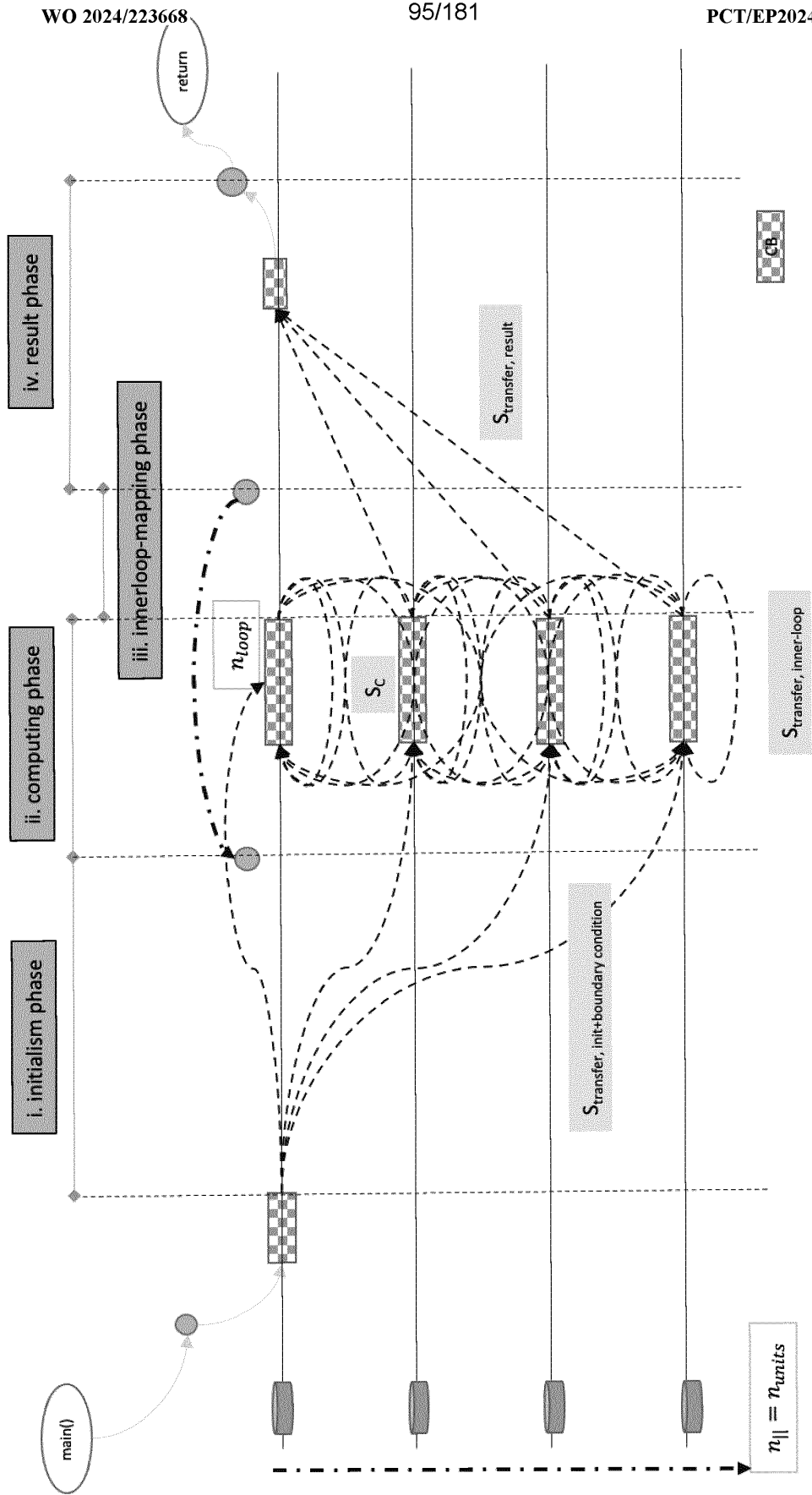


Fig. 95

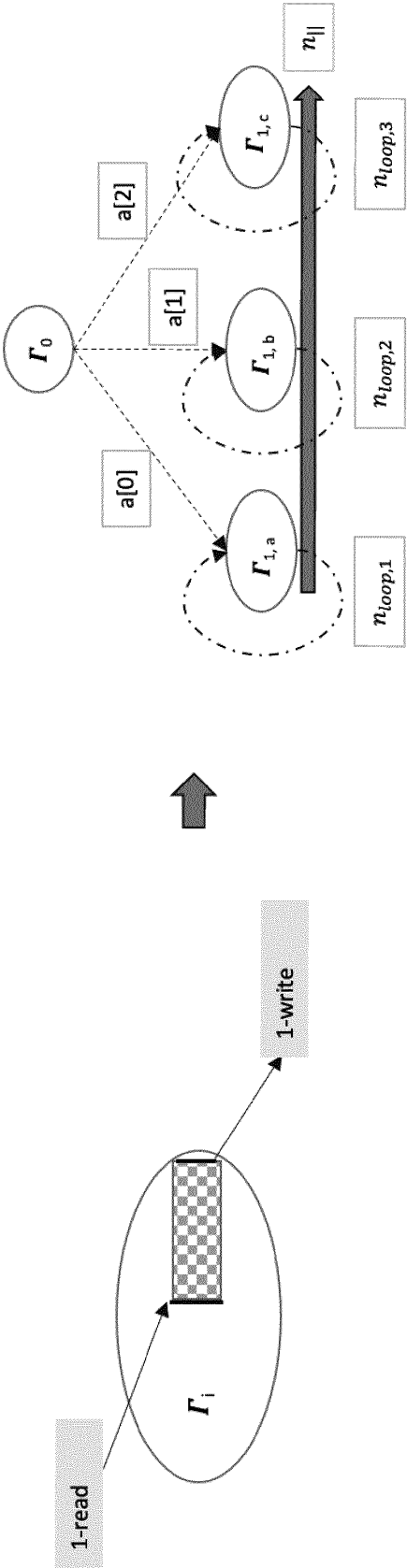


Fig. 96

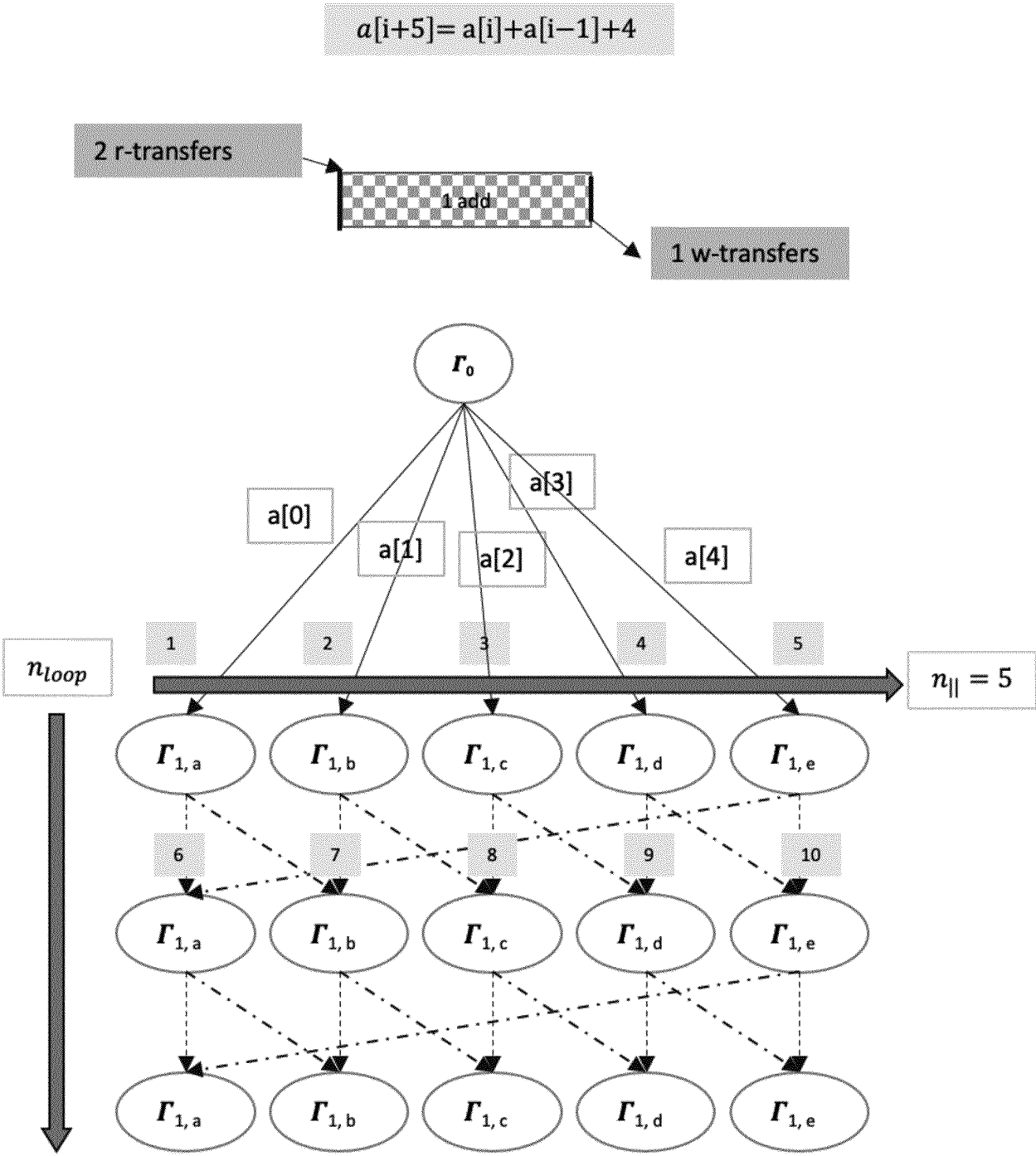


Fig. 97

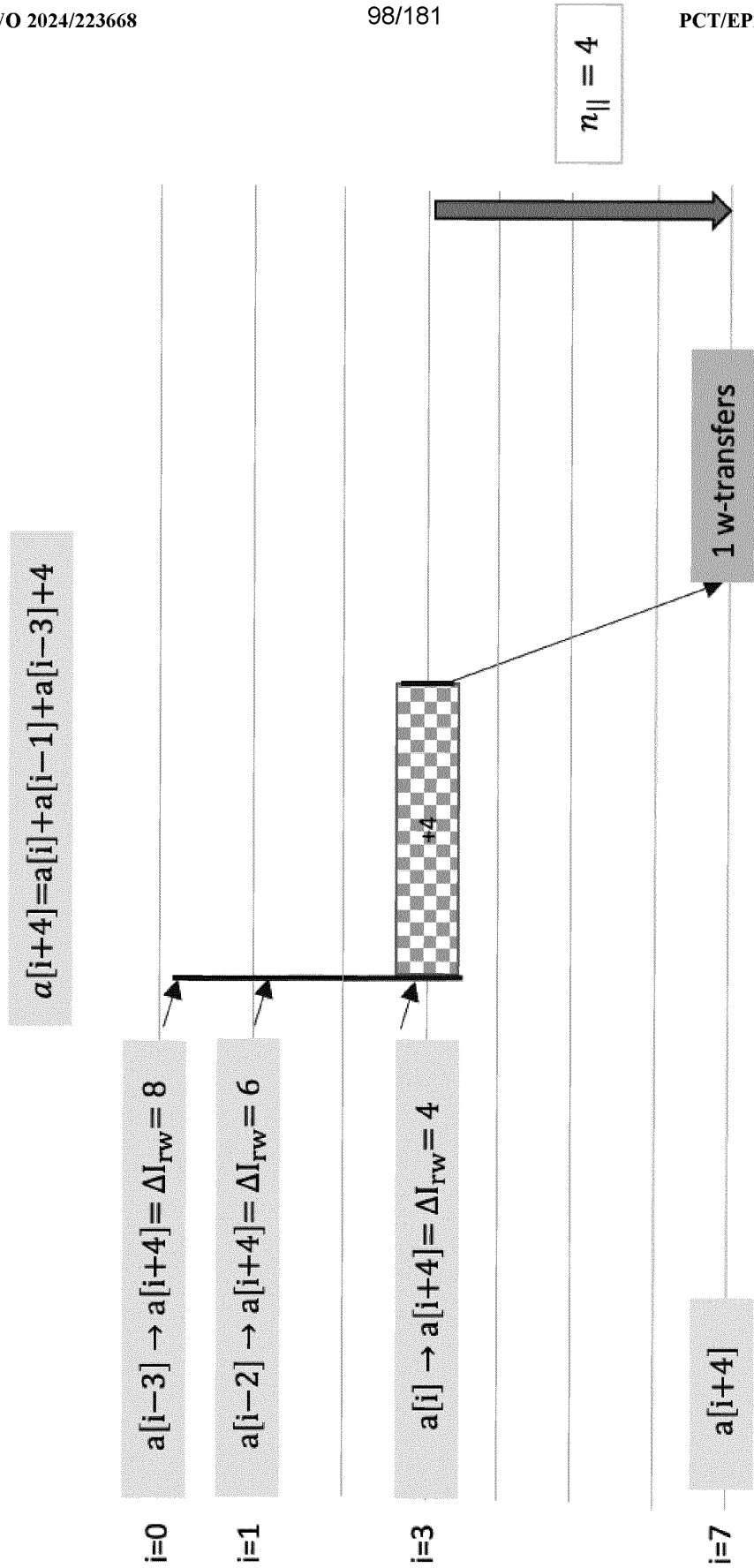


Fig. 98

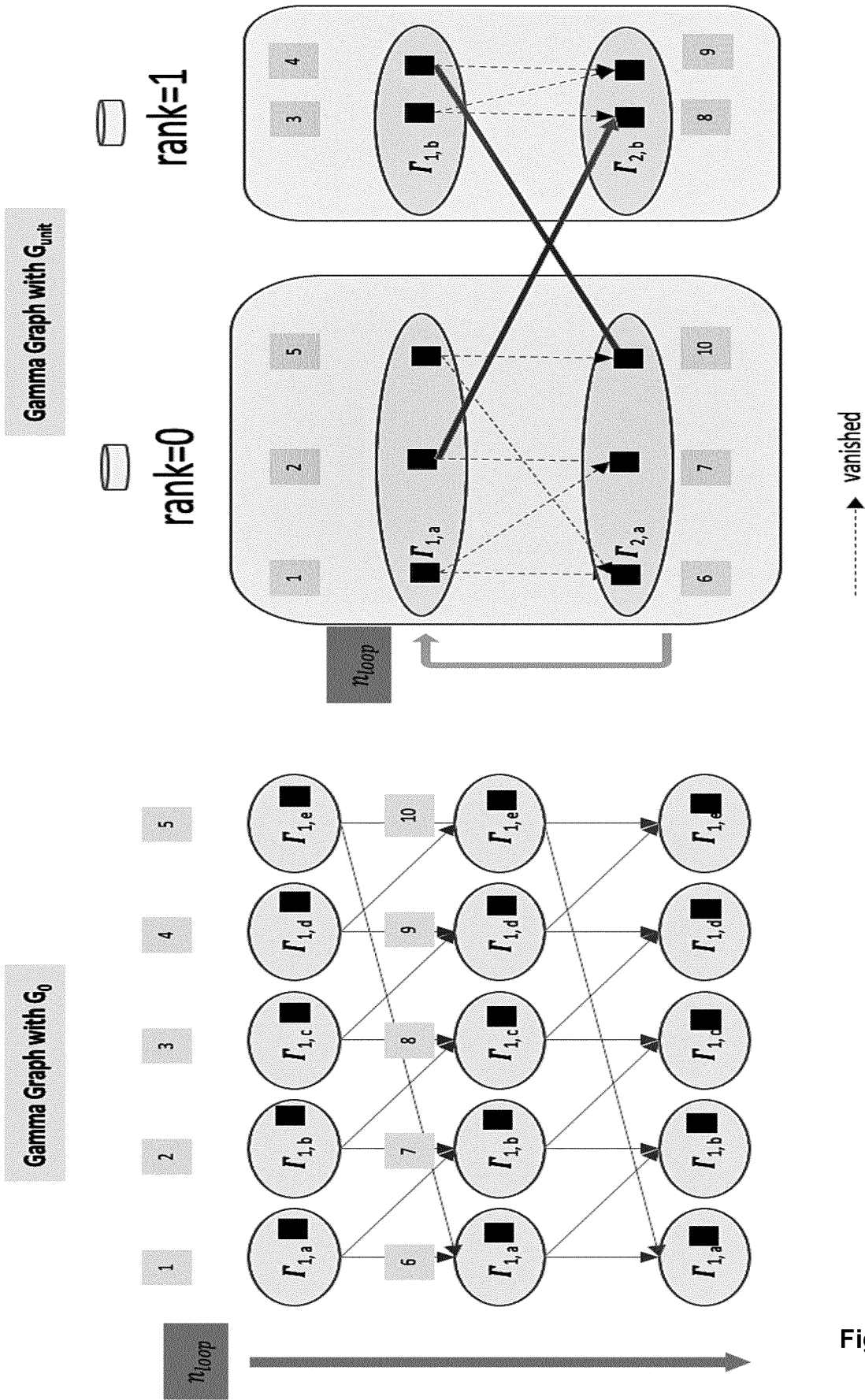


Fig. 99

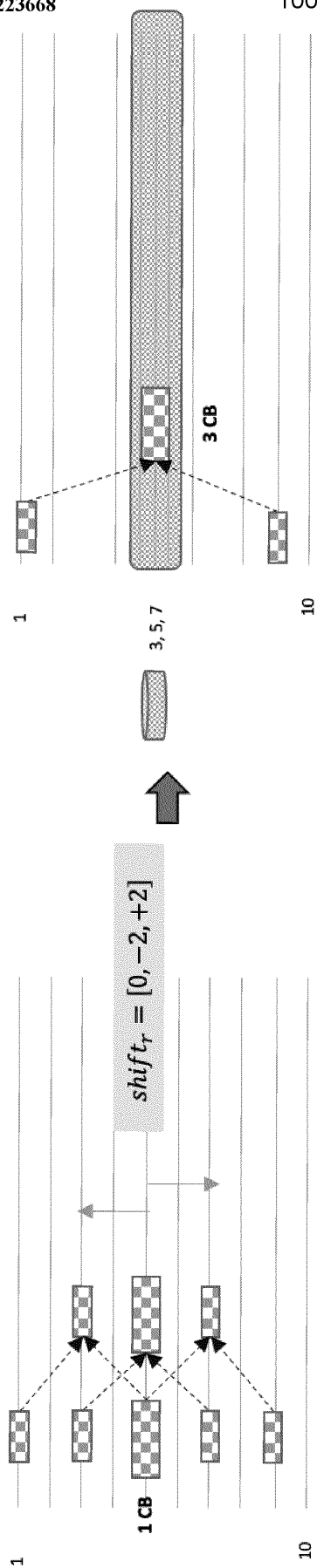


Fig. 100

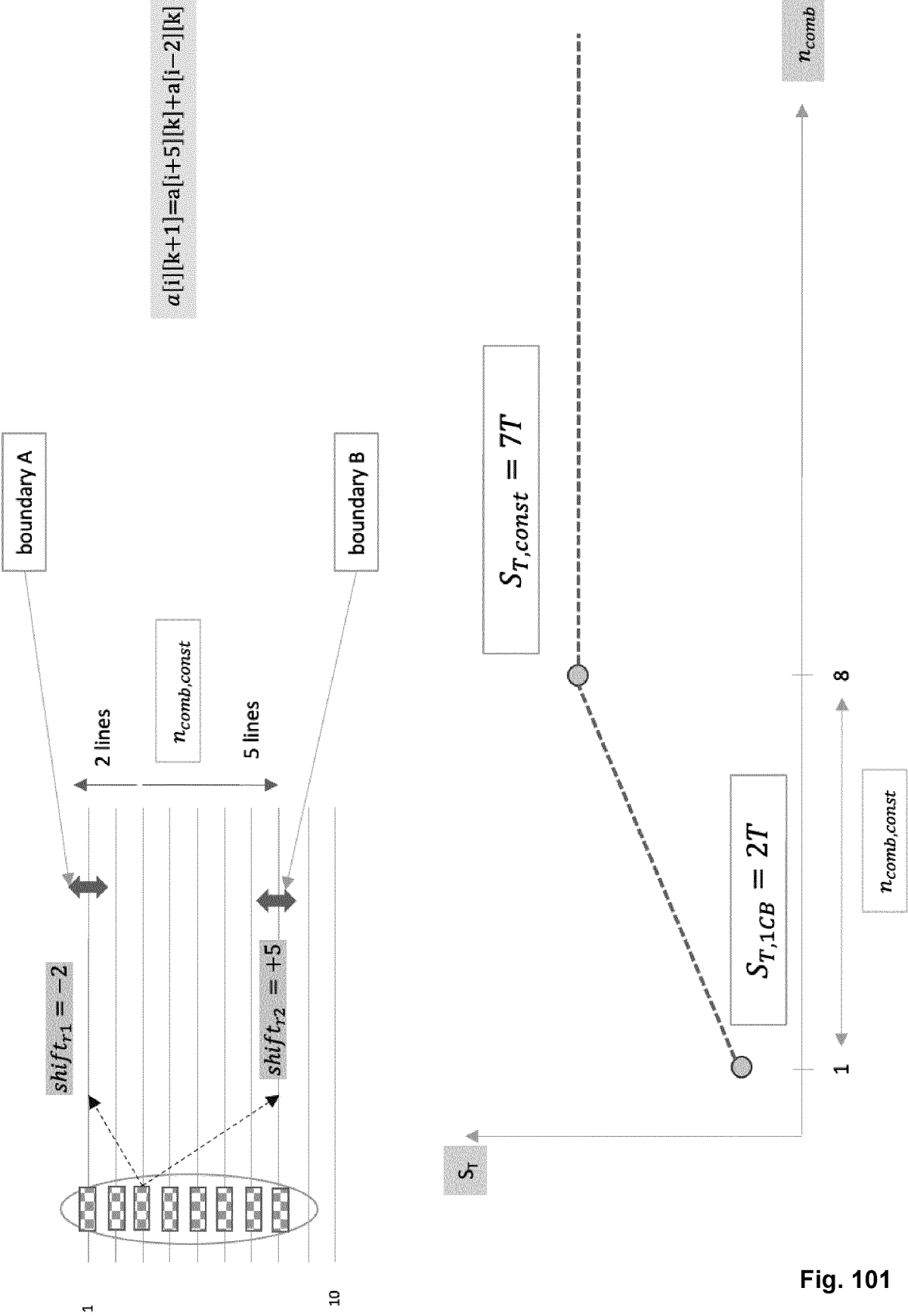


Fig. 101

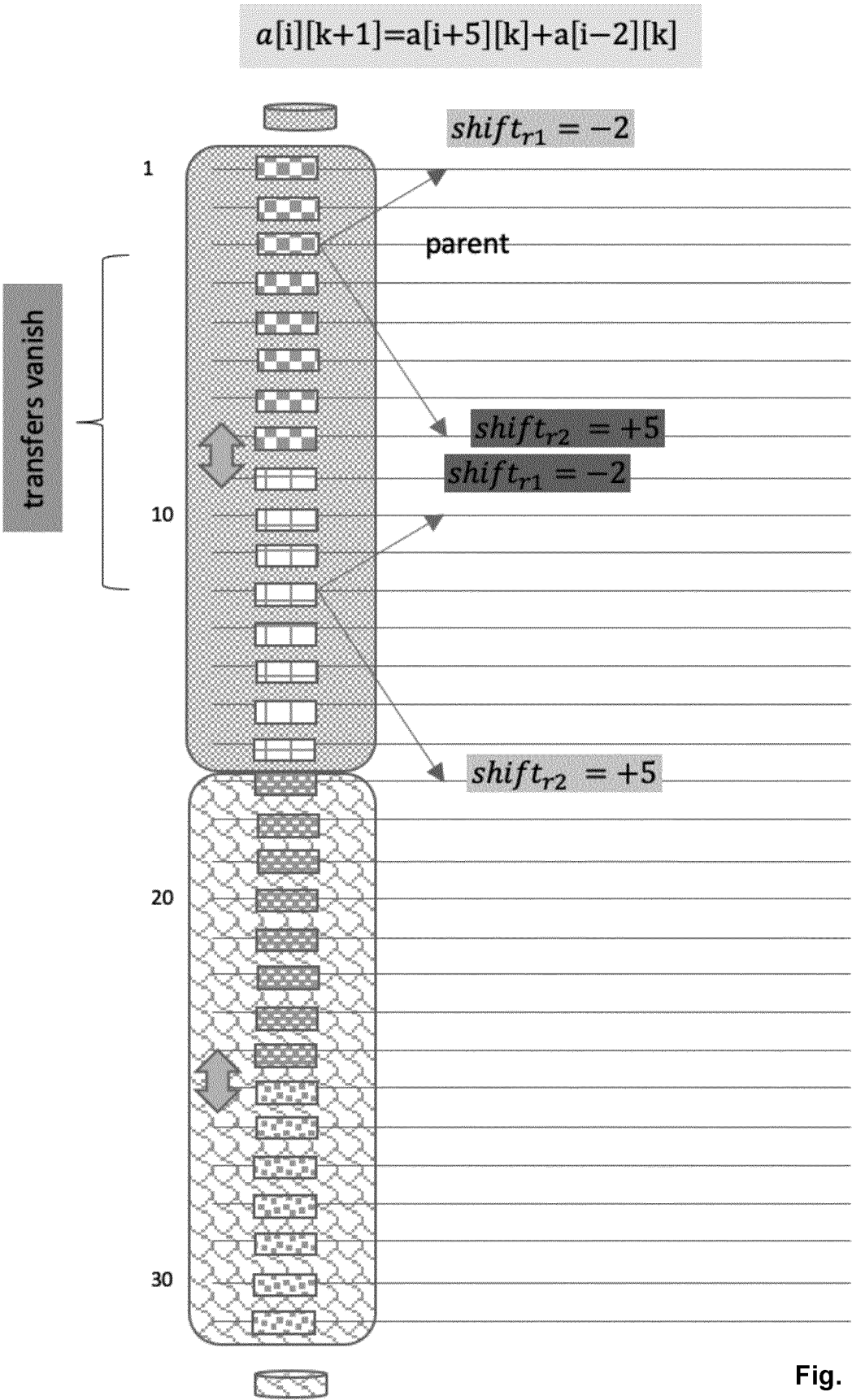


Fig. 102

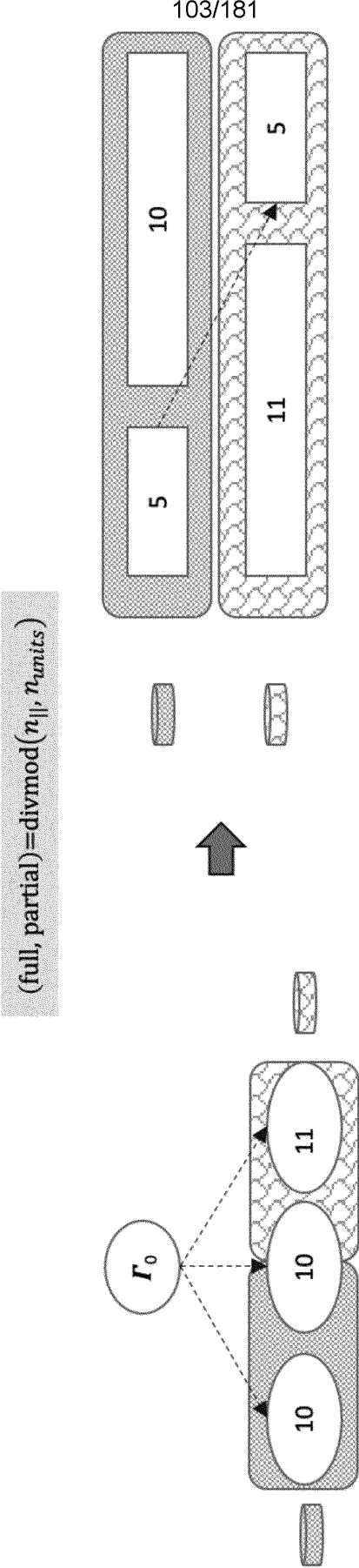


Fig. 103

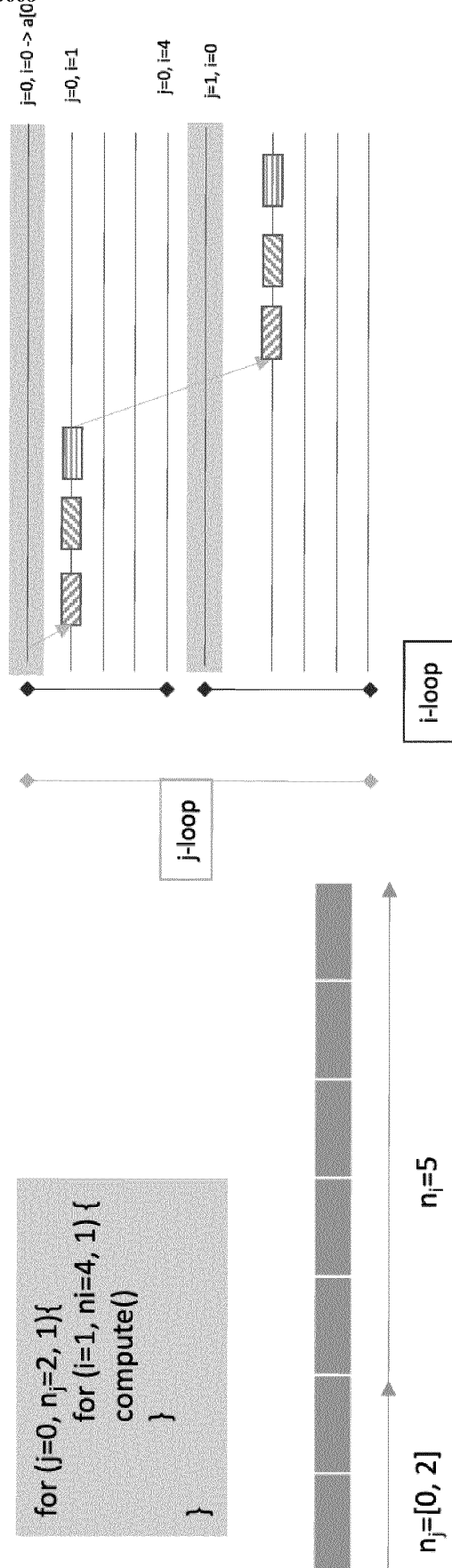


Fig. 104

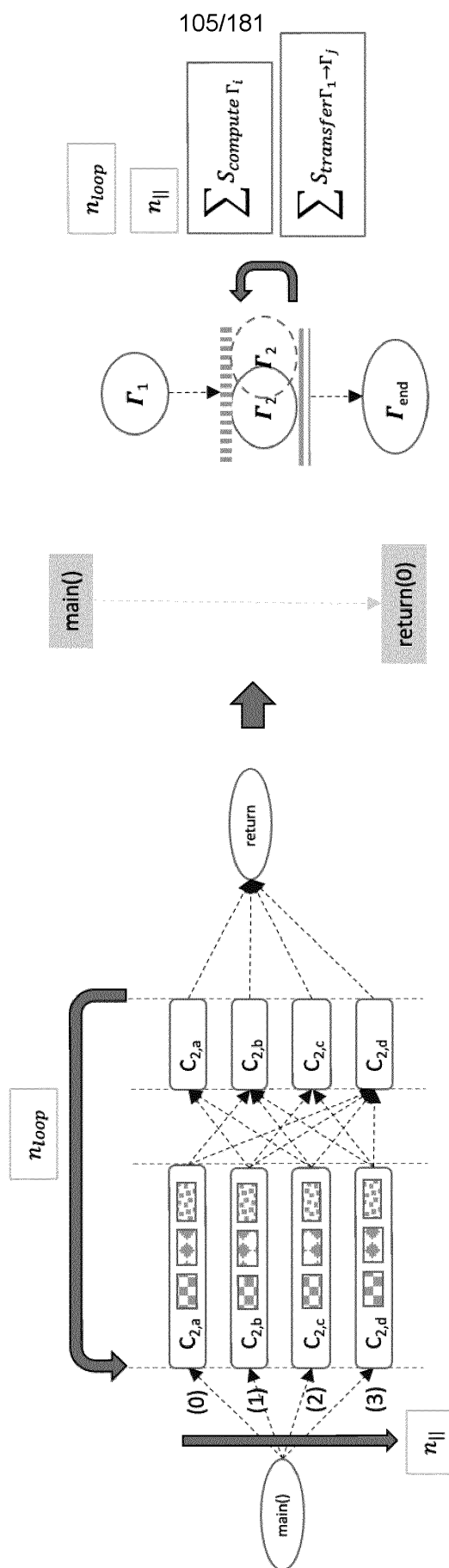


Fig. 105

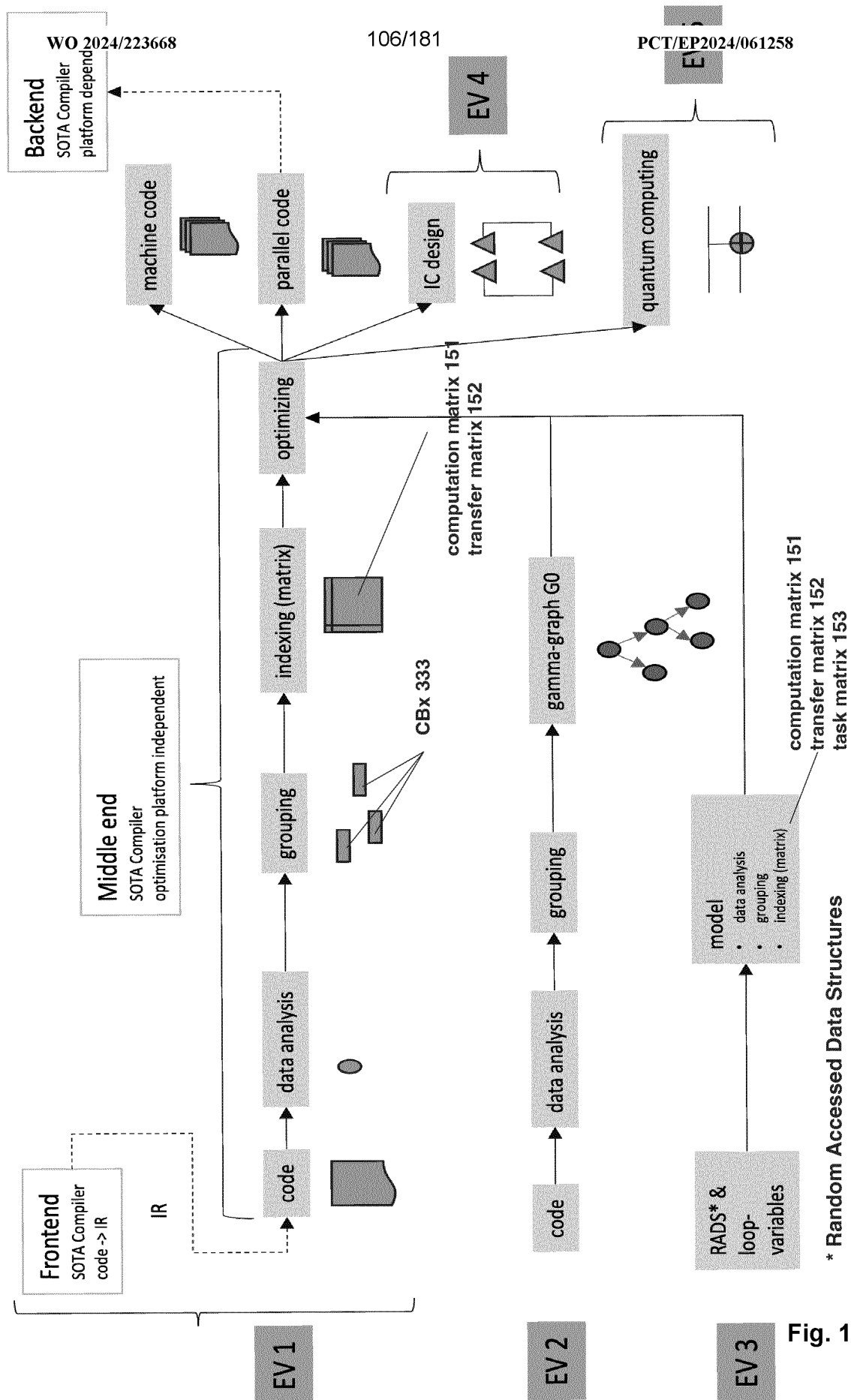
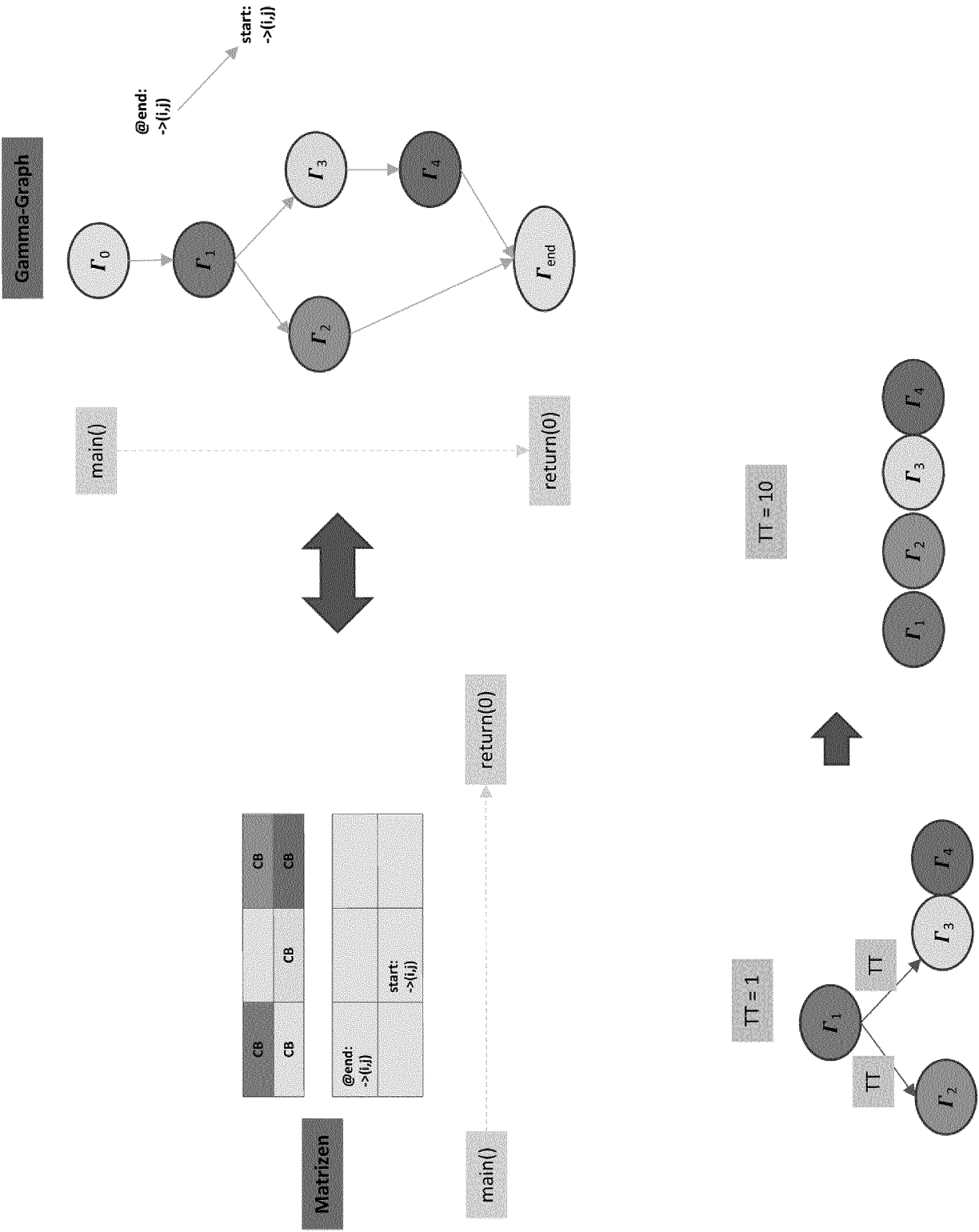


Fig. 106



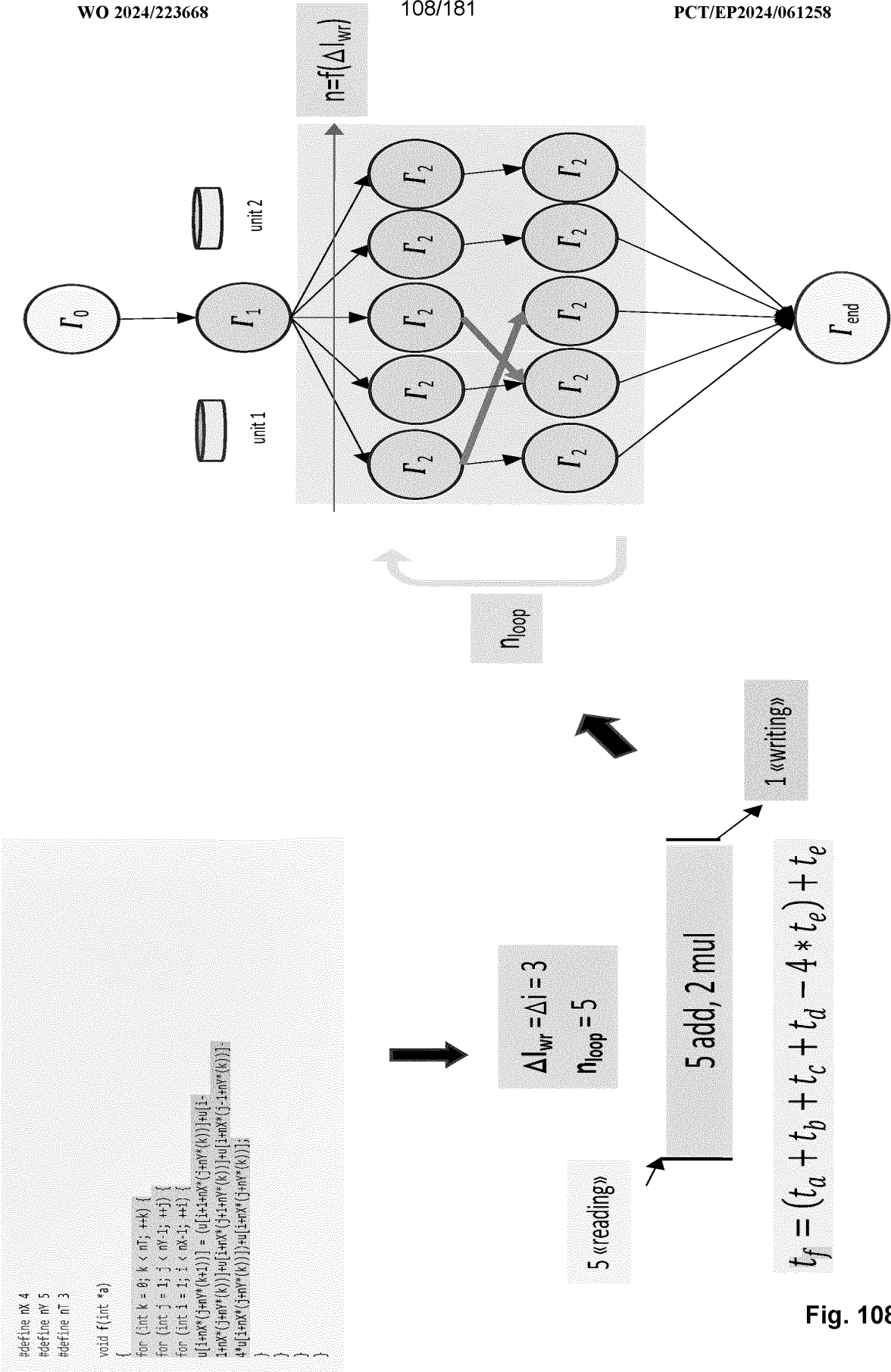


Fig. 108

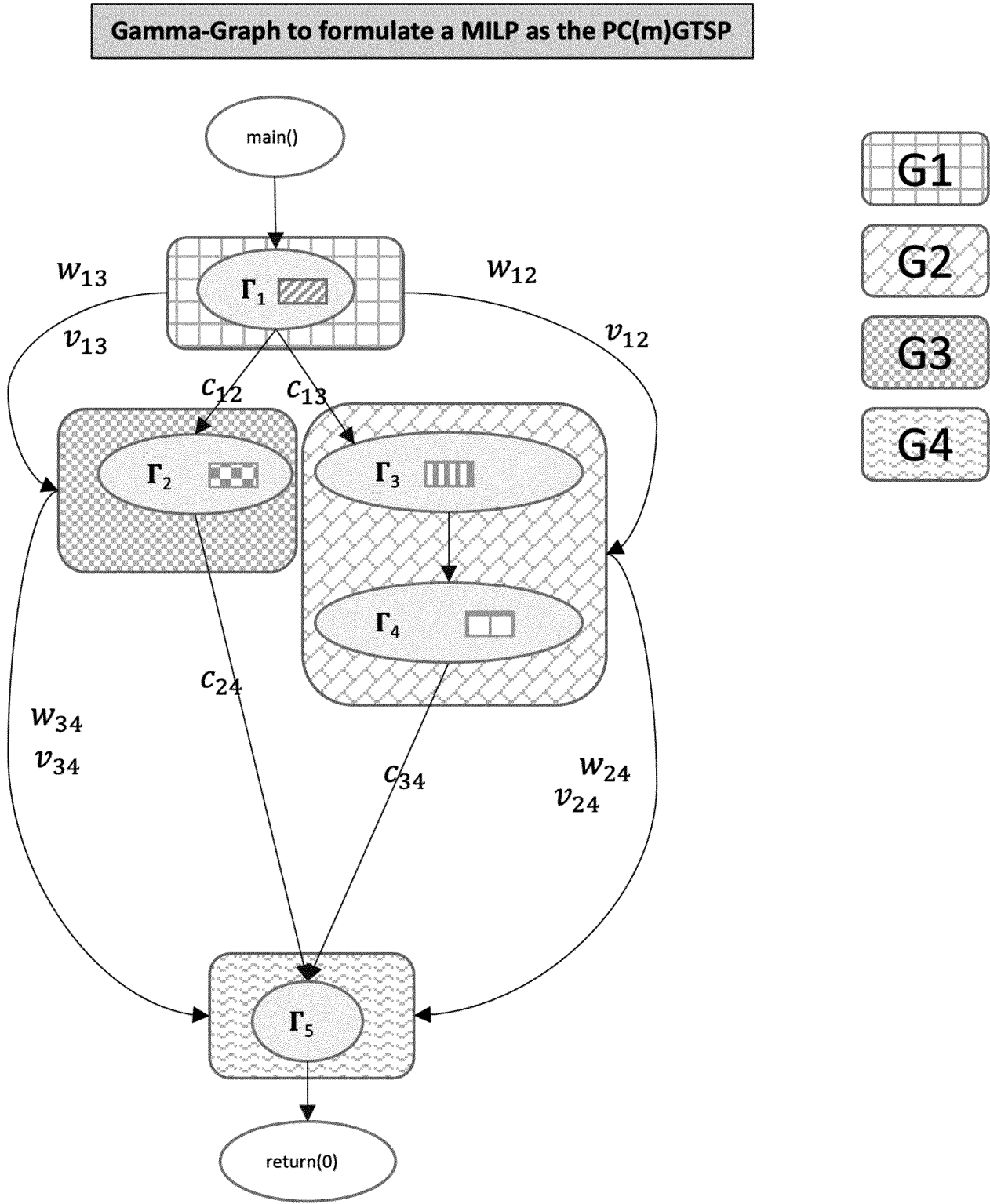


Fig. 109a

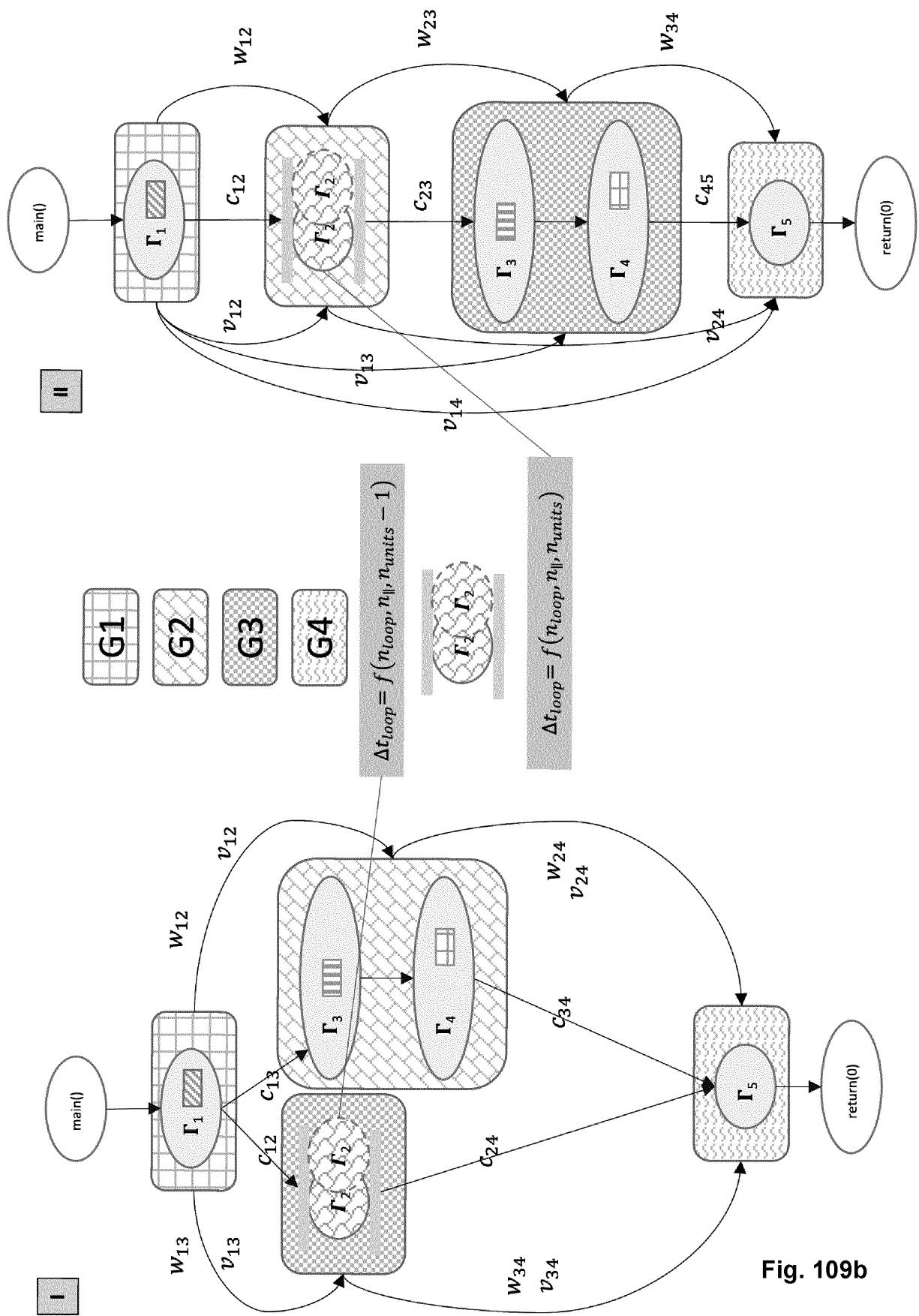


Fig. 109b

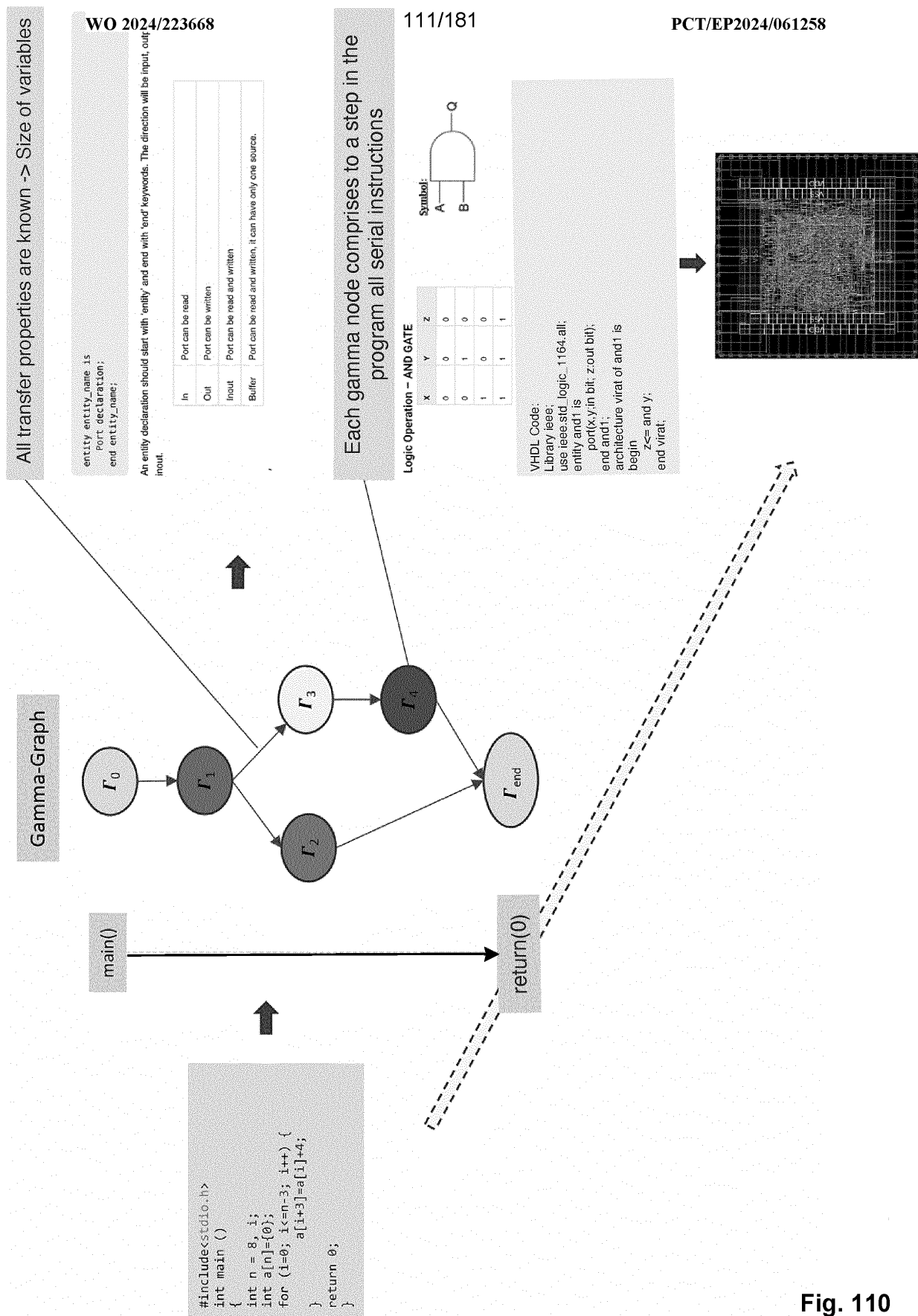


Fig. 110

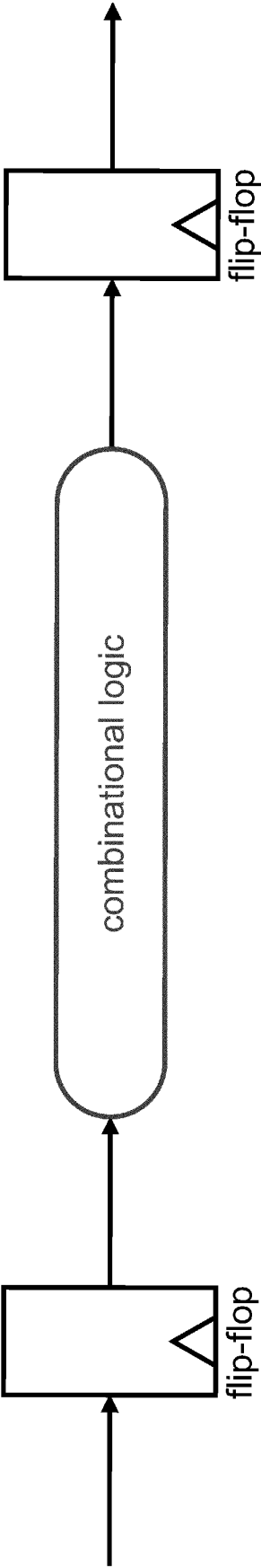


Fig. 111a



Fig. 111b

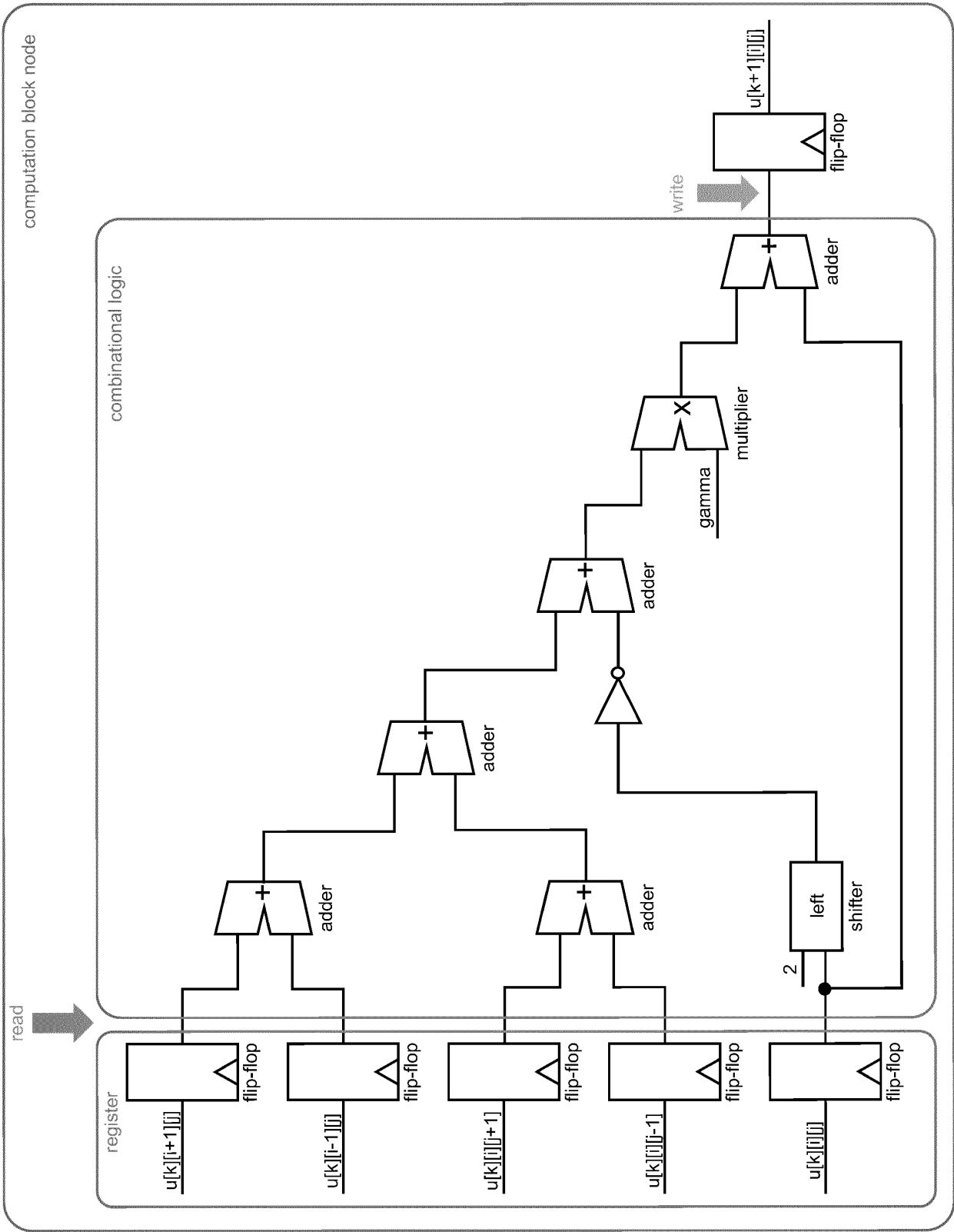


Fig. 111c

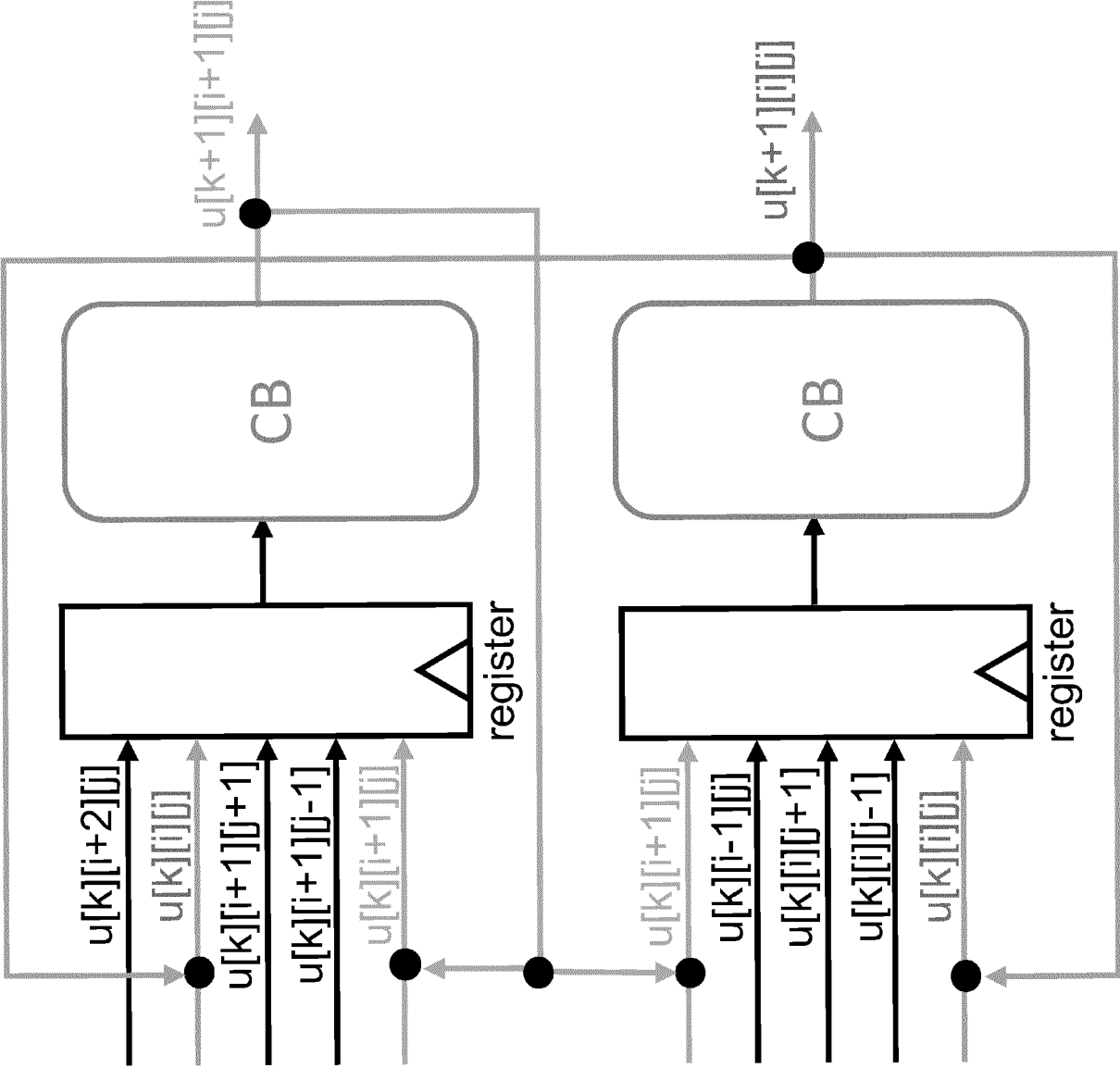


Fig. 111d

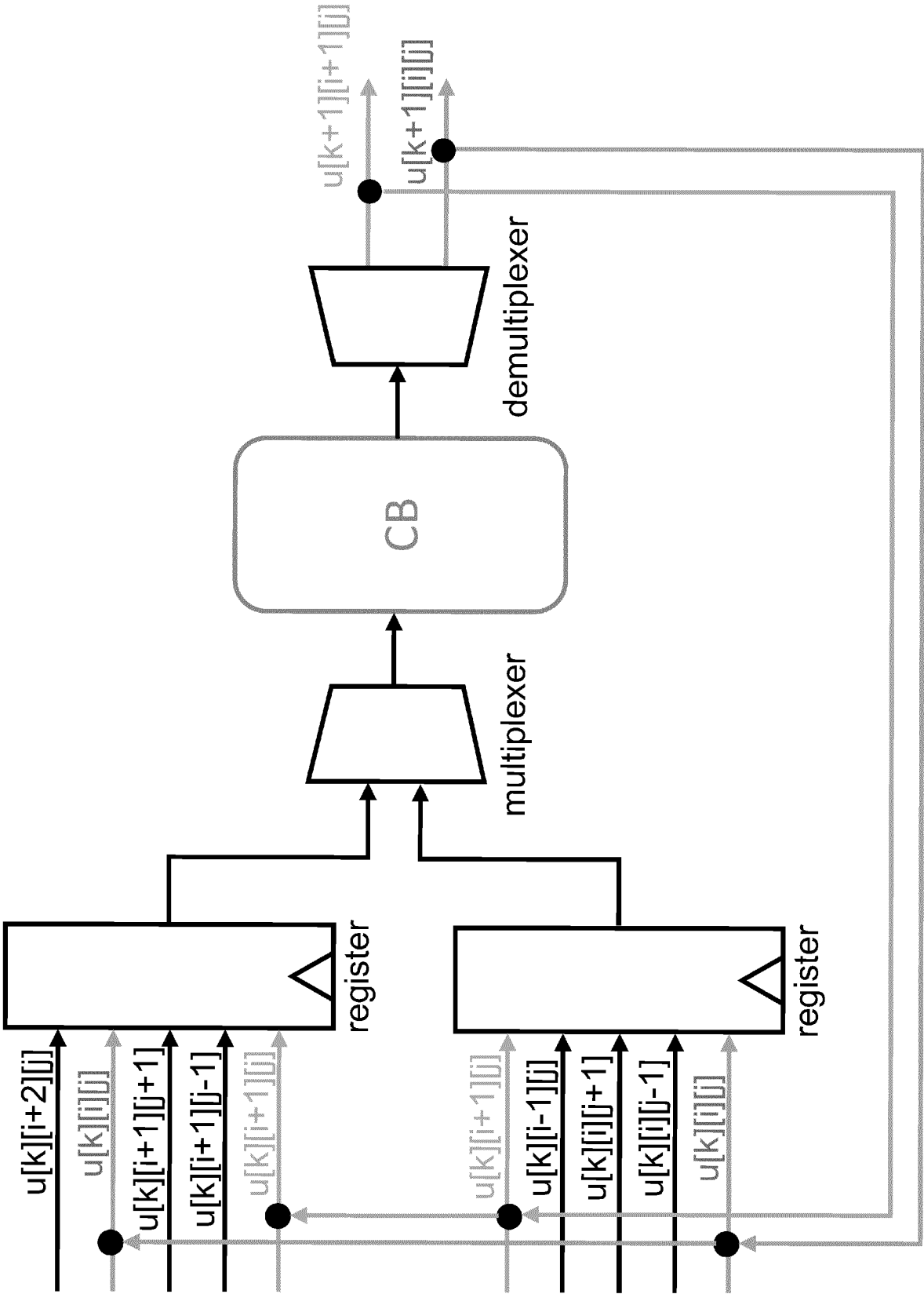


Fig. 111e

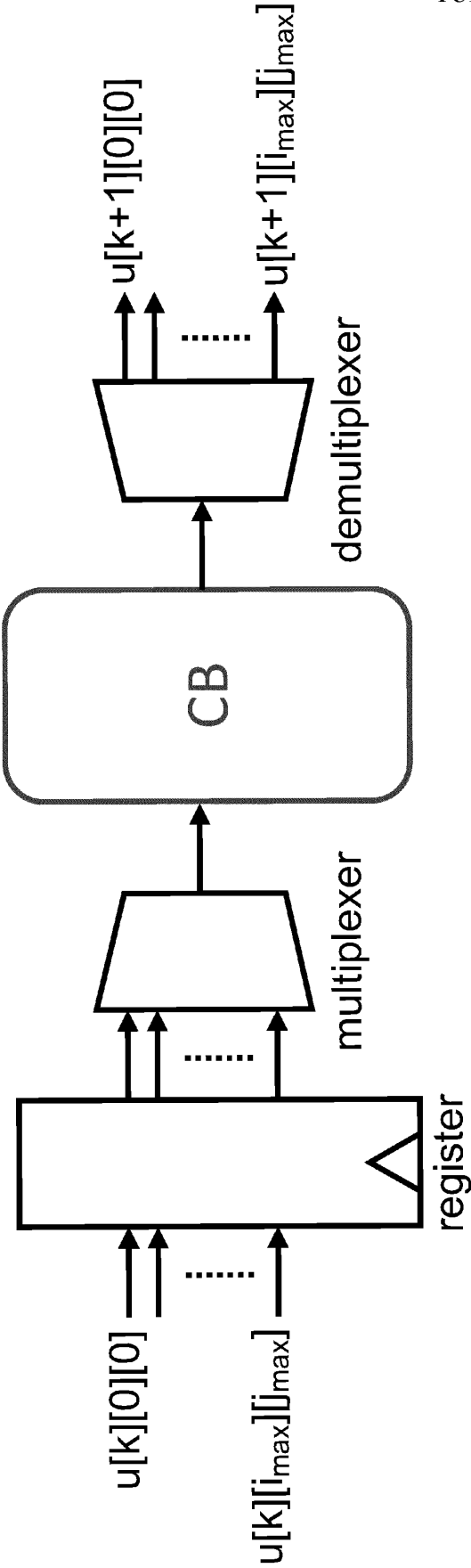


Fig. 111f

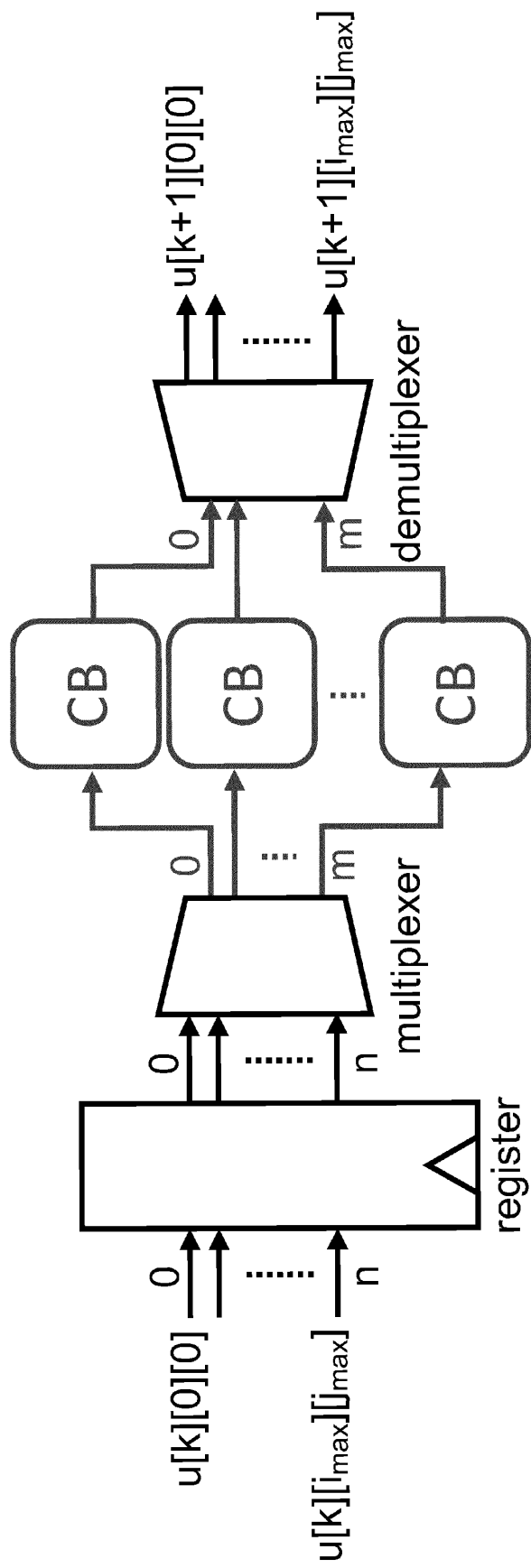


Fig. 111g

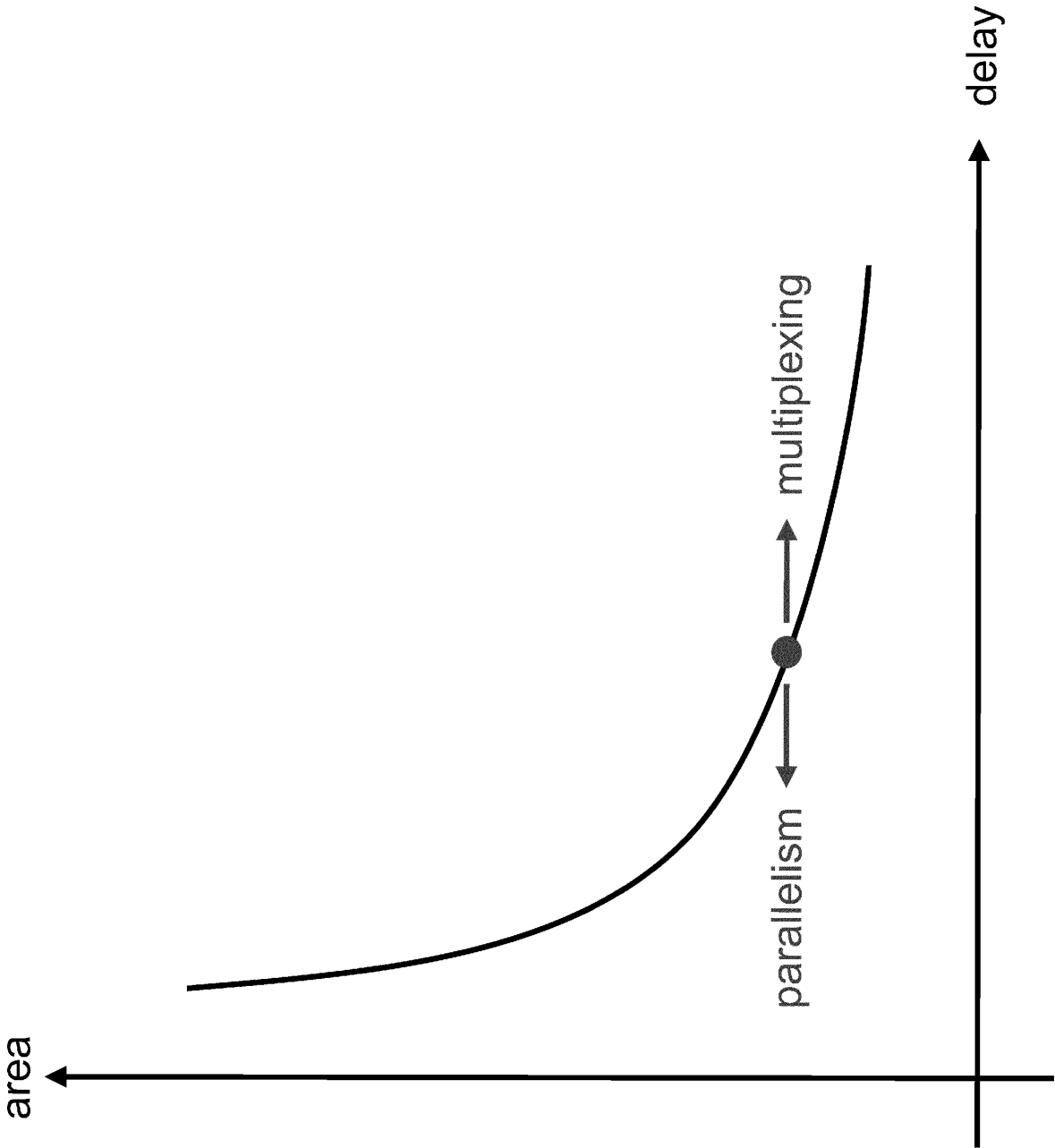


Fig. 111h

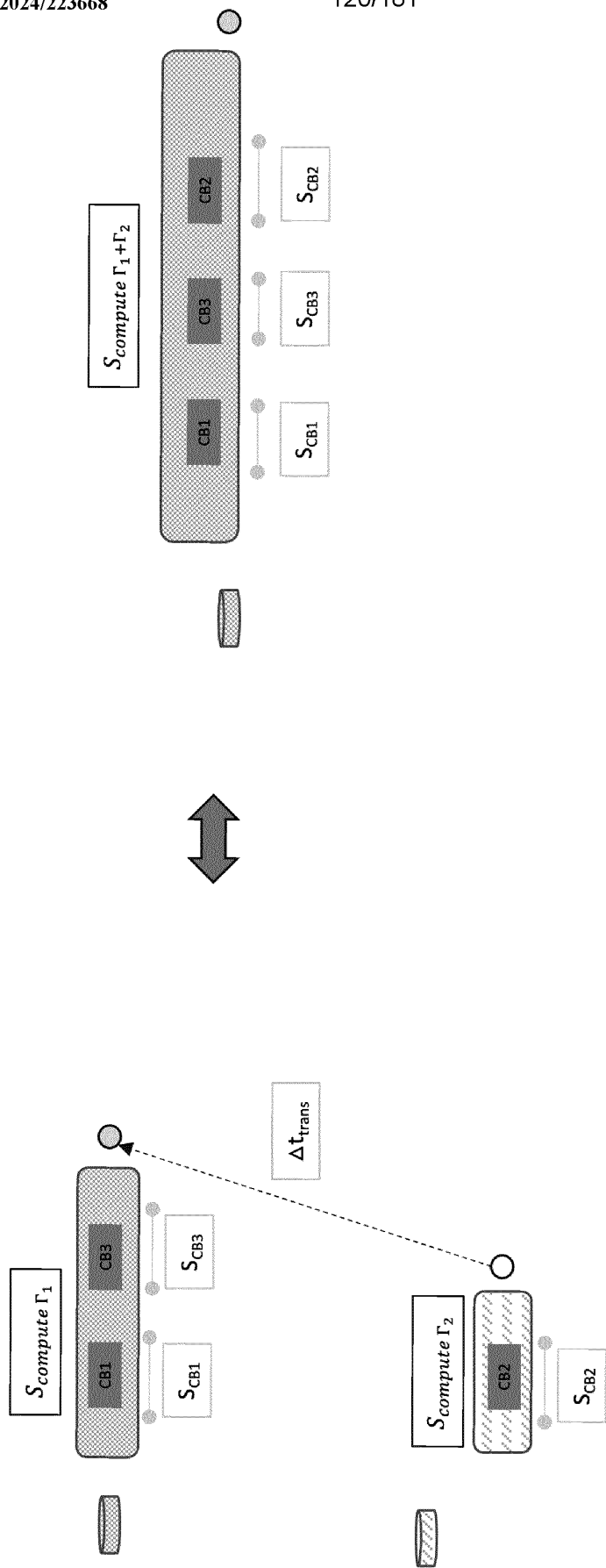


Fig. 112

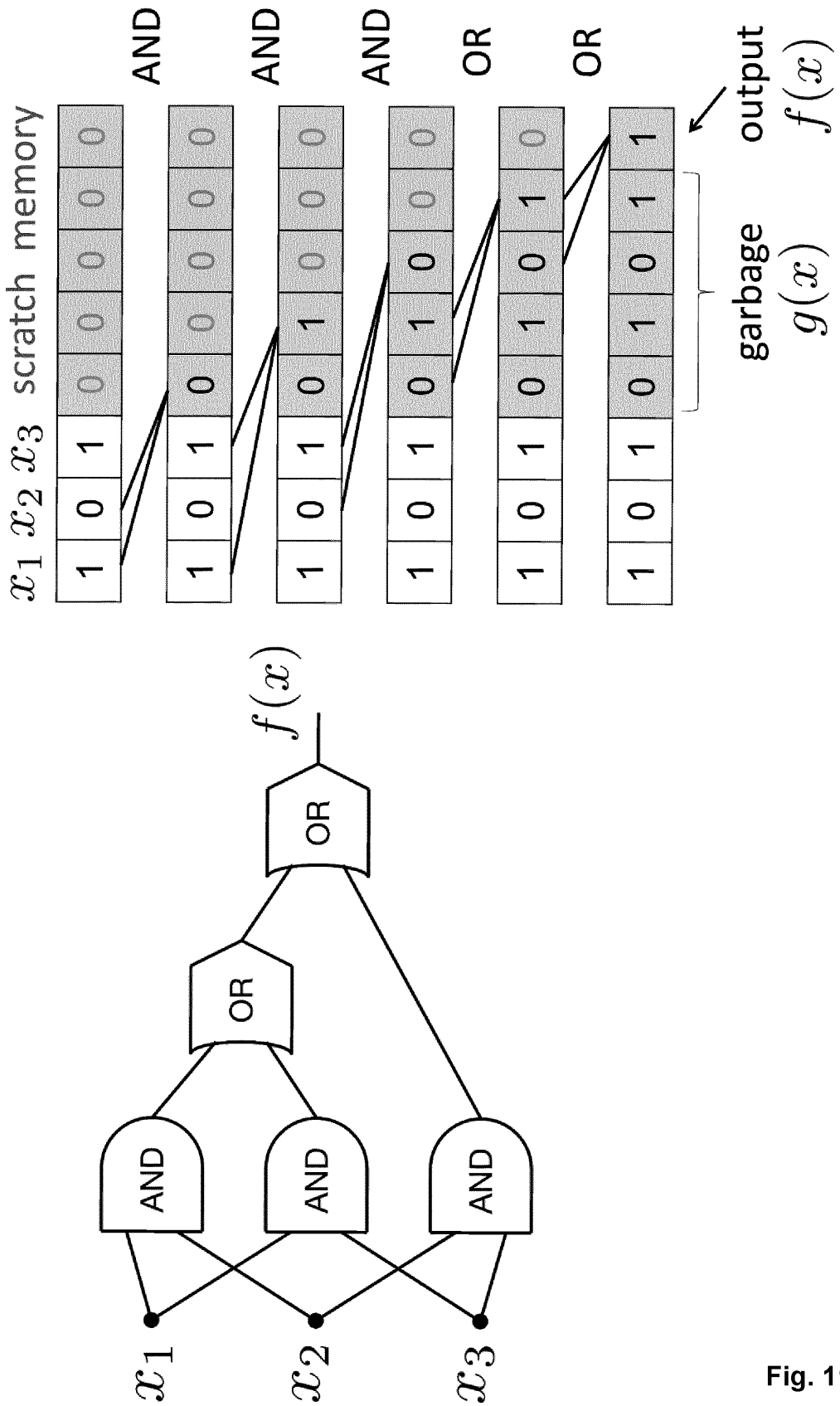


Fig. 113

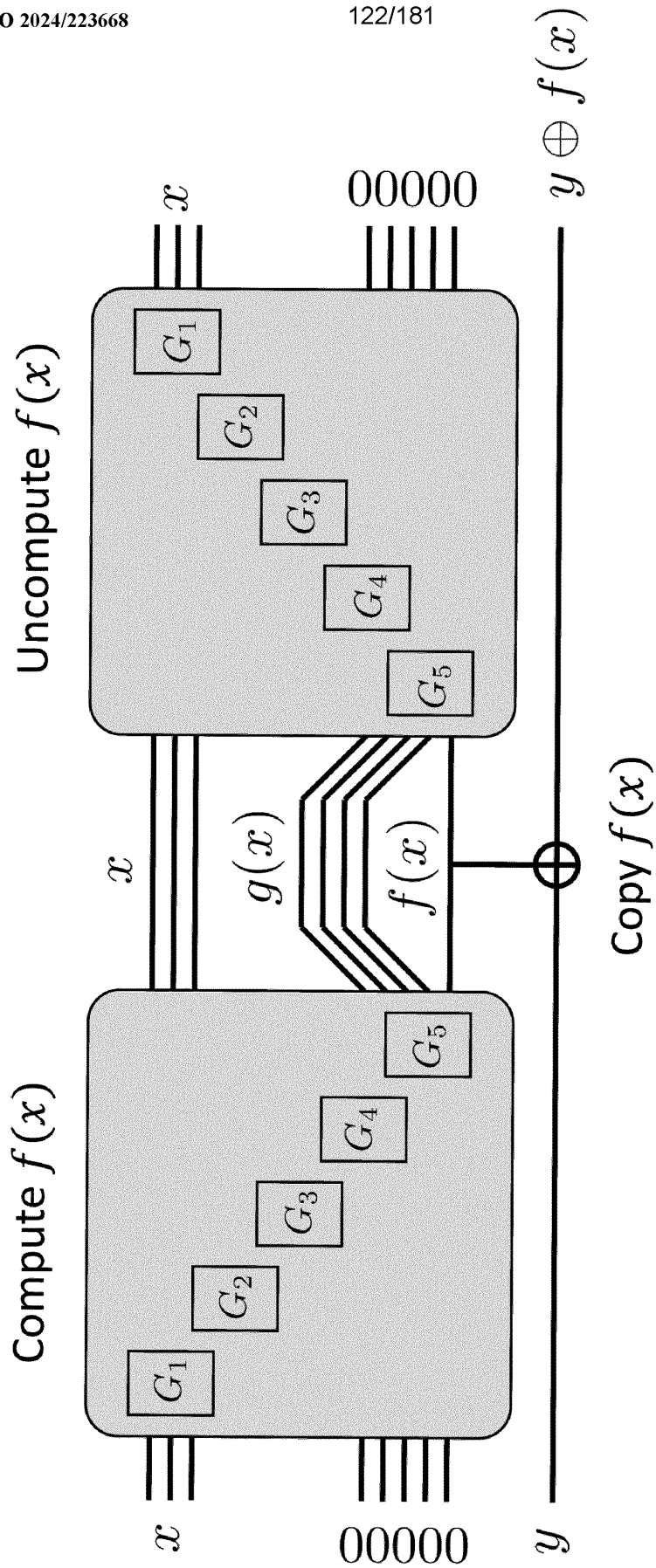


Fig. 114

Fig. 59: $t_w = \kappa * (t_a + t_b + t_c + t_d - 4 * t_e) + t_e$

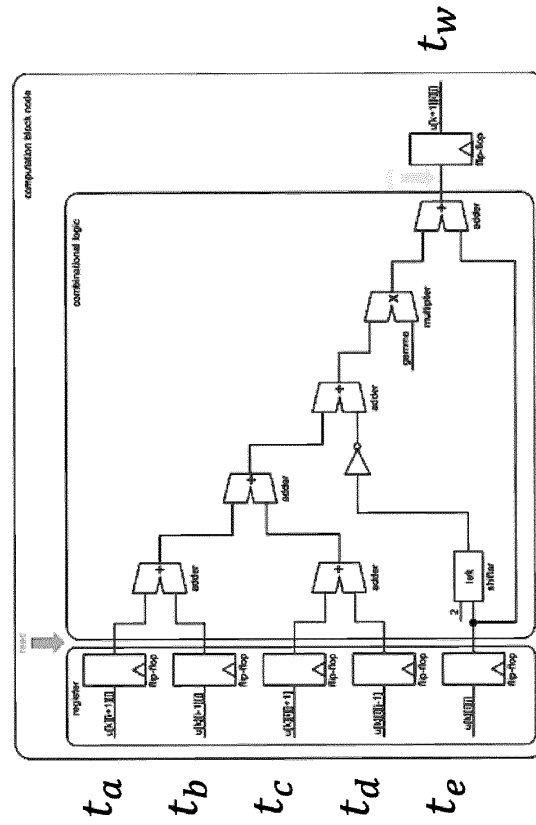
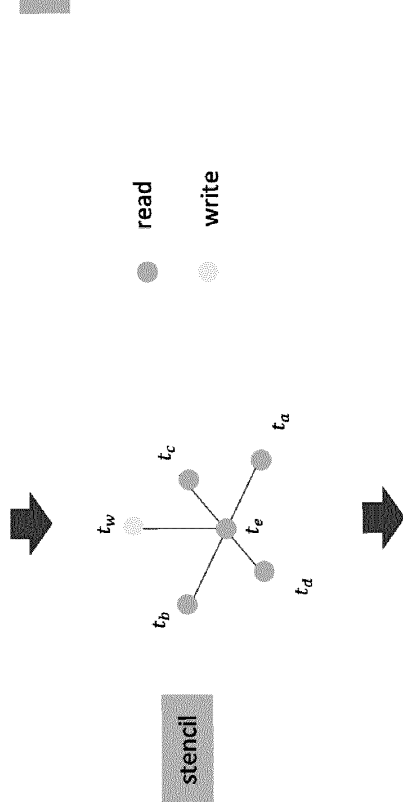


Fig. 115

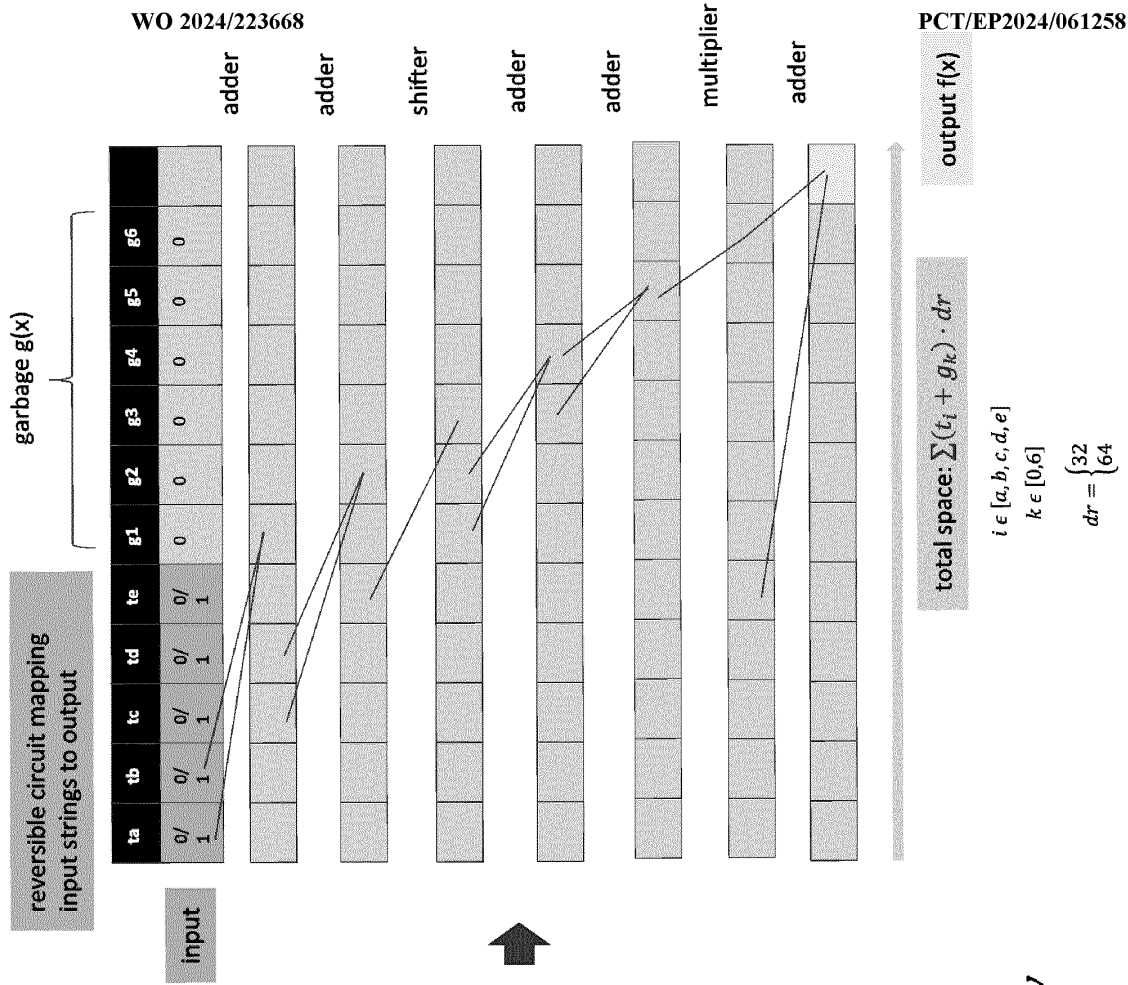


Fig. 111c

combinational ic-circuit

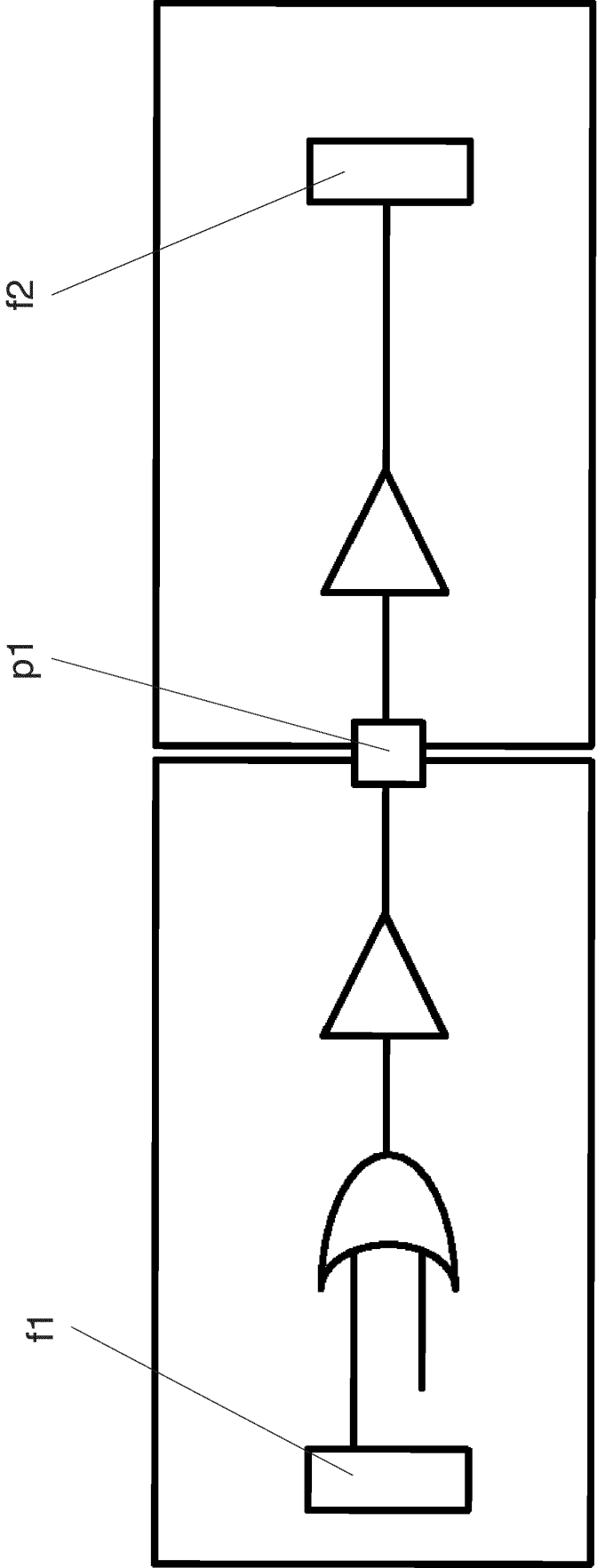


Fig. 116

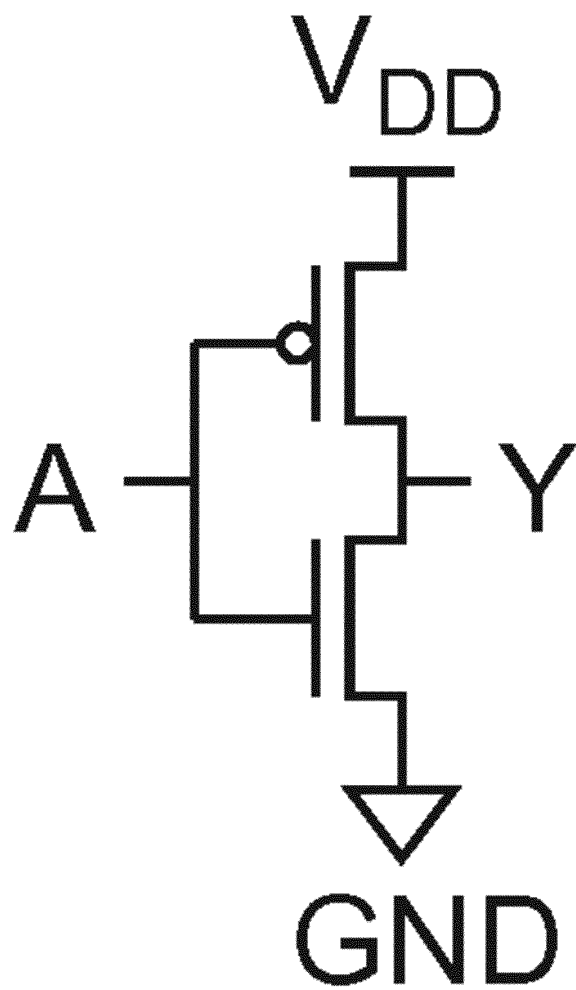


Fig. 117a

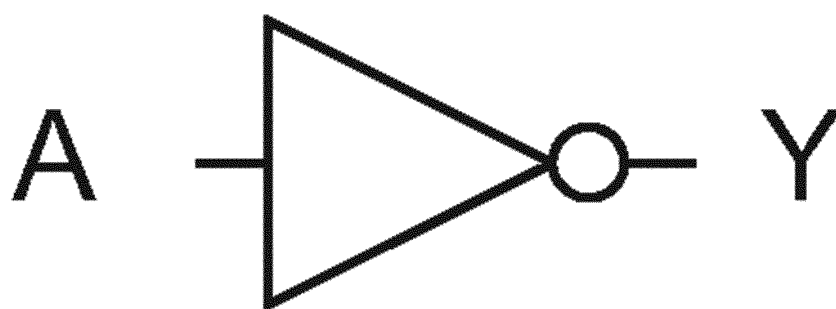


Fig. 117b

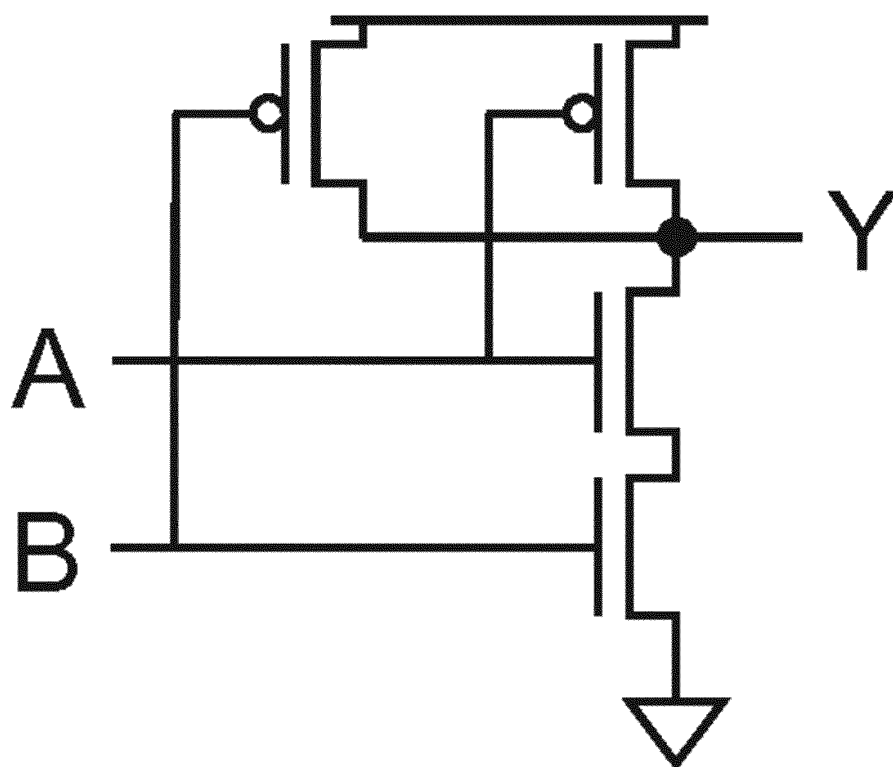


Fig. 117c

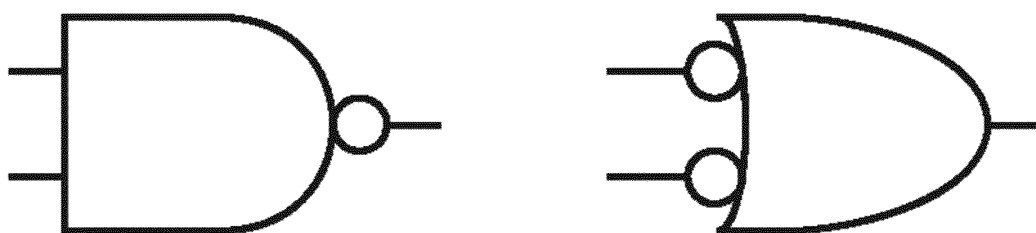


Fig. 117d

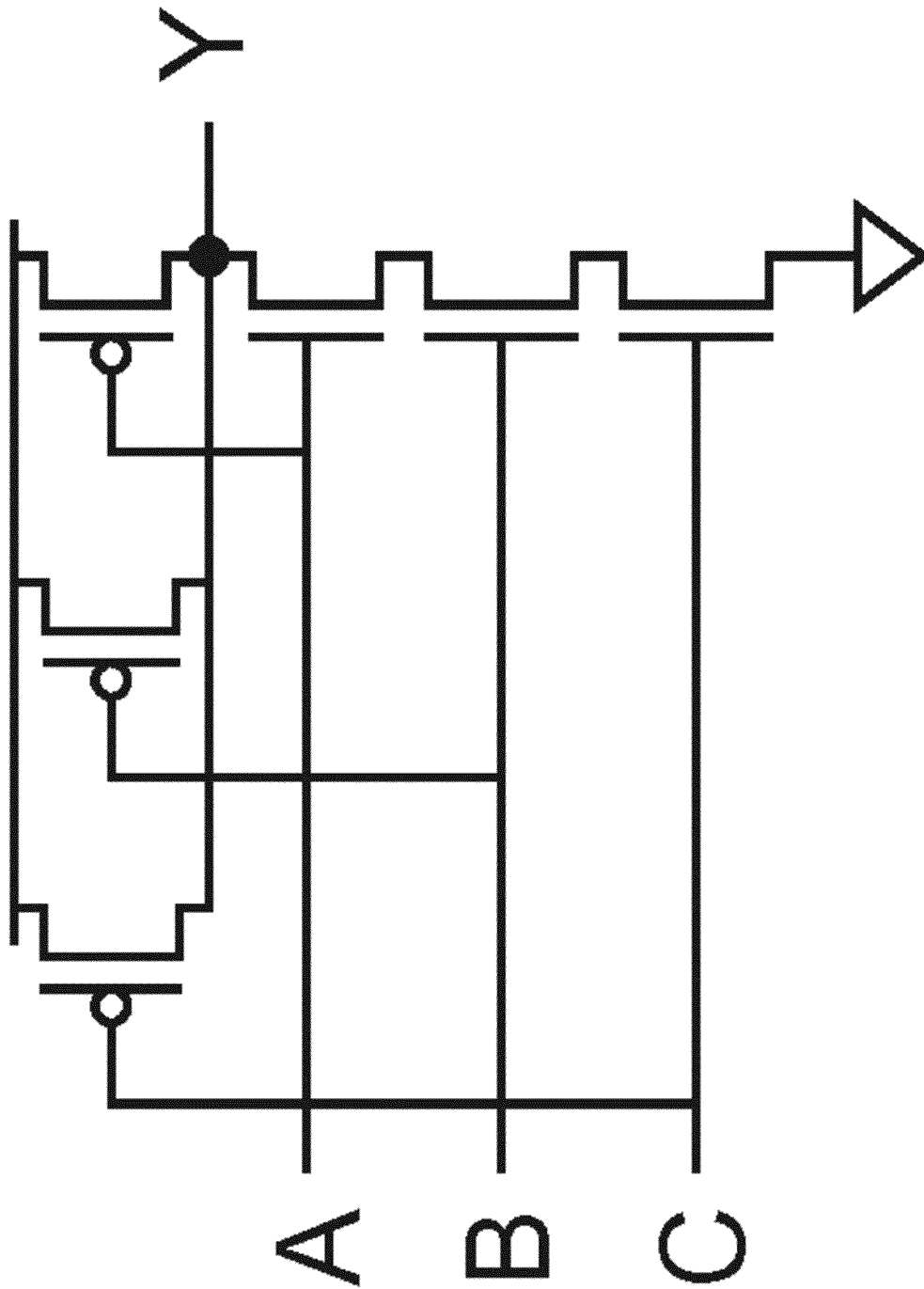


Fig. 117e

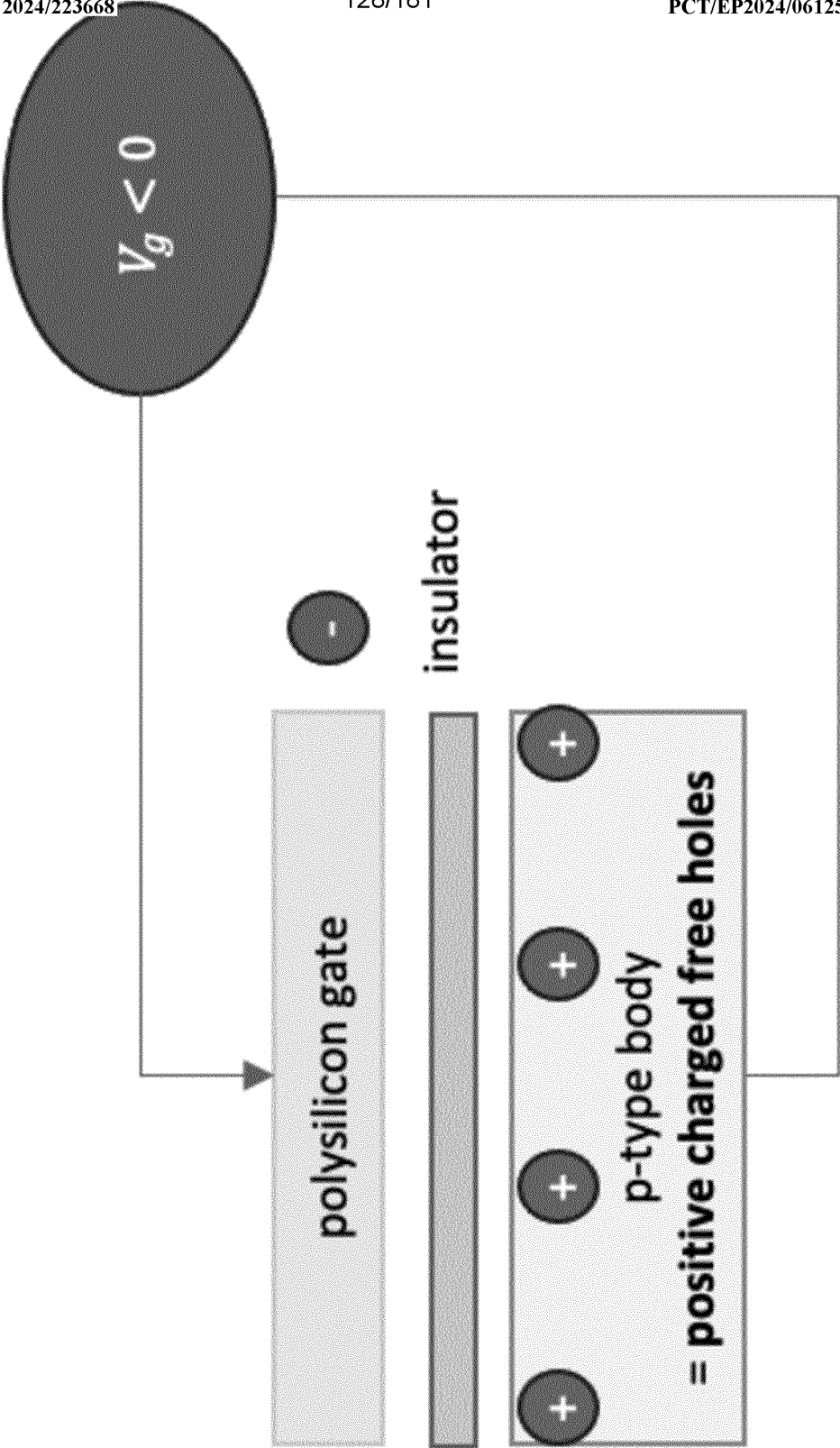


Fig. 118a

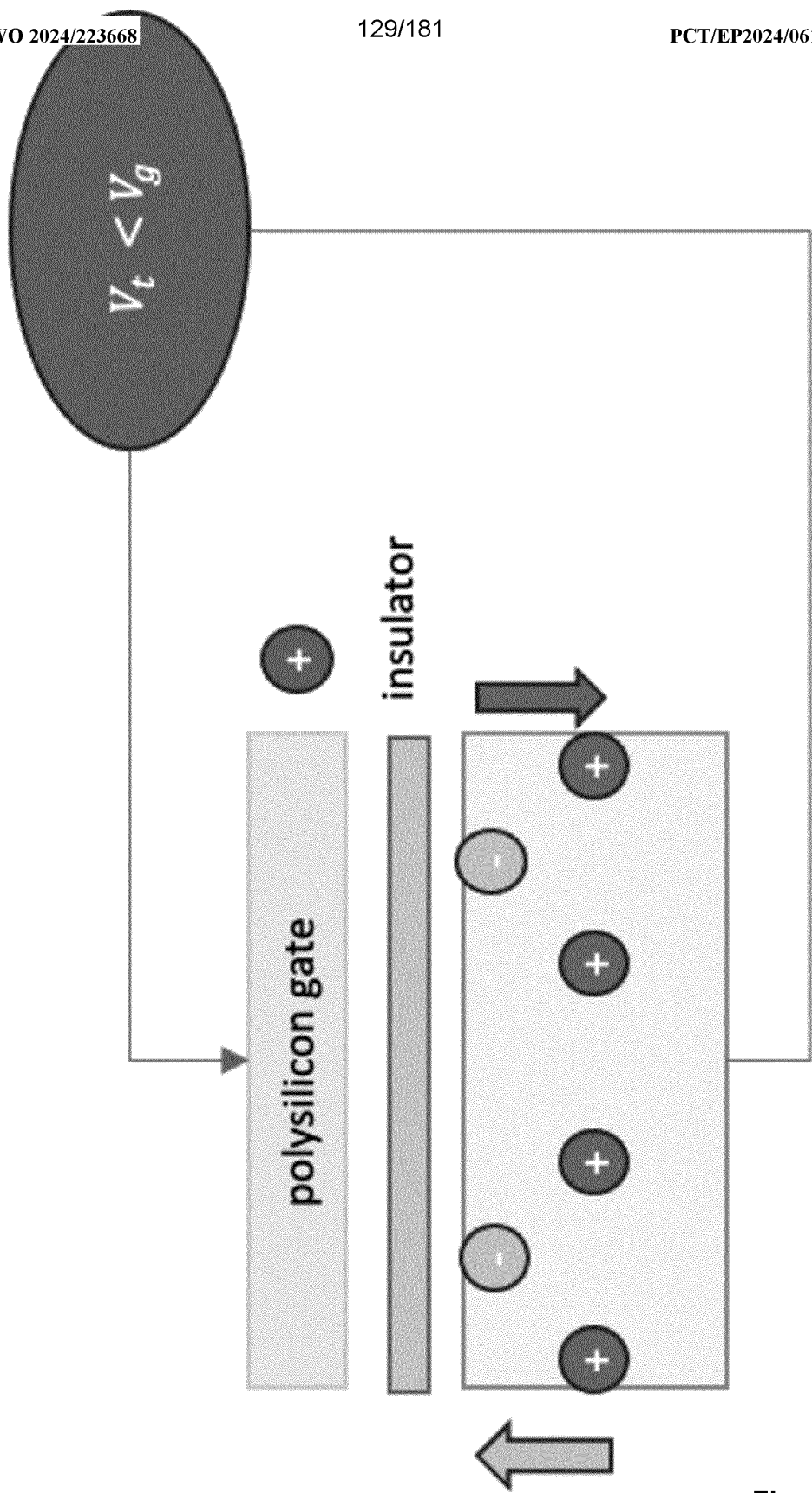


Fig. 118b

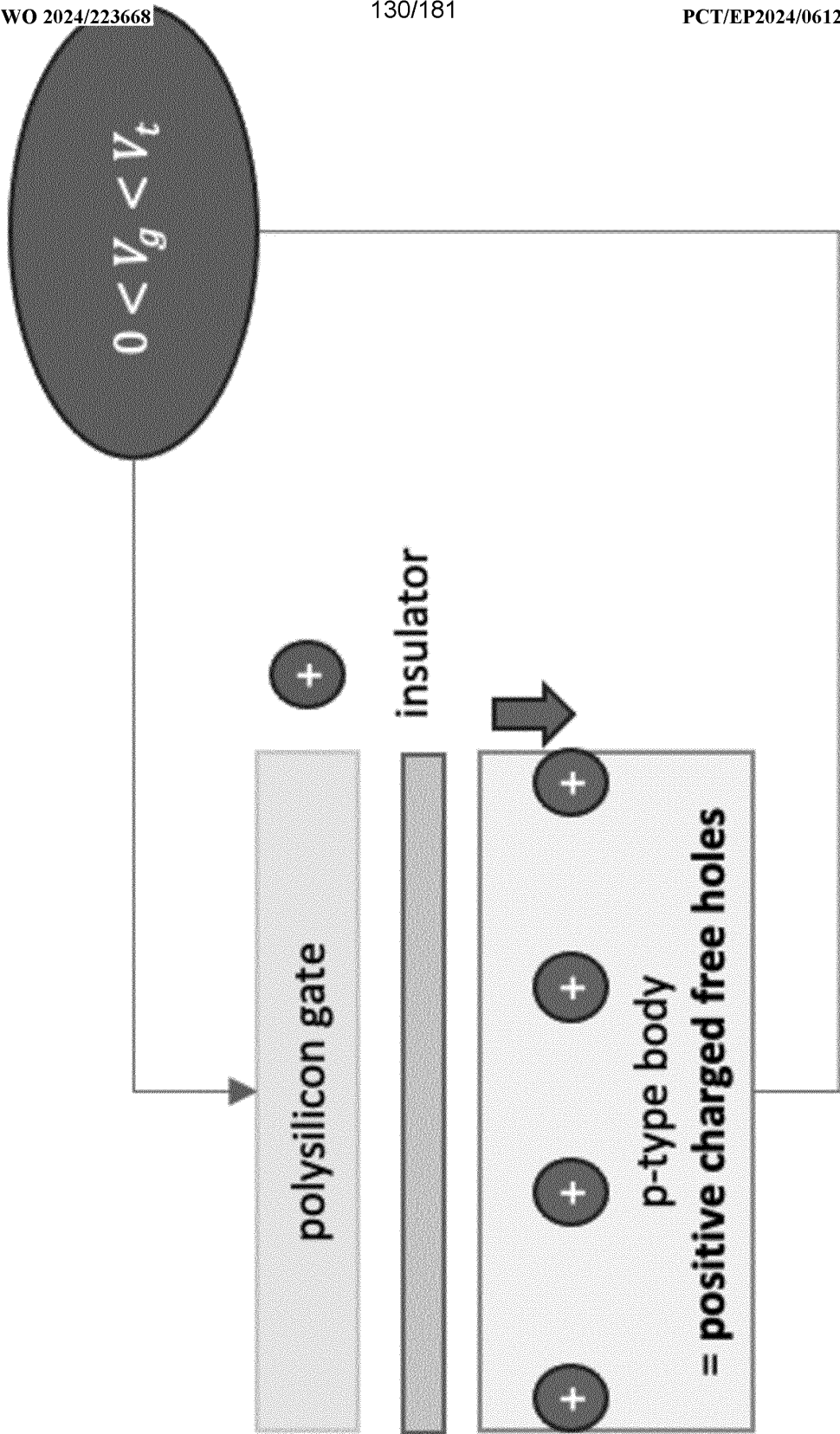


Fig. 118c

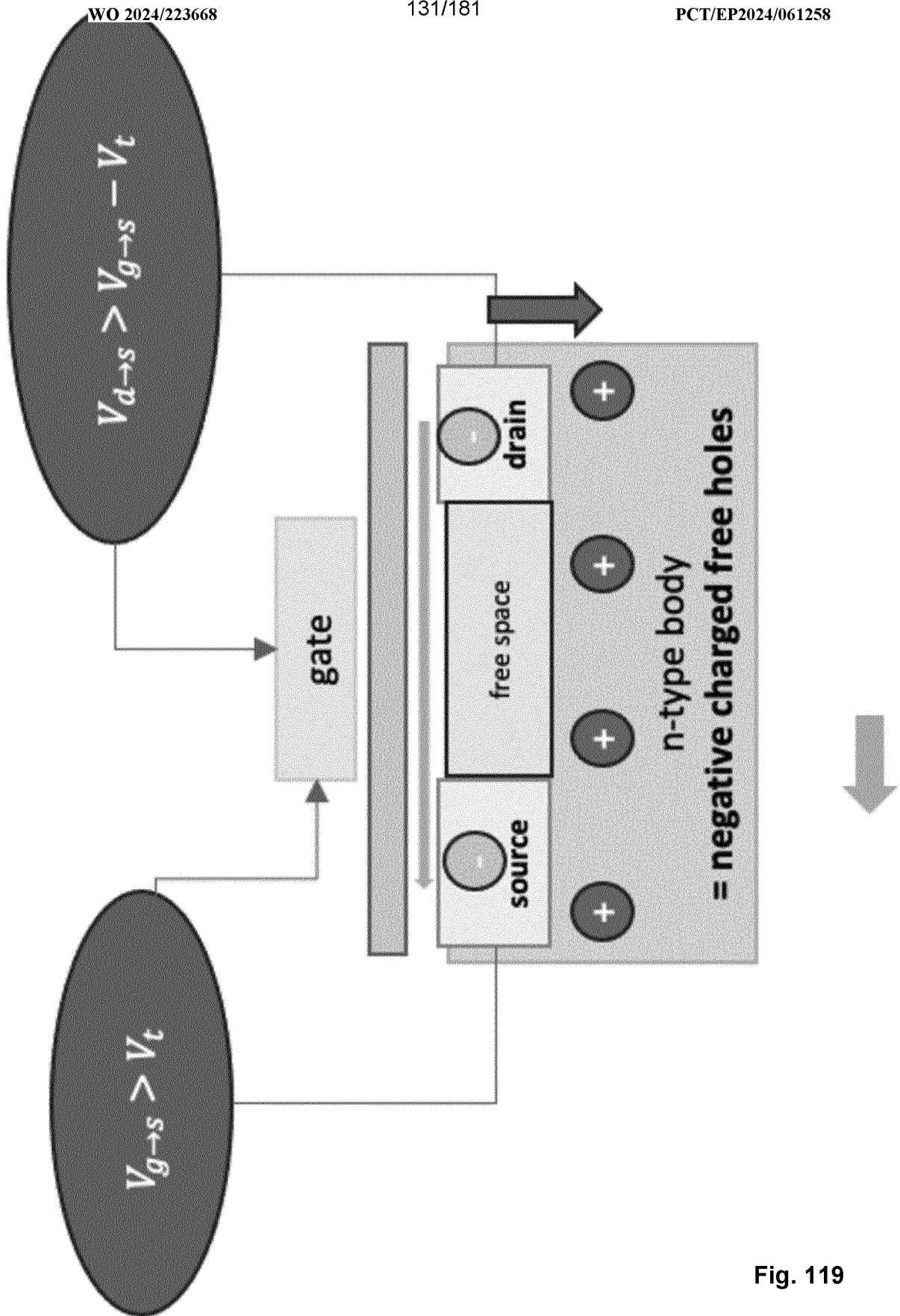


Fig. 119

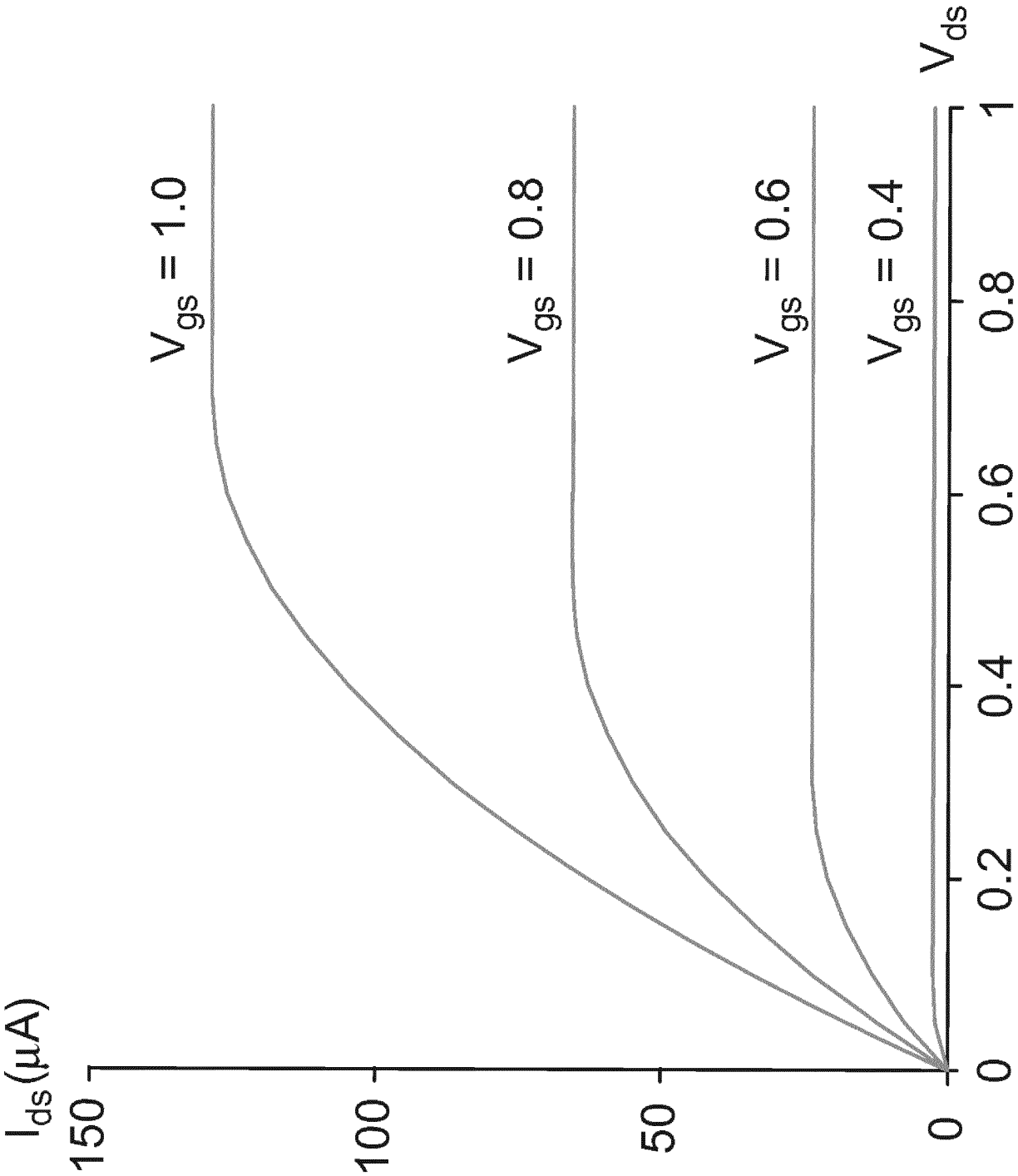


Fig. 120a

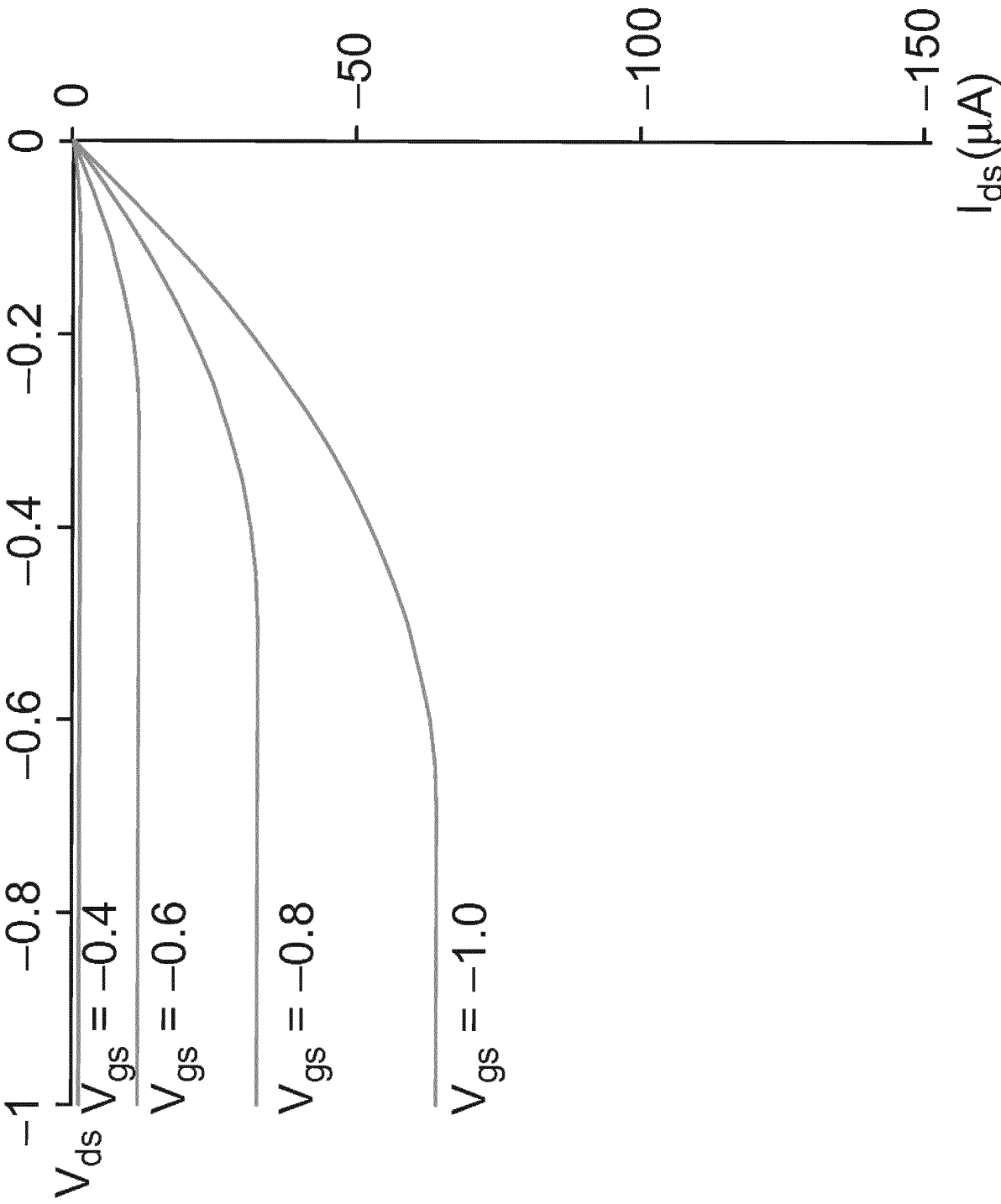


Fig. 120b

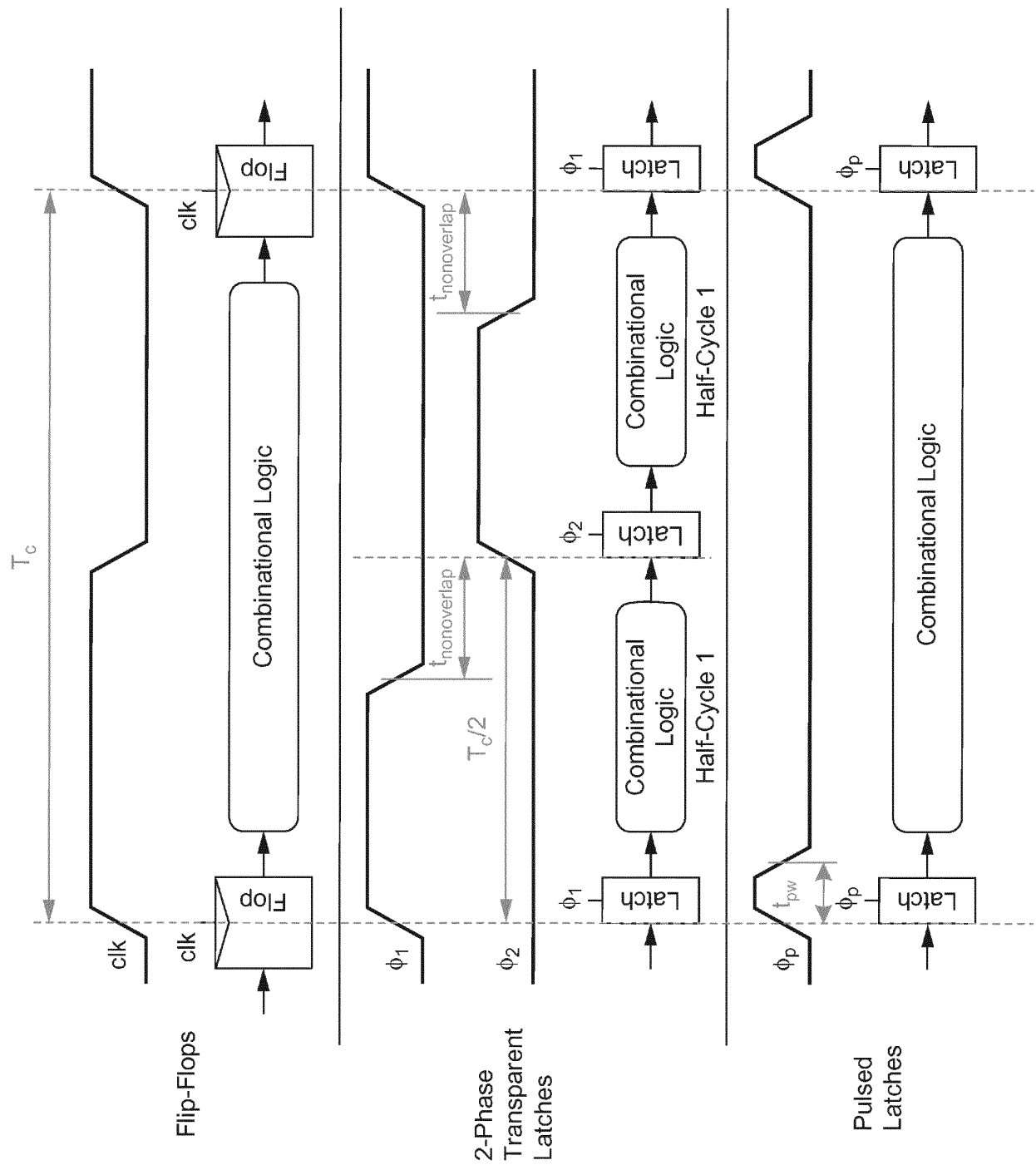


Fig. 121

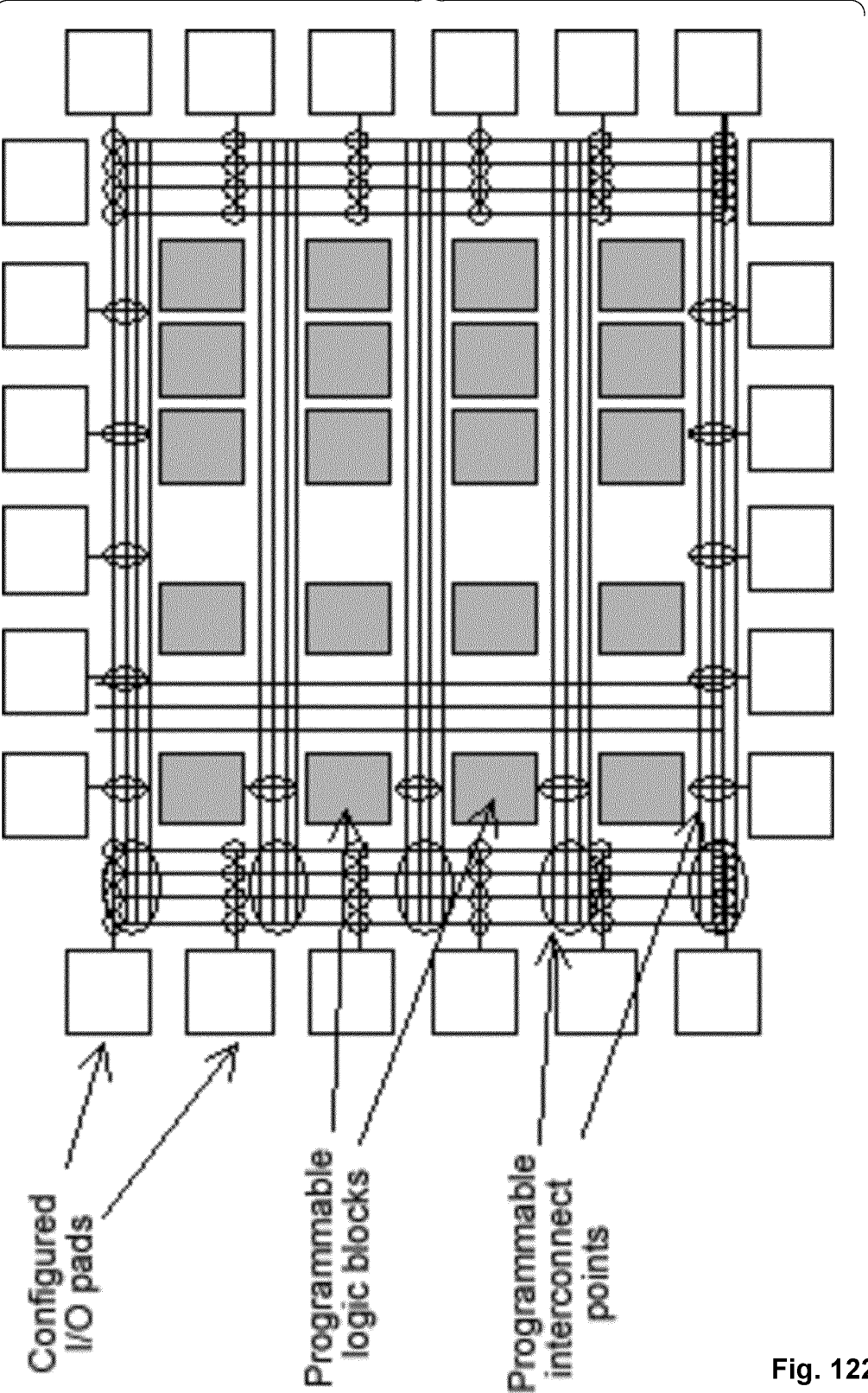


Fig. 122

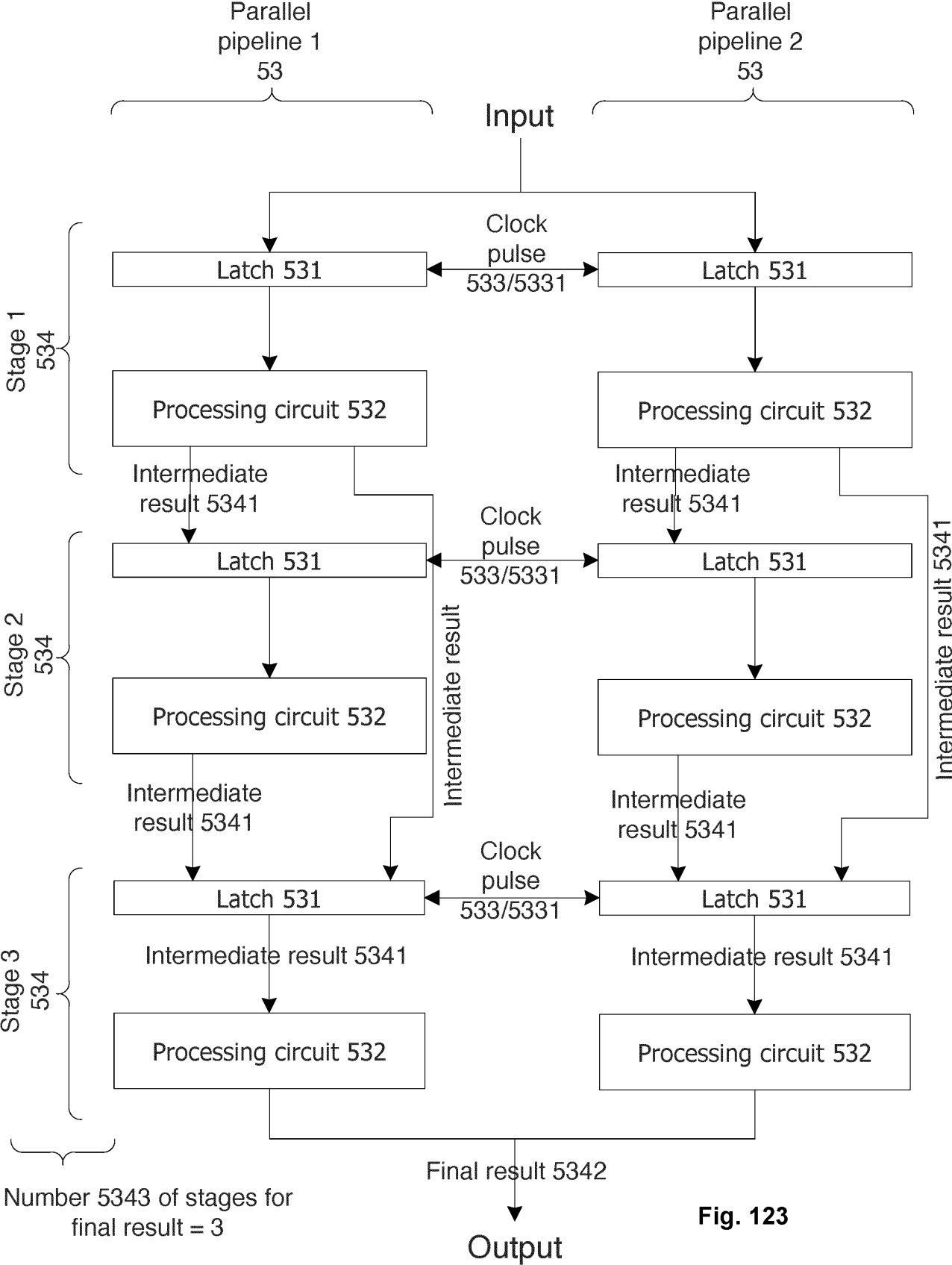


Fig. 123

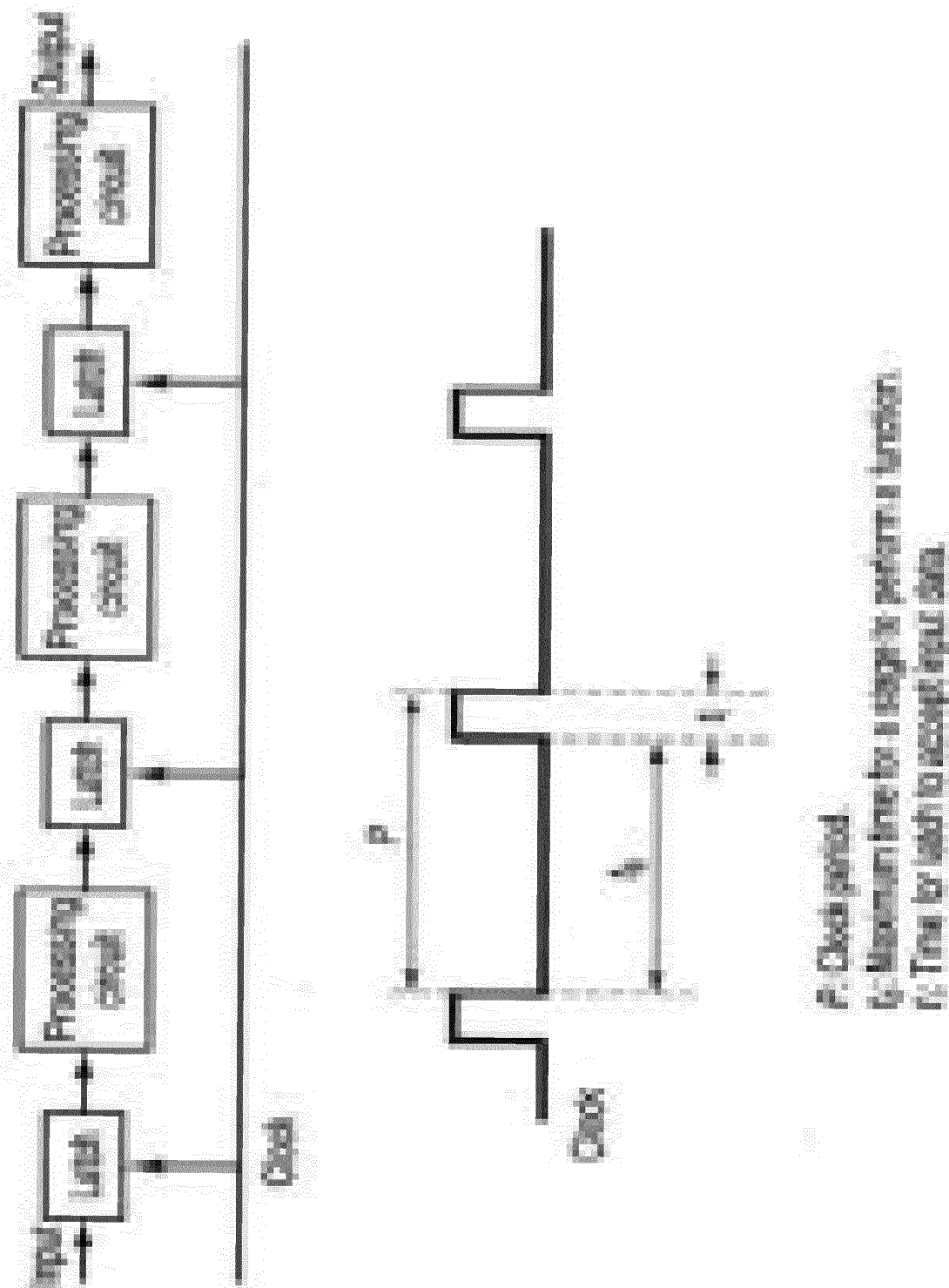


Fig. 124

Computer-aided IC design and manufacturing system 0

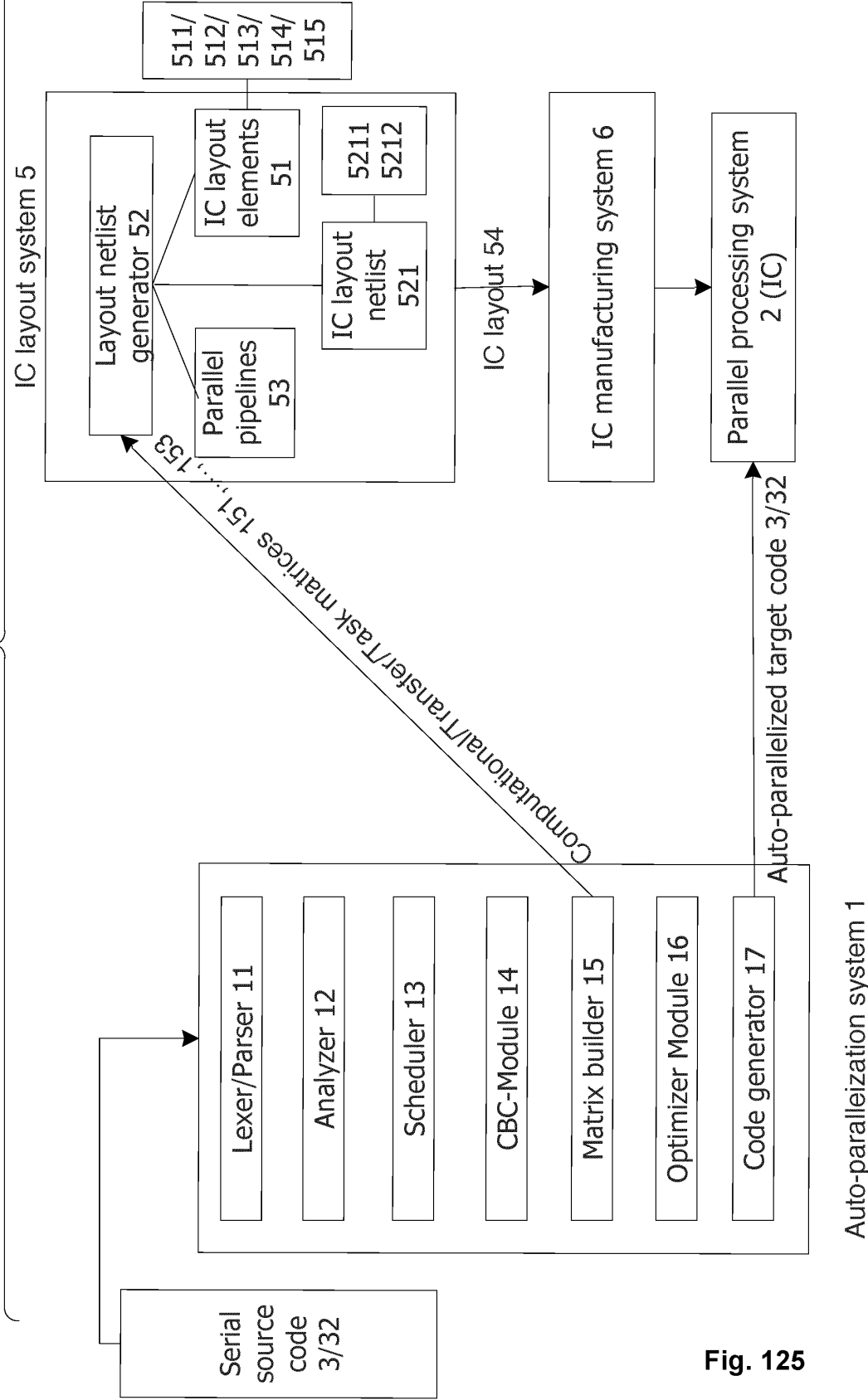


Fig. 125

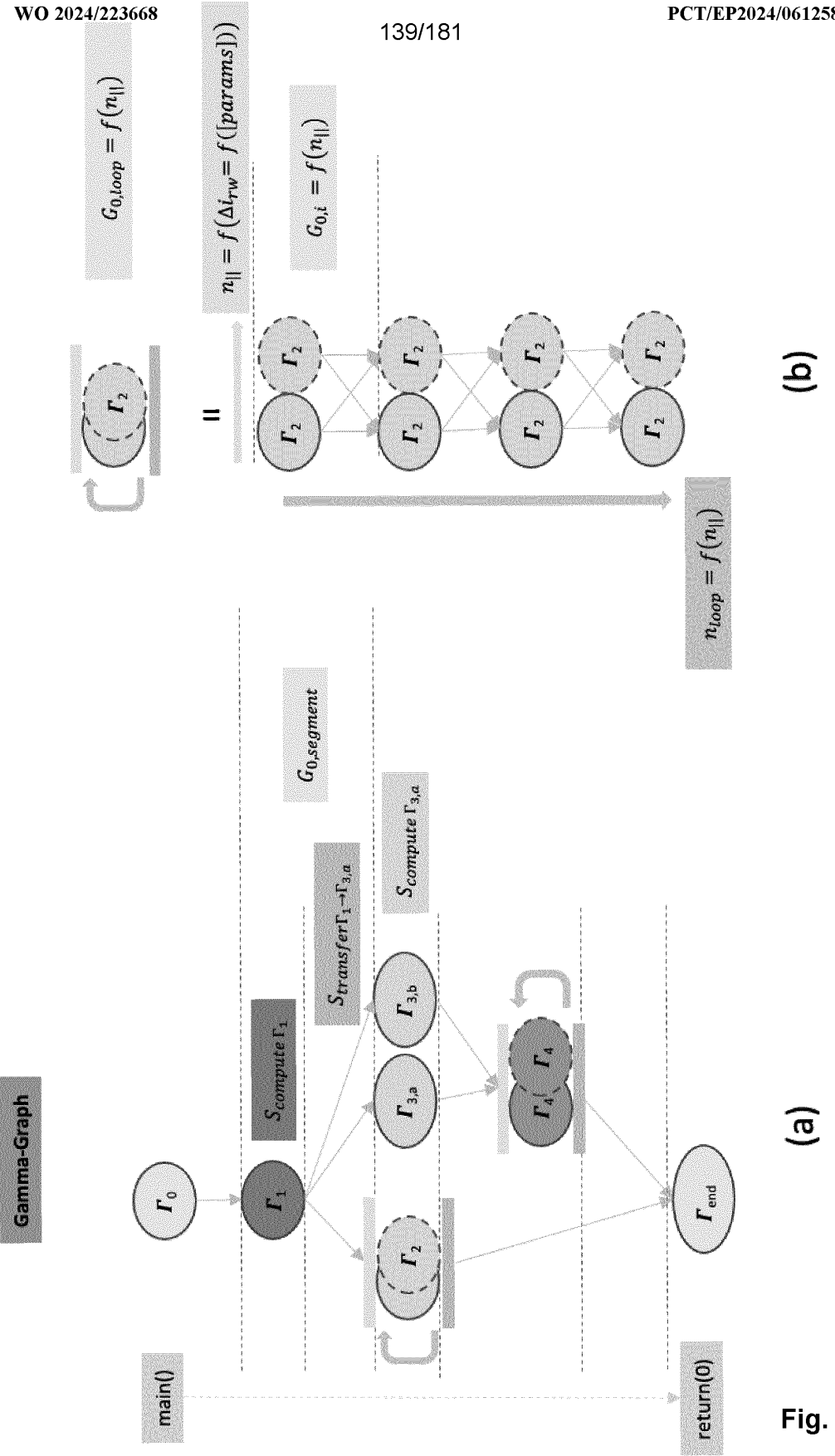


Fig. 126

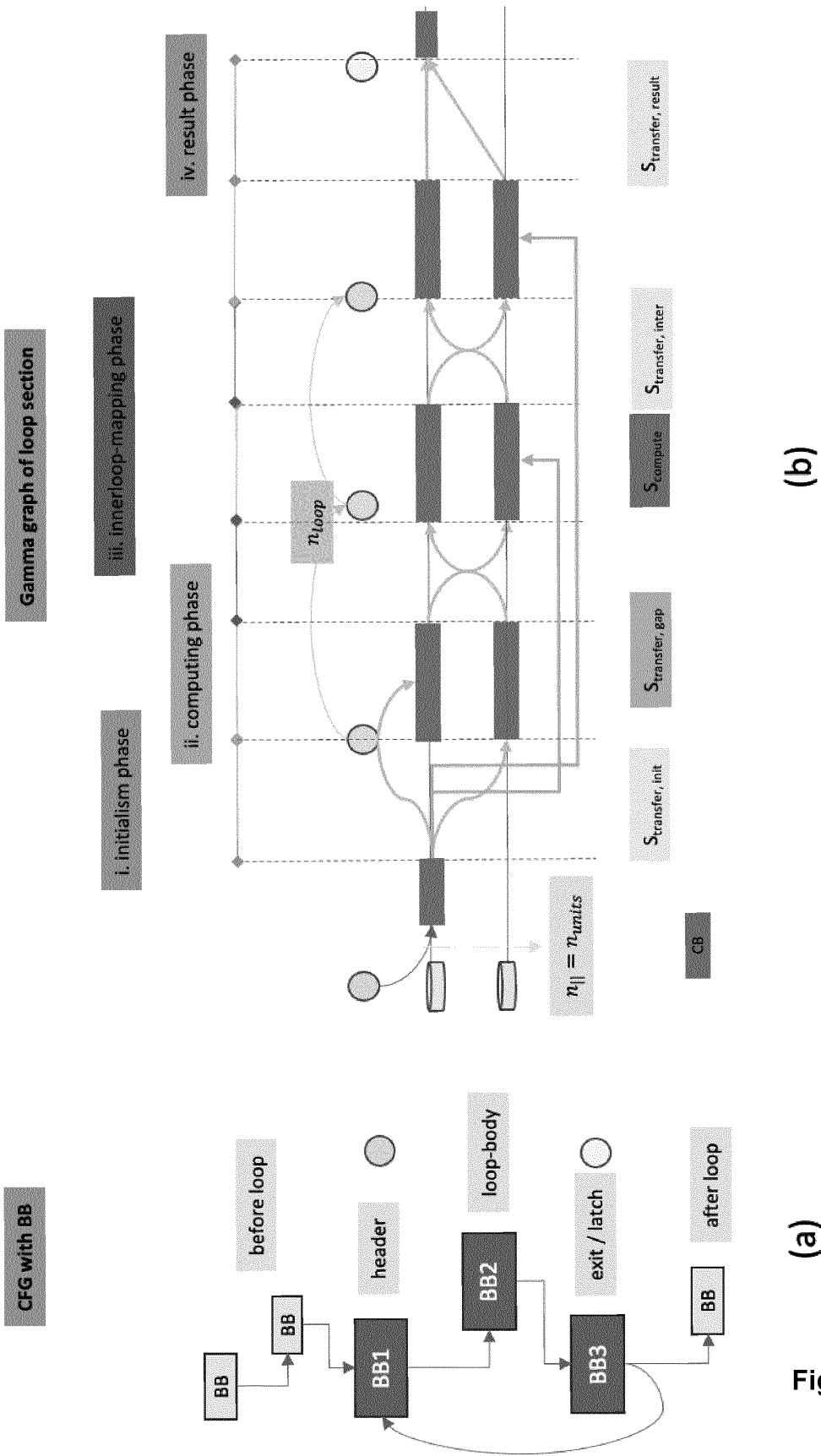


Fig. 127

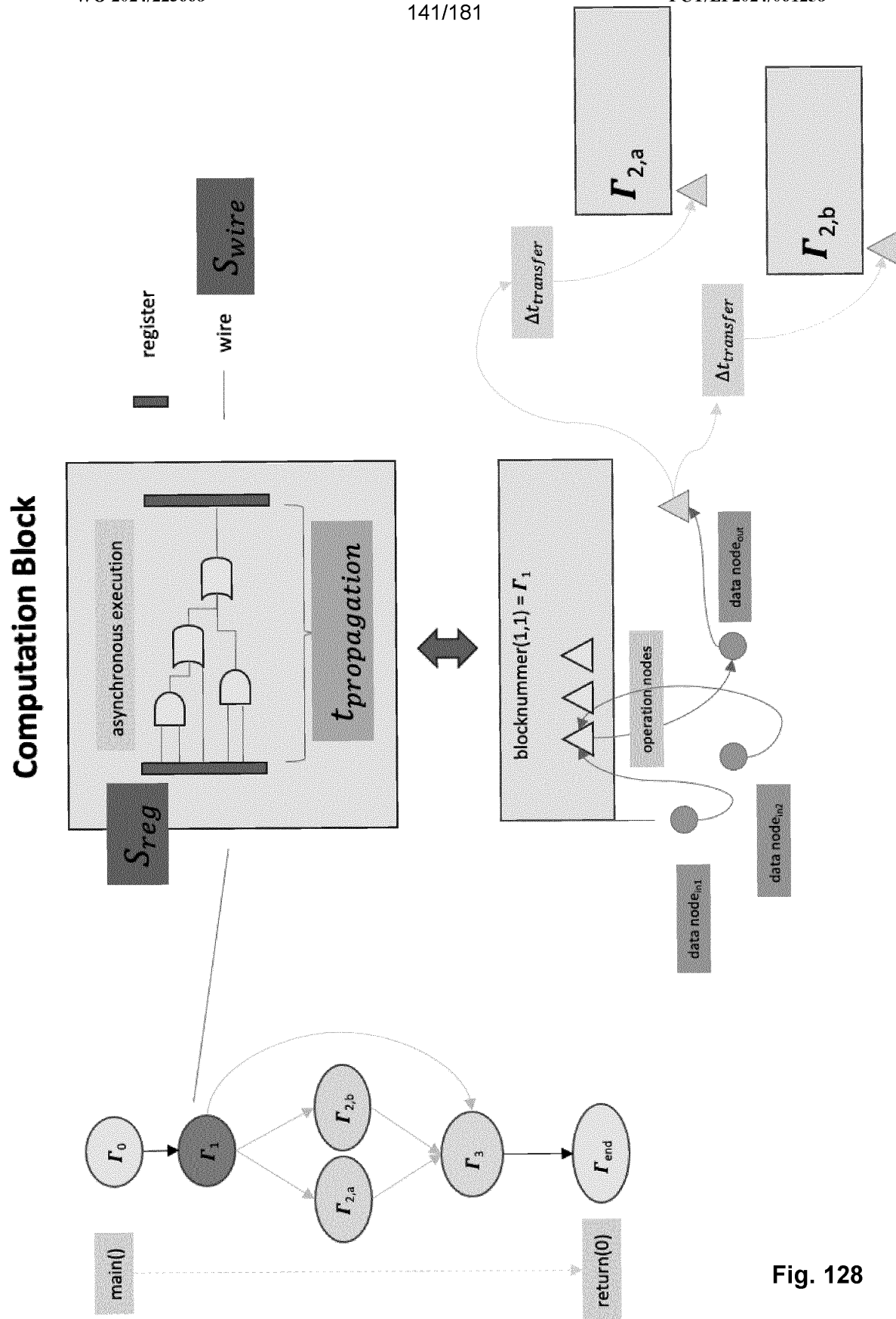


Fig. 128

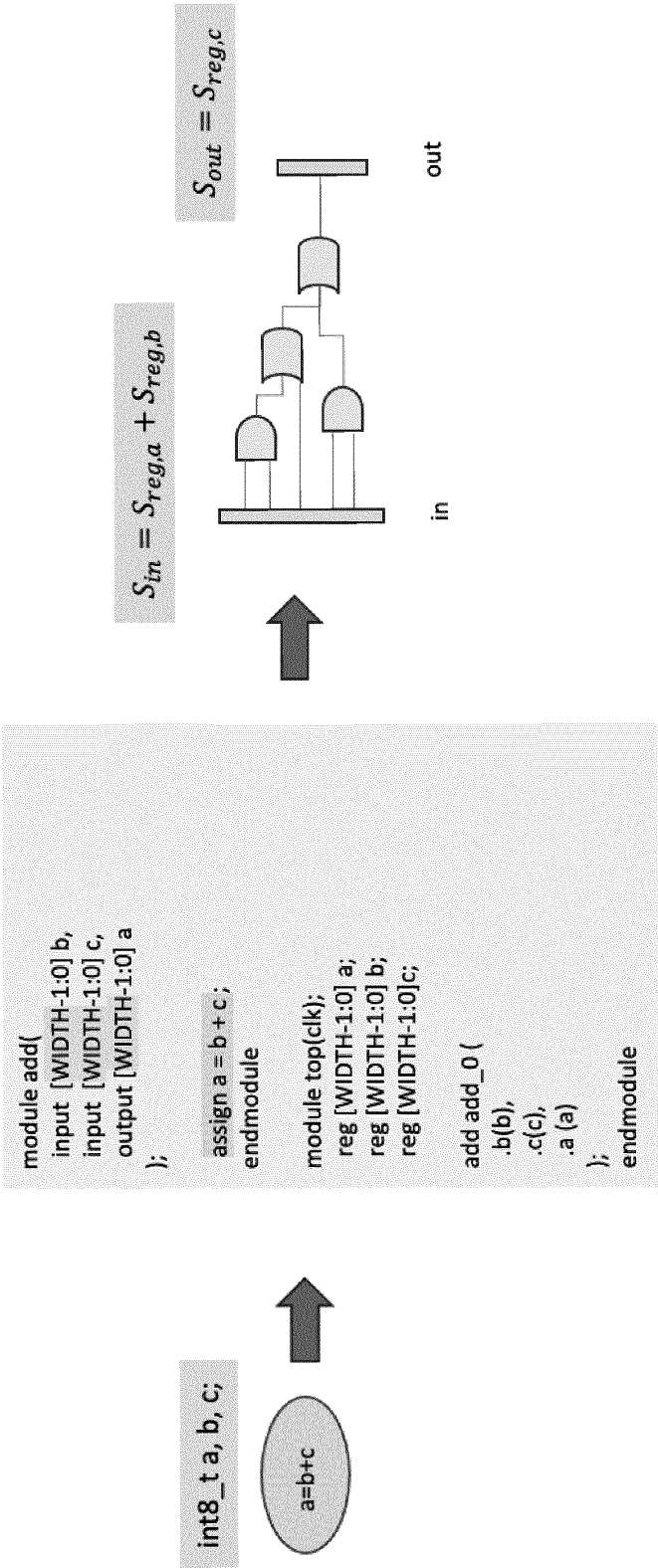


Fig. 129

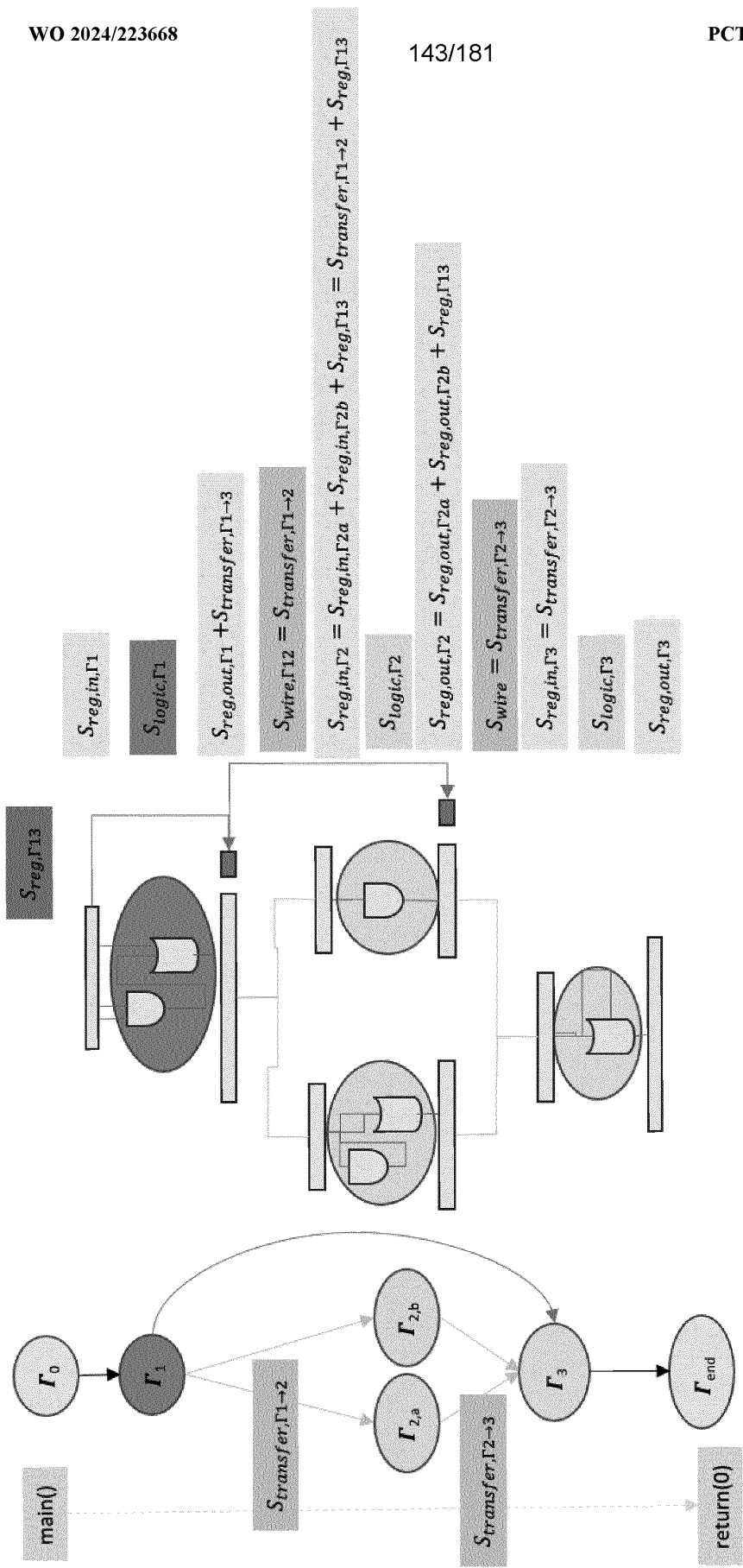


Fig. 130

Finite states

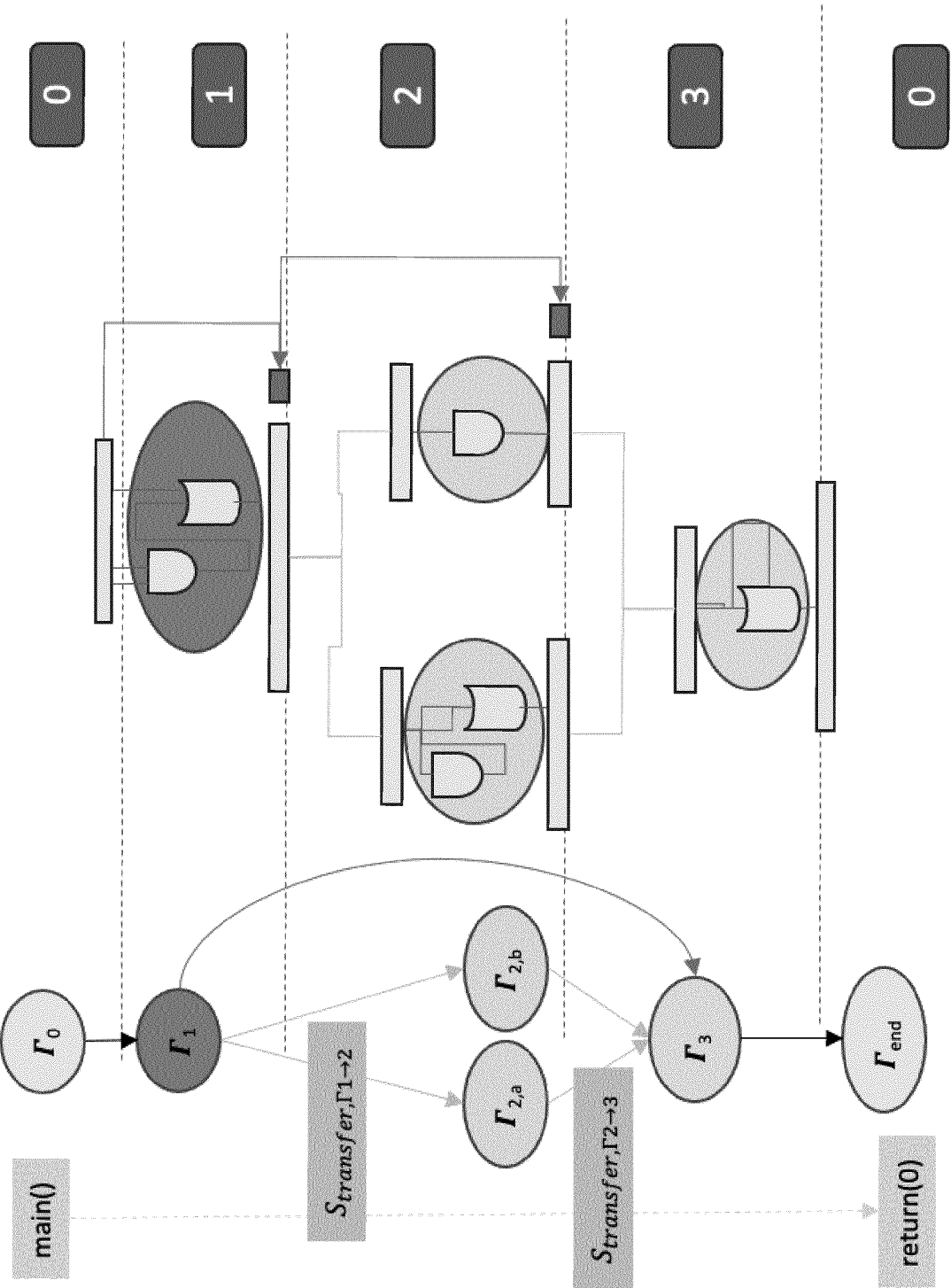


Fig. 131

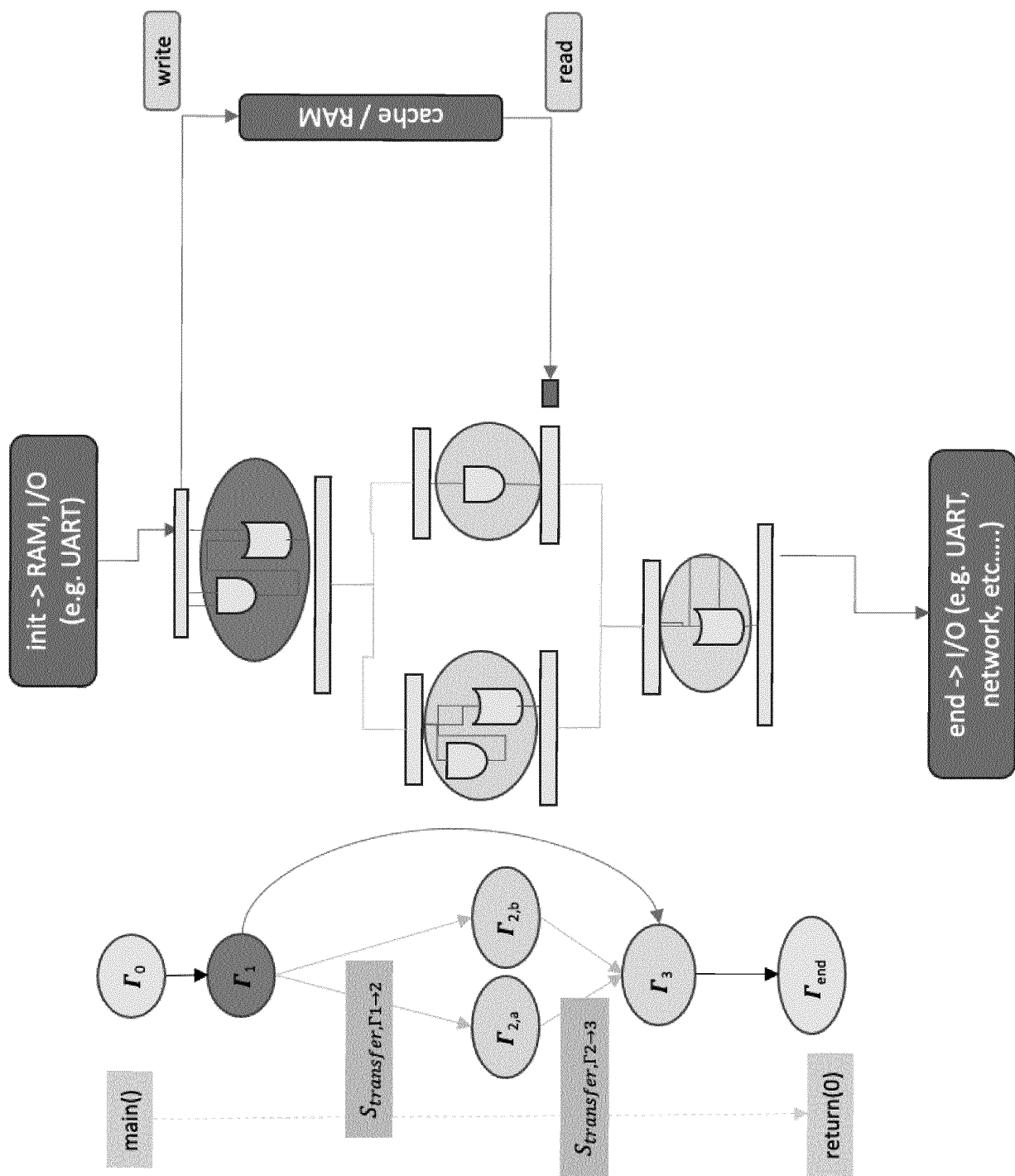


Fig. 132

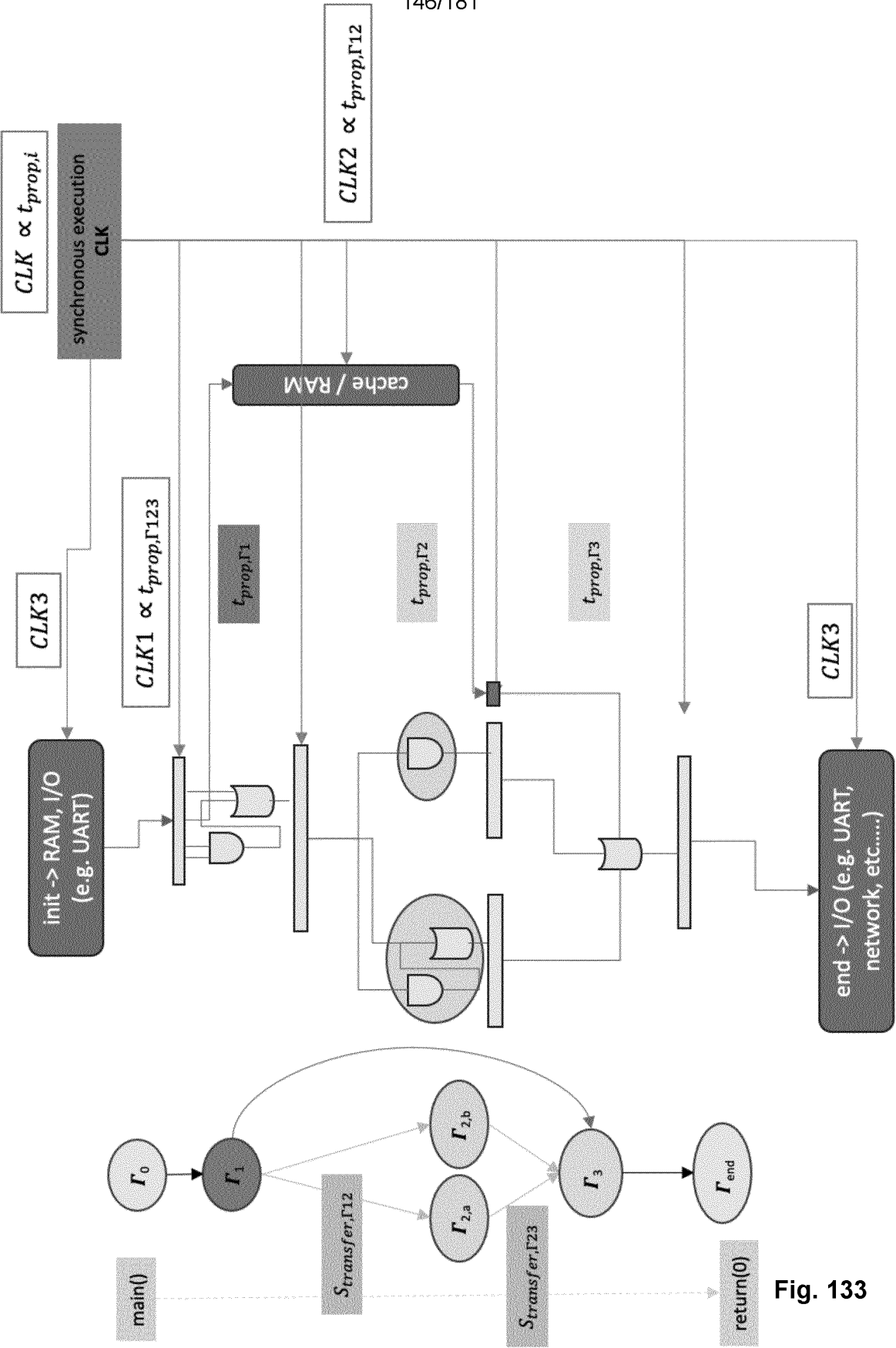
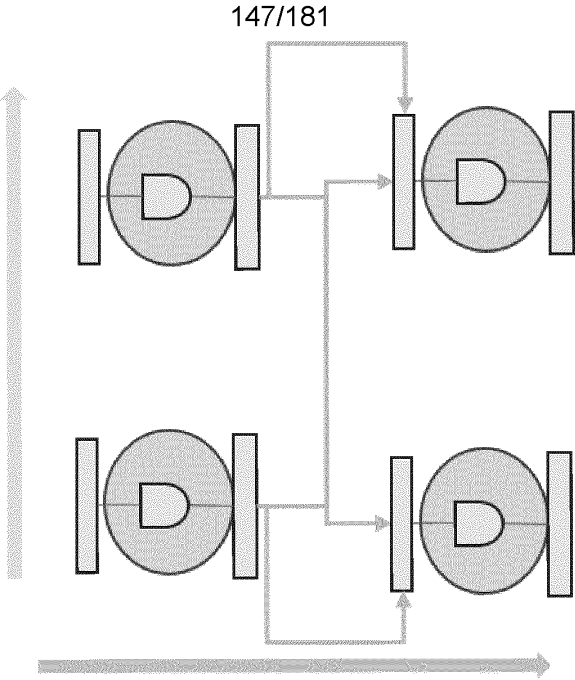


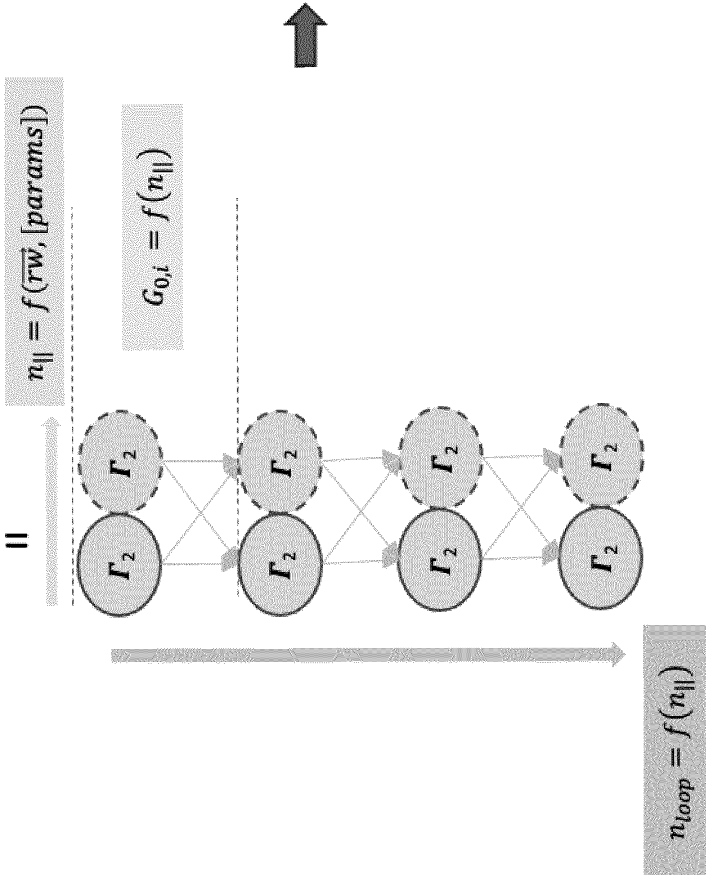
Fig. 133

RTL definition



(b)

Gamma-Graph



(a)

Fig. 134

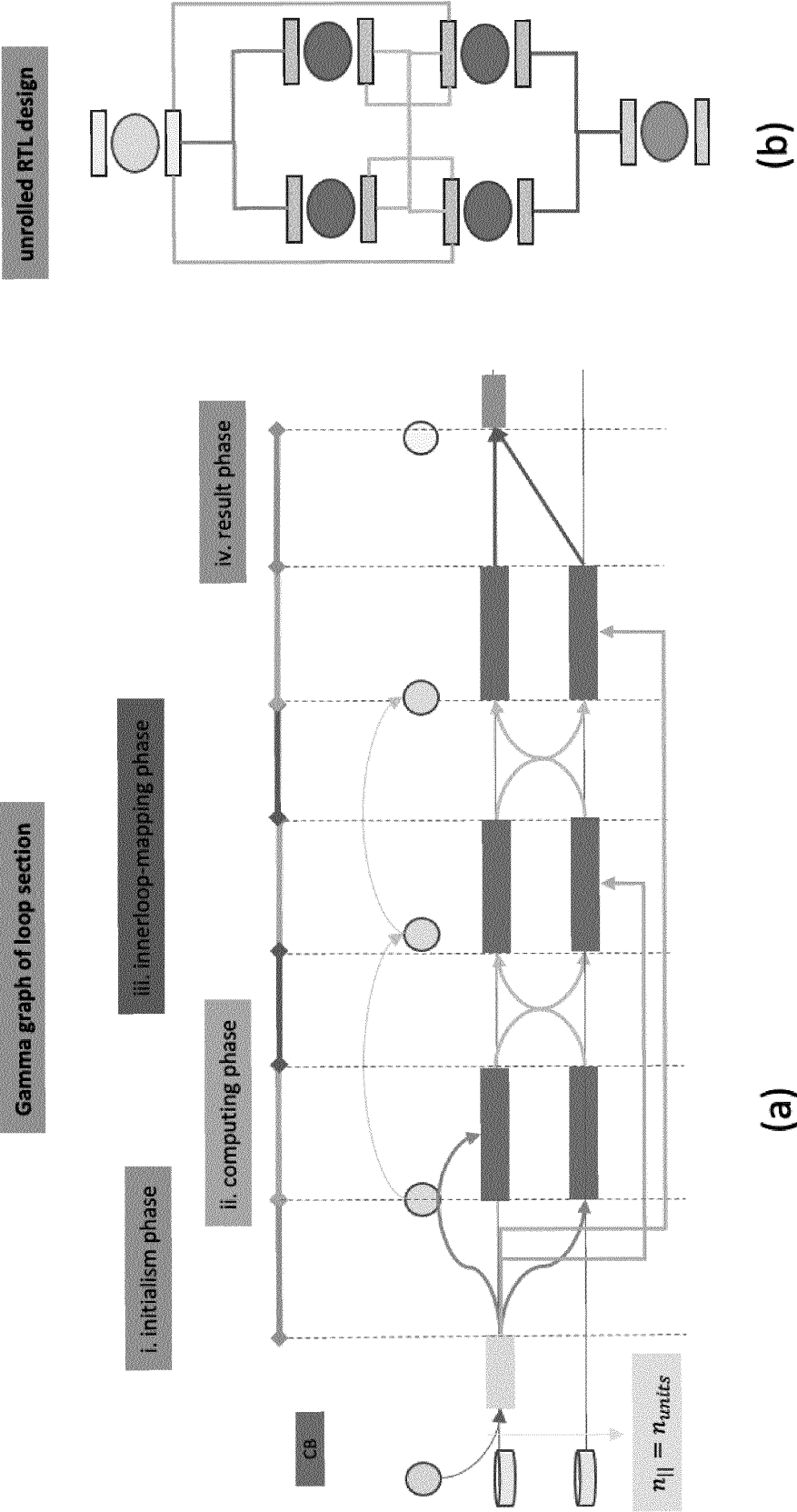


Fig. 135

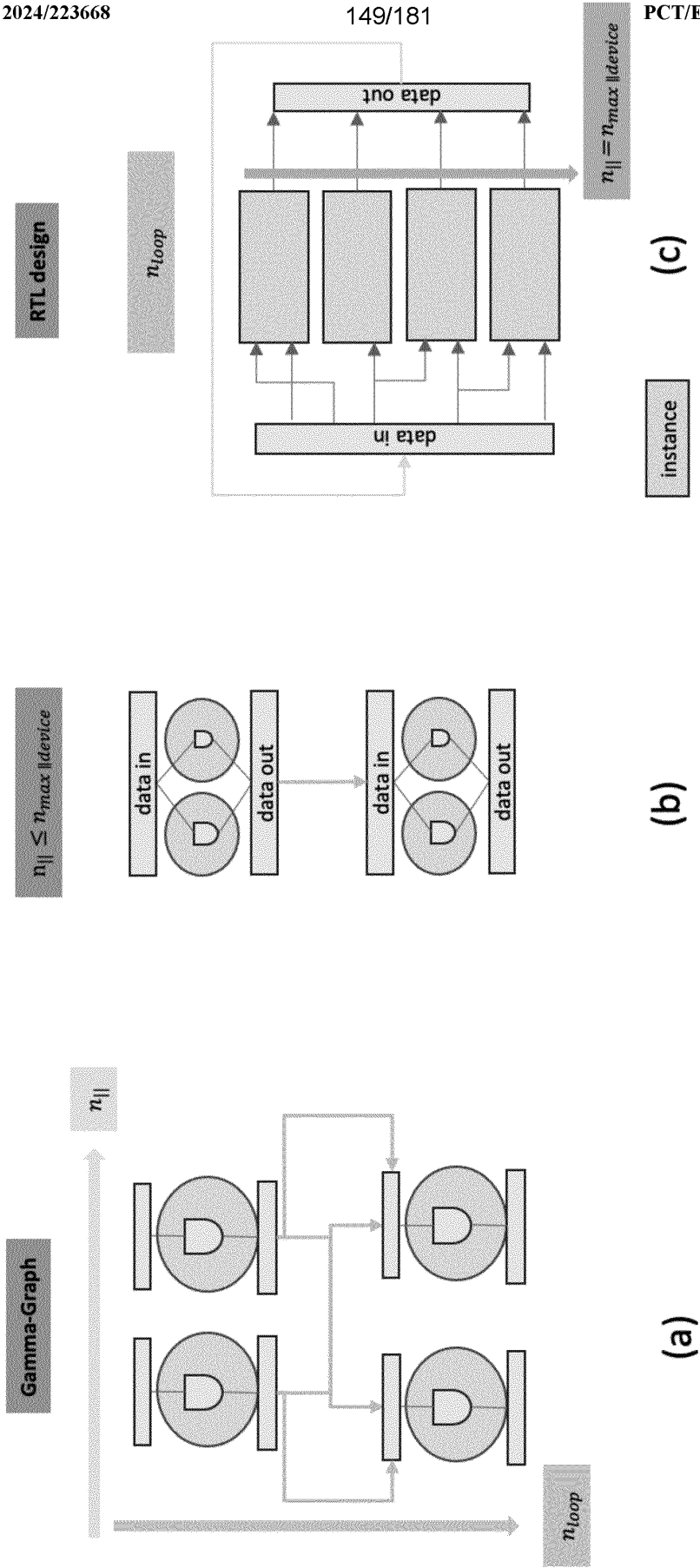
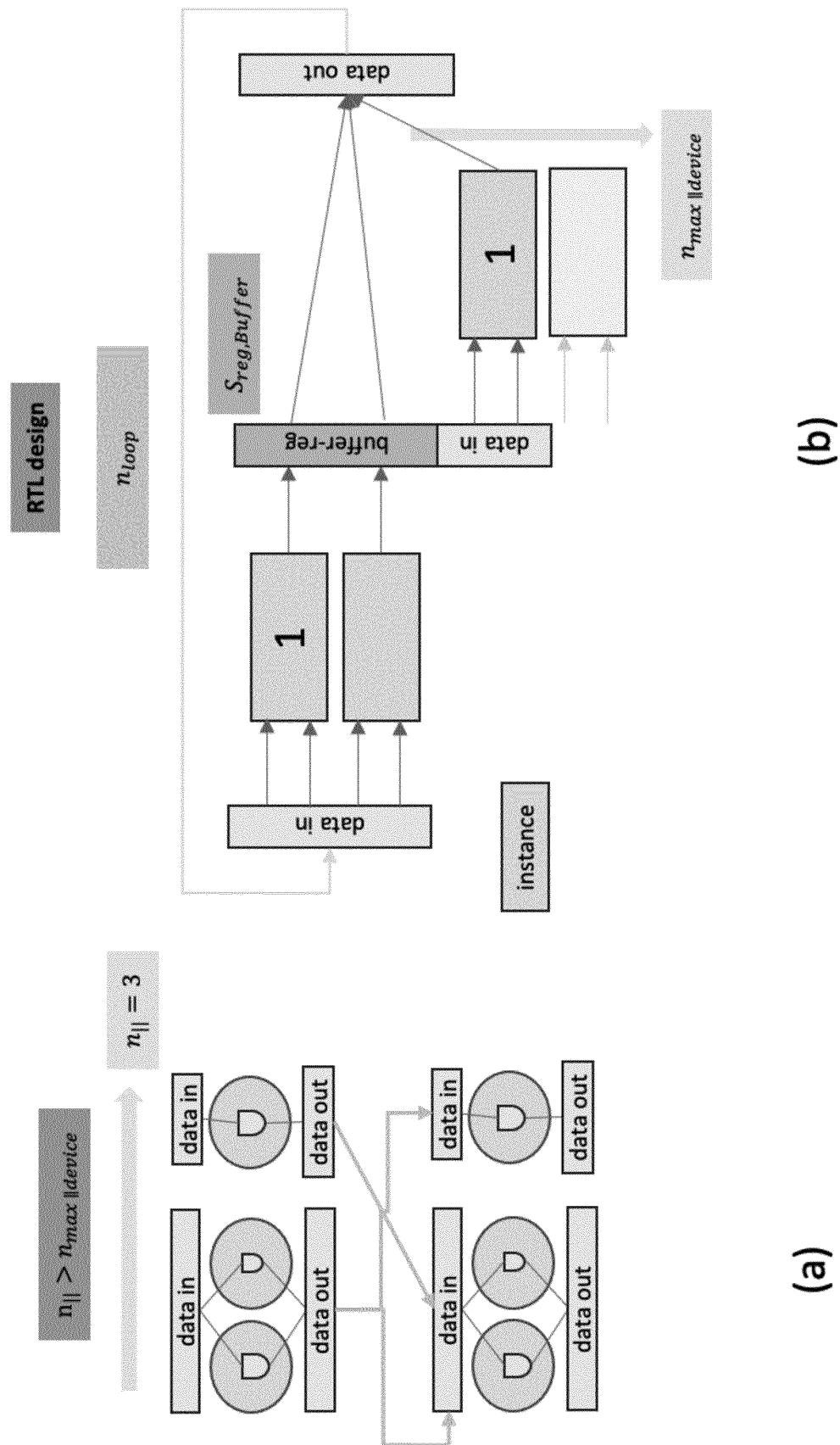


Fig. 136



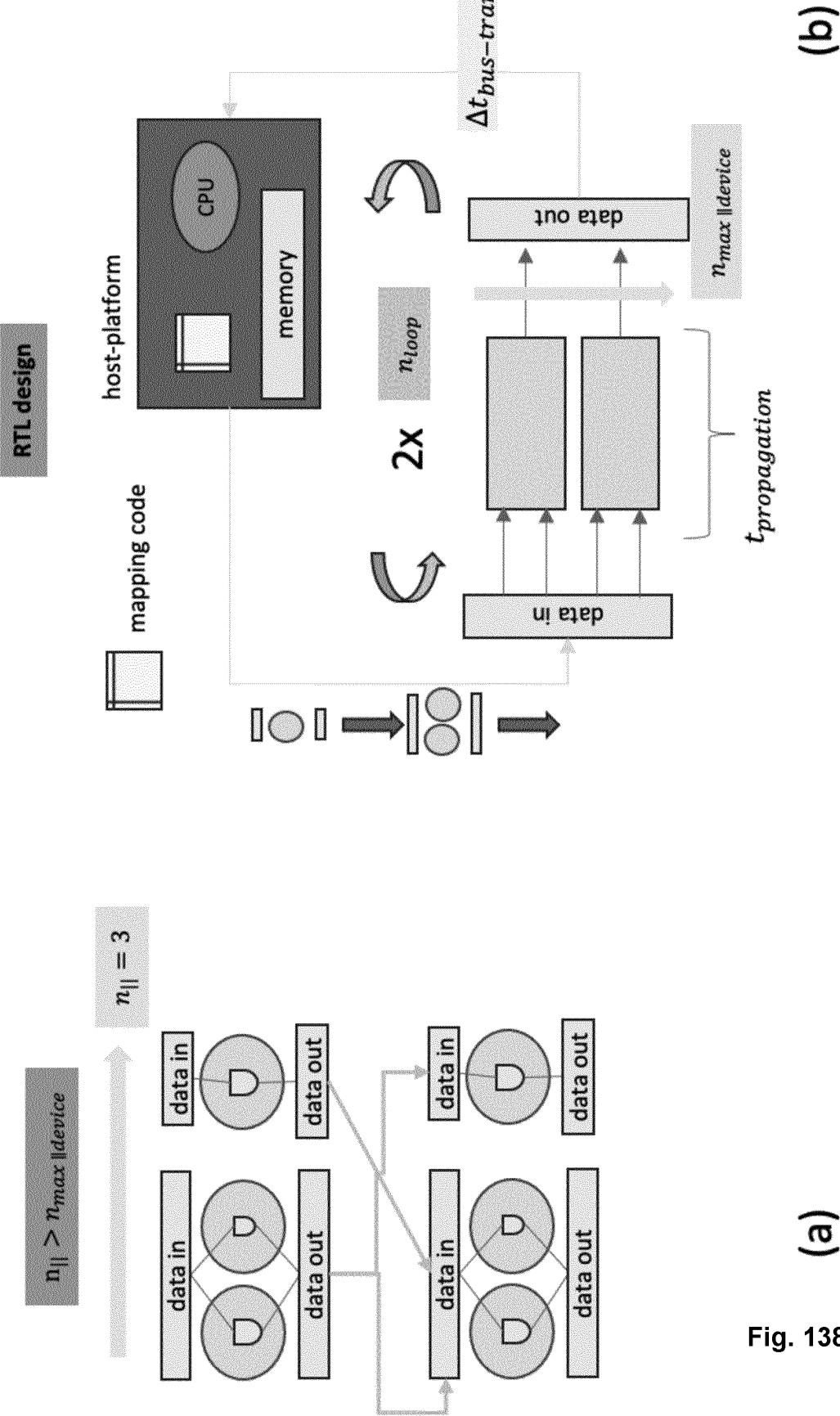


Fig. 138

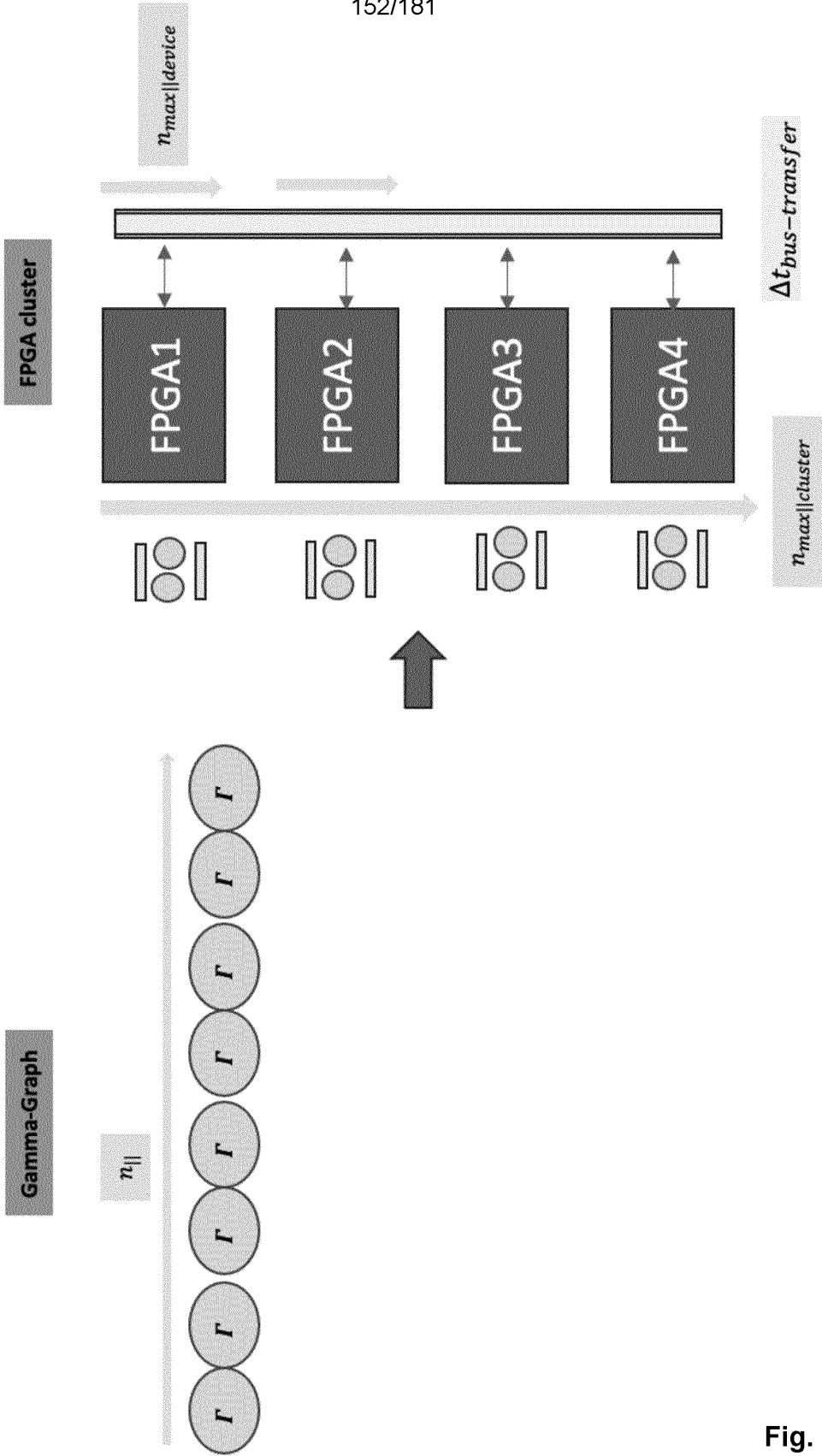


Fig. 139

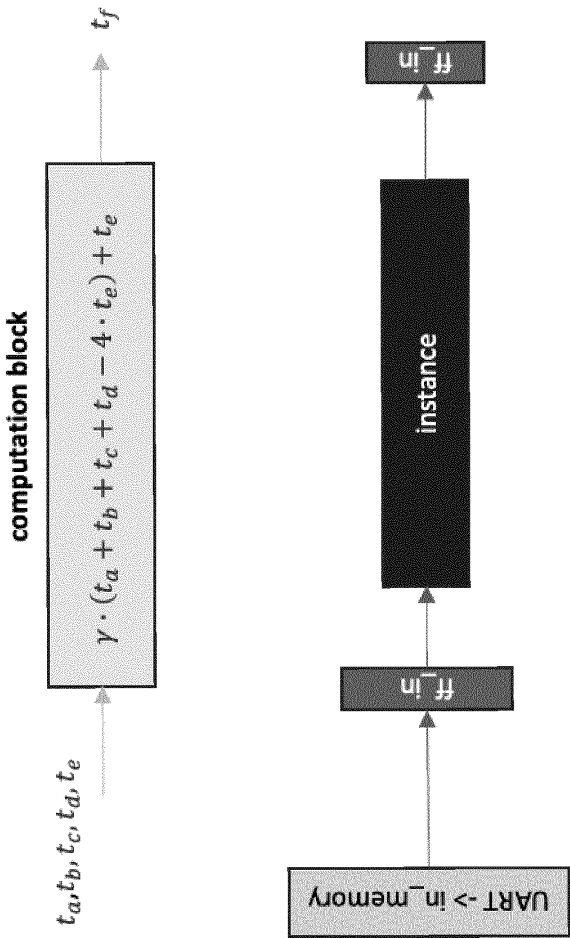


Fig. 140

```
flipflop #(NBDN, NIN, NOUT) flipflop0 (  
    .clk(CLK),  
    .copy_flag(copy1_flag),  
    .in_memory({}),  
    .ff_in({}),  
    .ff_out({}),  
    .copy_state(copy1_state)  
);
```

```

`ifndef _compute_v_
`define _compute_v_

module compute (
    input wire clk,
    input wire signed [8:1:0] ta, tb, tc, td, te,
    output wire signed [8:1:0] tf
);

    // Intermediate signals
    wire [8:1:0] adders [3:0]; // 4 adders
    wire [8*2-1:0] multi, multi2, multigamma;

    // Adders
    assign adders[0] = ta+tb;
    assign adders[1] = tc+td;
    assign adders[2] = adders[0] + adders[1];
    assign multi = te * 16'b000000000100_0000; // 4*te
    assign multi2 = (~multi)+16'b1;
    assign adders[3] = adders[2] + multi2[11:4];
    assign multigamma = 16'b000000000000_0100 * {8'b00000000, adders[3]};
    assign tf = multigamma[11:4] + te;

endmodule

`endif // _compute_v_

```

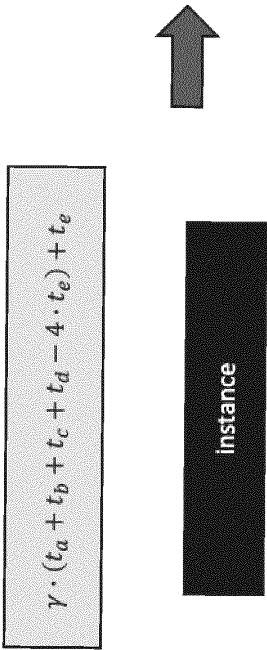


Fig. 141

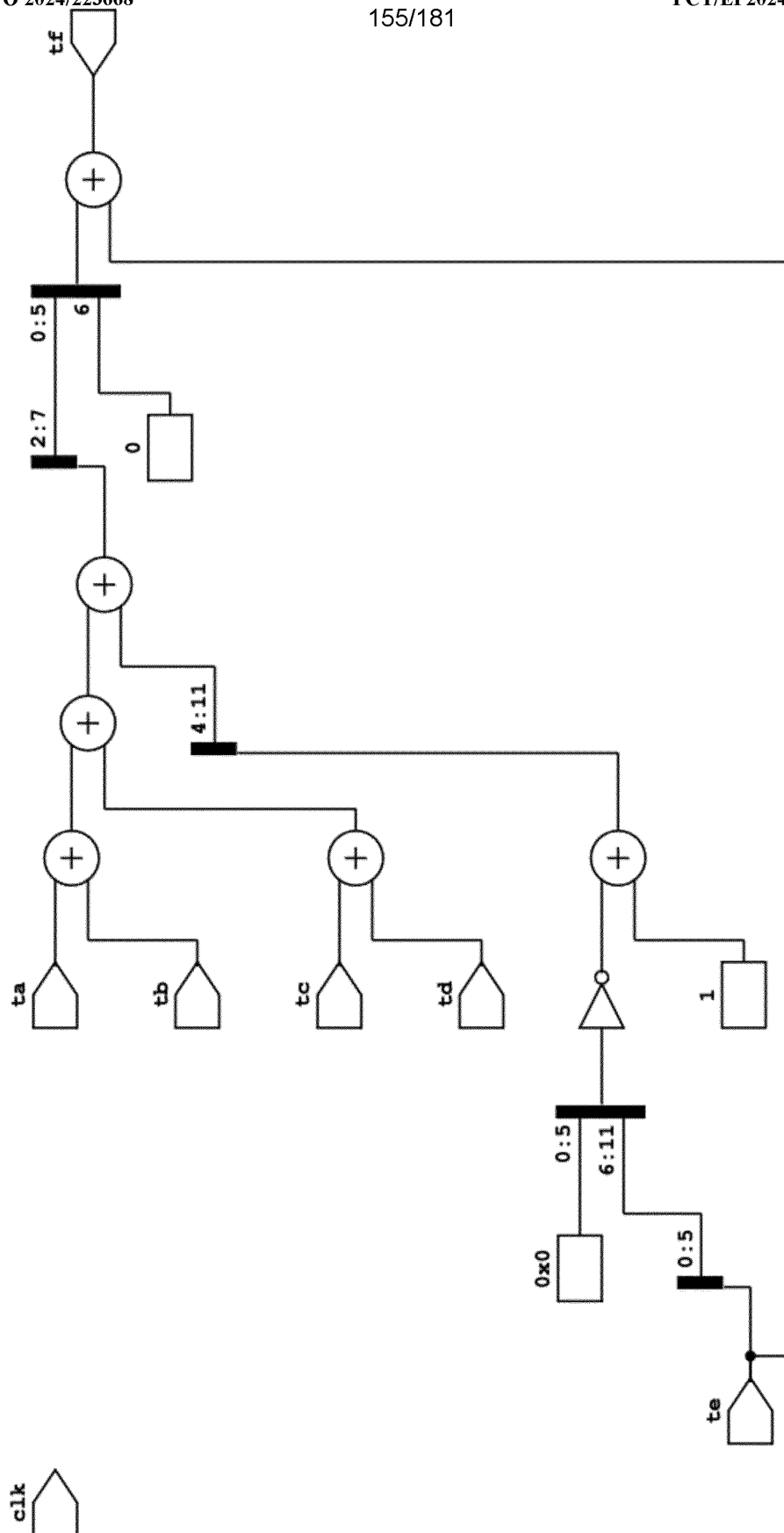


Fig. 142

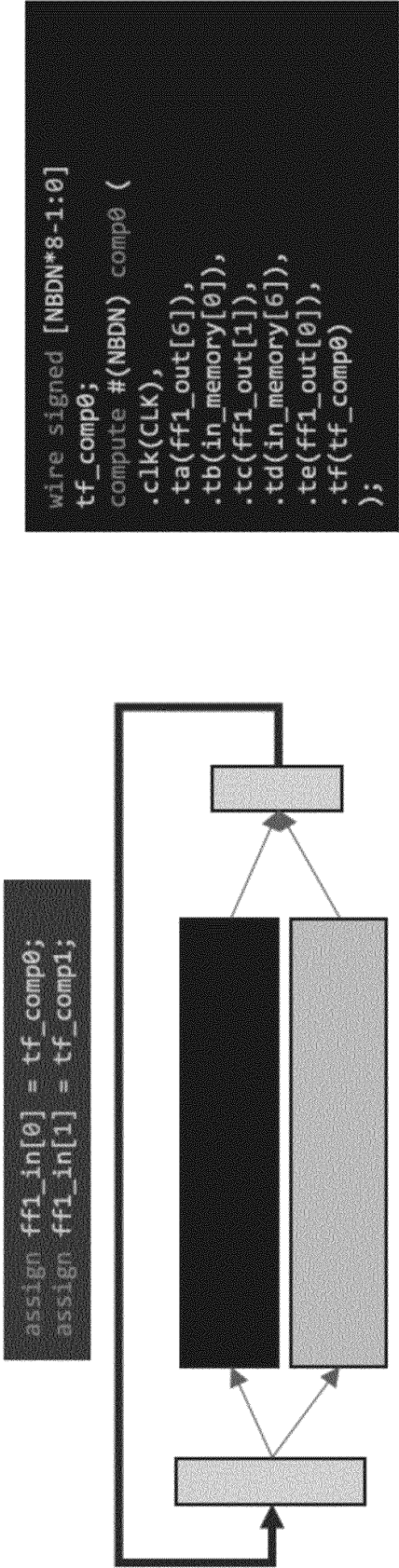


Fig. 143

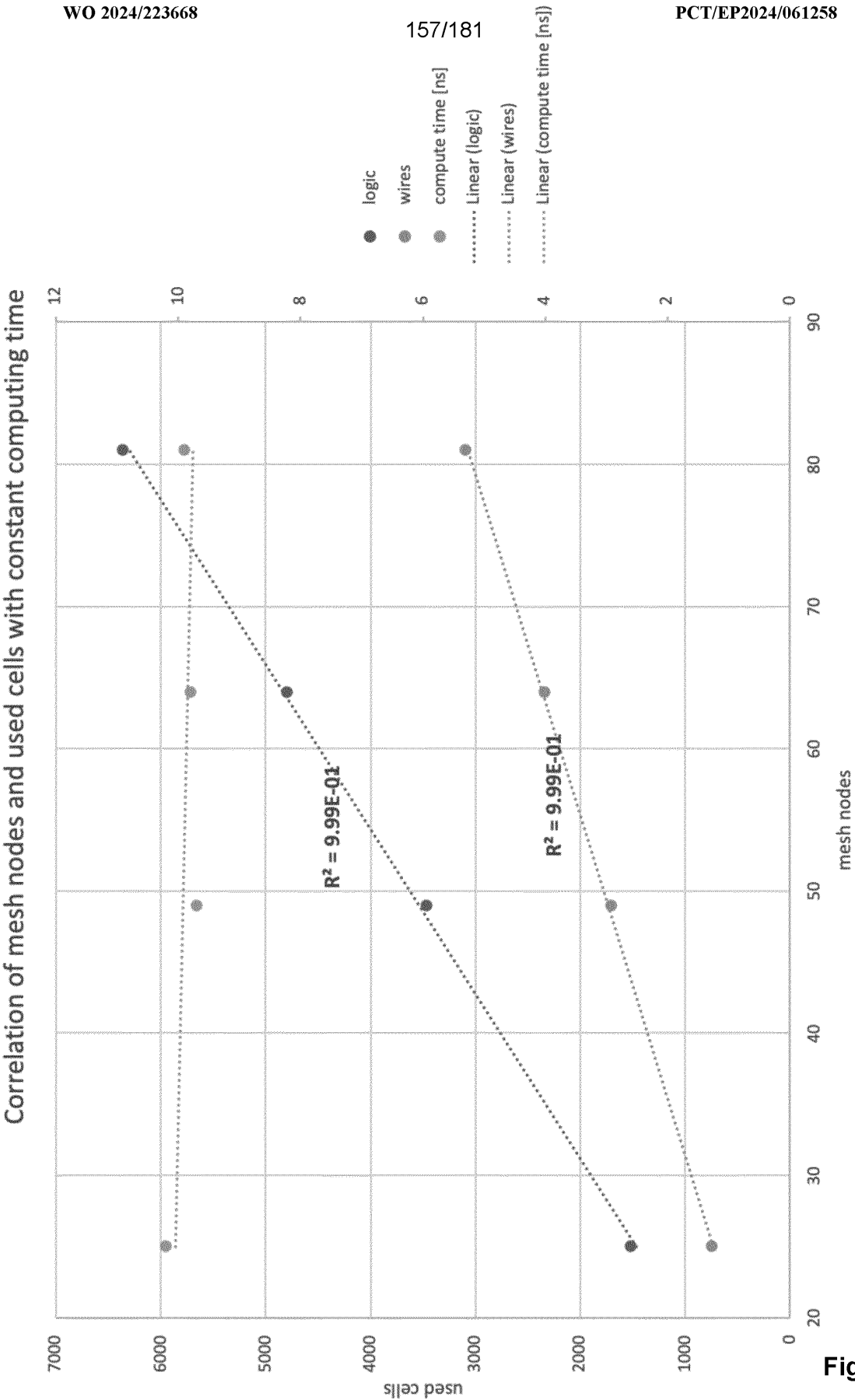


Fig. 144

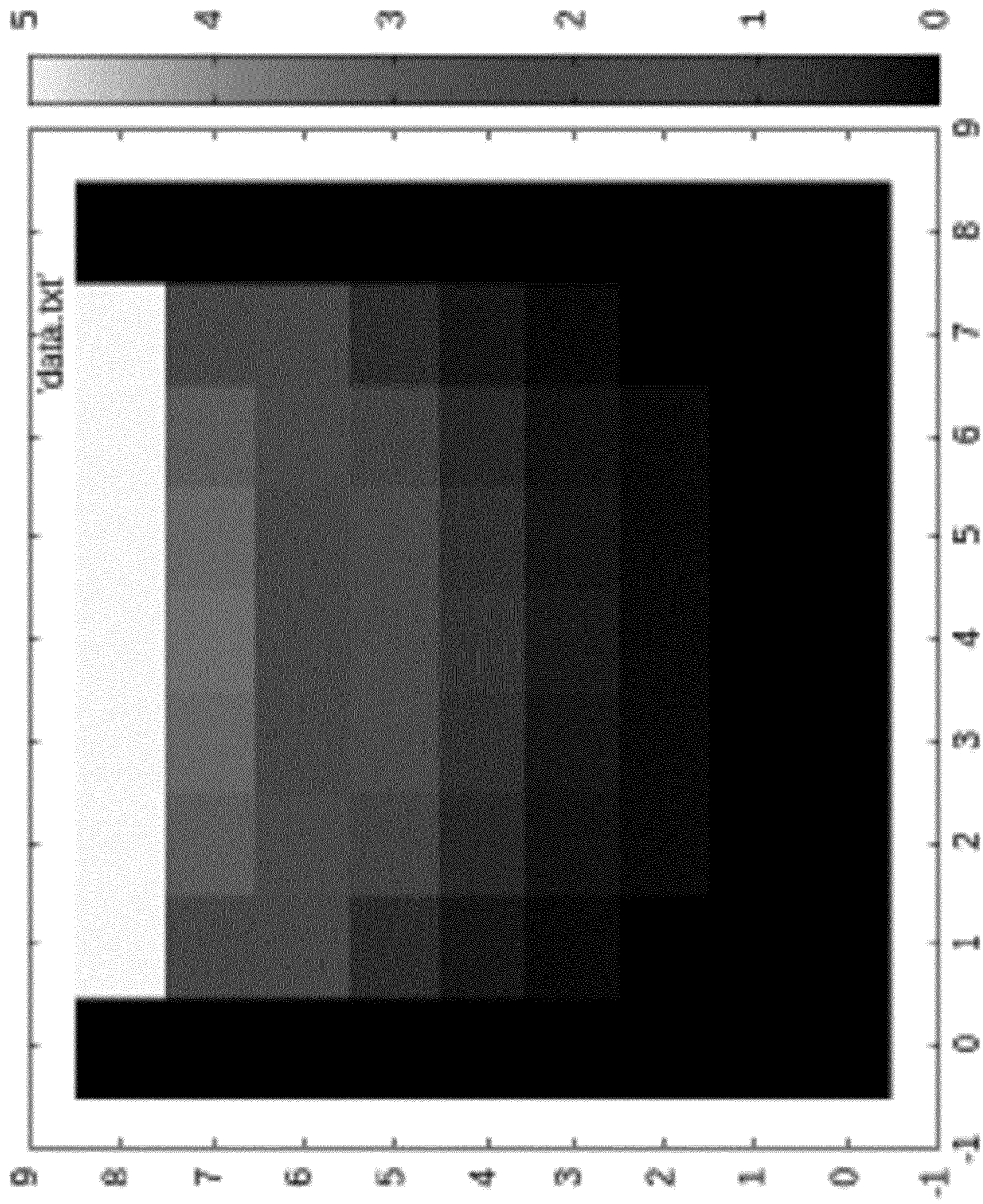


Fig. 145

Field	35 or 43	rs	Rt	Address
Bit-position	31:26	25:21	20:16	15:0
		\$t2	\$t1	Offset

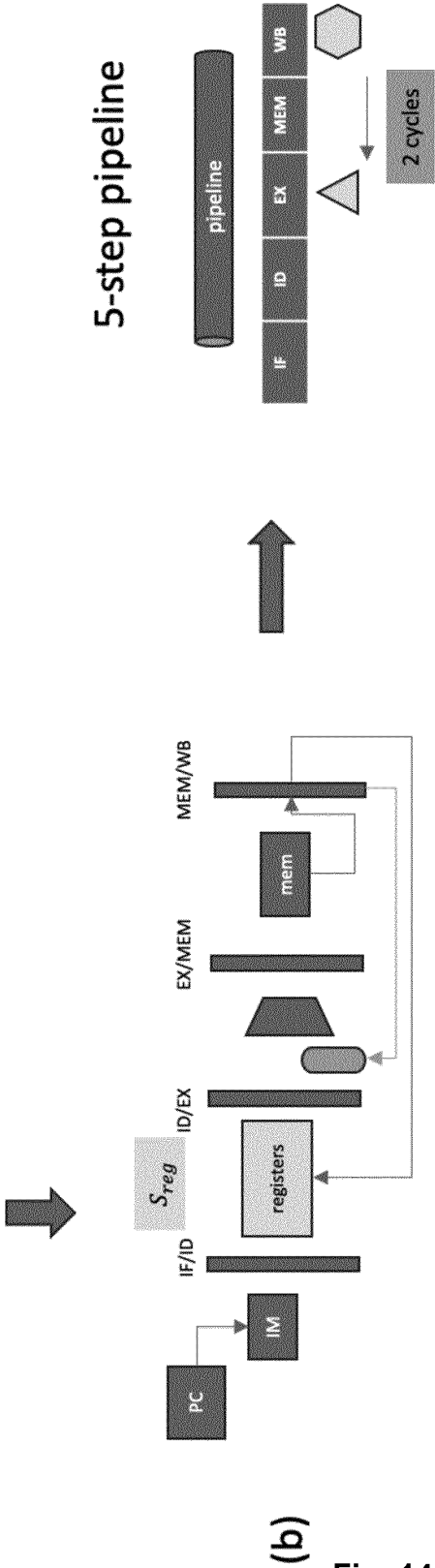
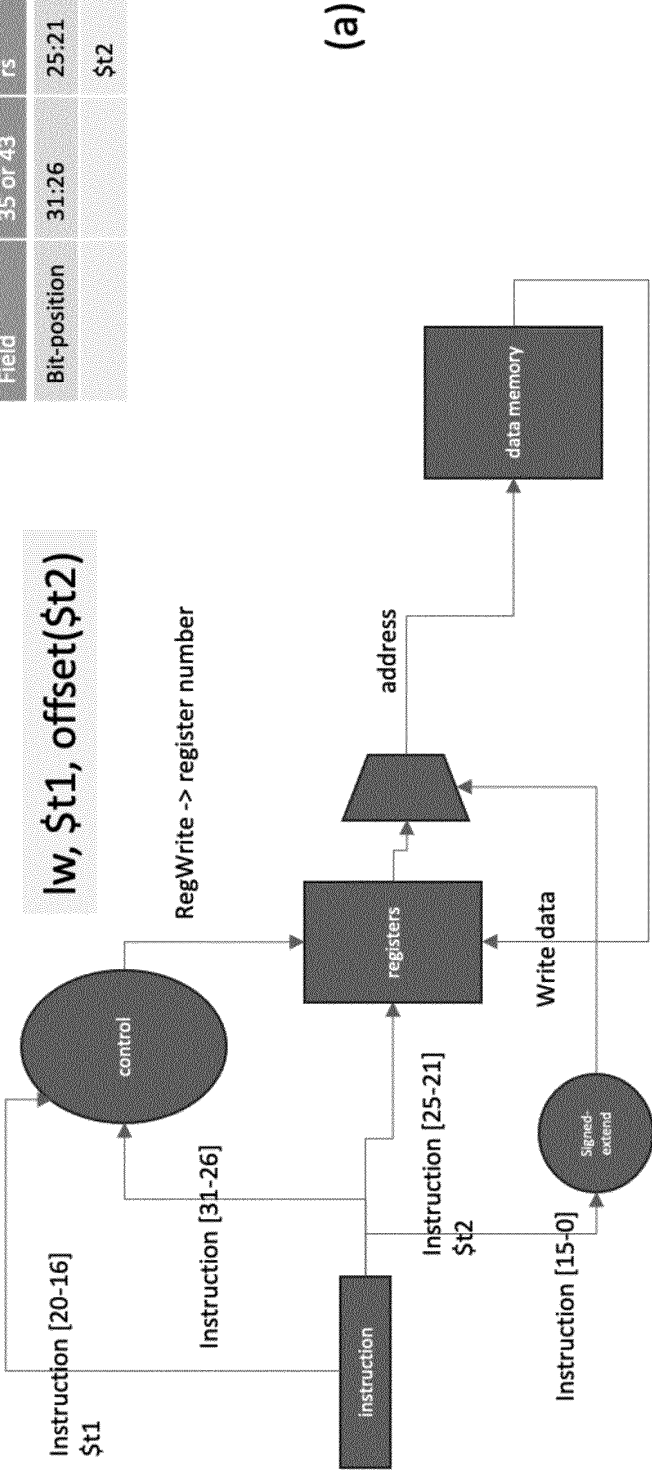
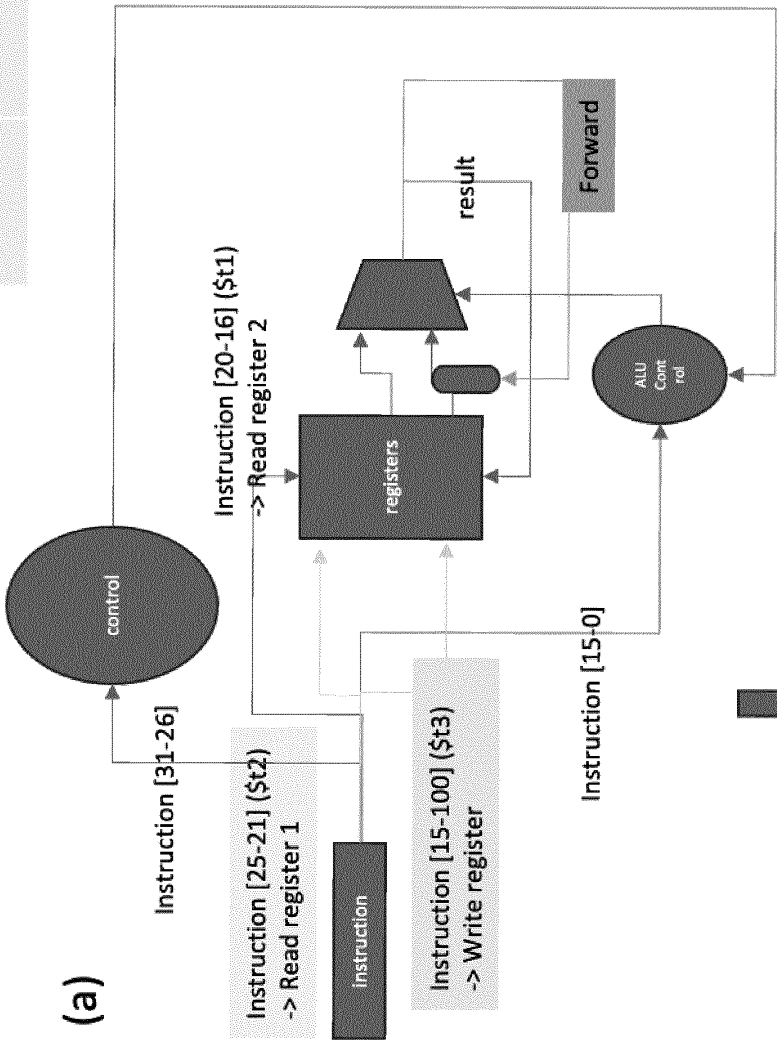


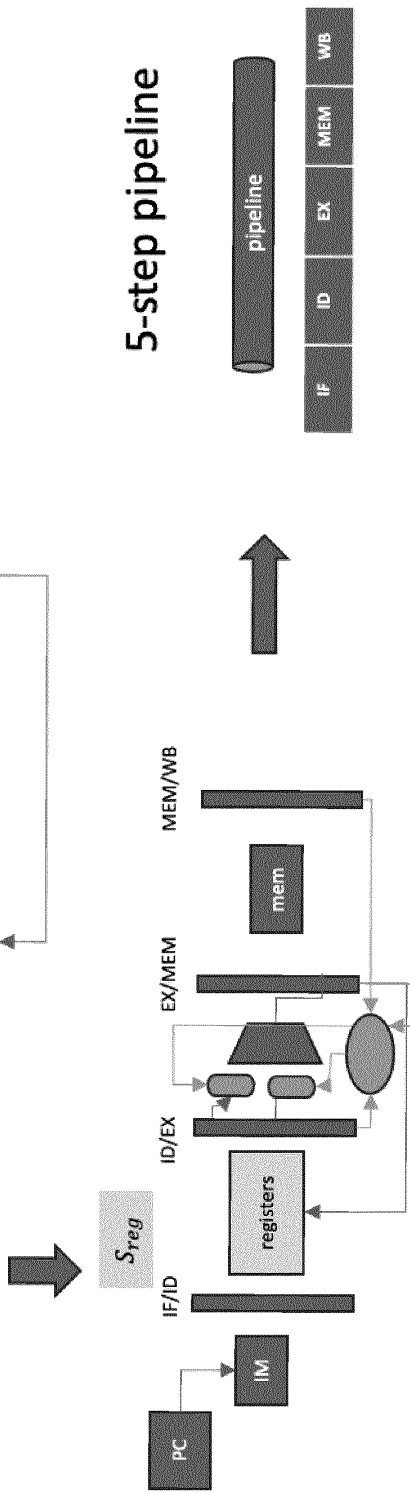
Fig. 146

Field	0	rs	rt	rd	shamt	Funct
Bit-position	31:26	25:21	20:16	15:11	10:6	5:0
		\$t2	\$t3	\$t1, dest		

add, \$t1, \$t2, \$t3



Forward Unit



5-step pipeline

Fig. 147

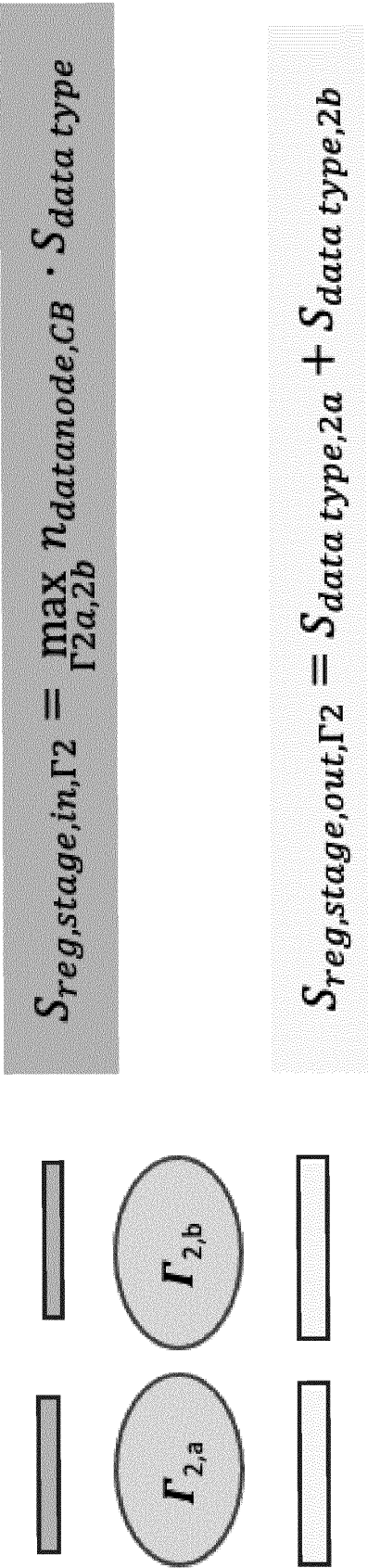


Fig. 148

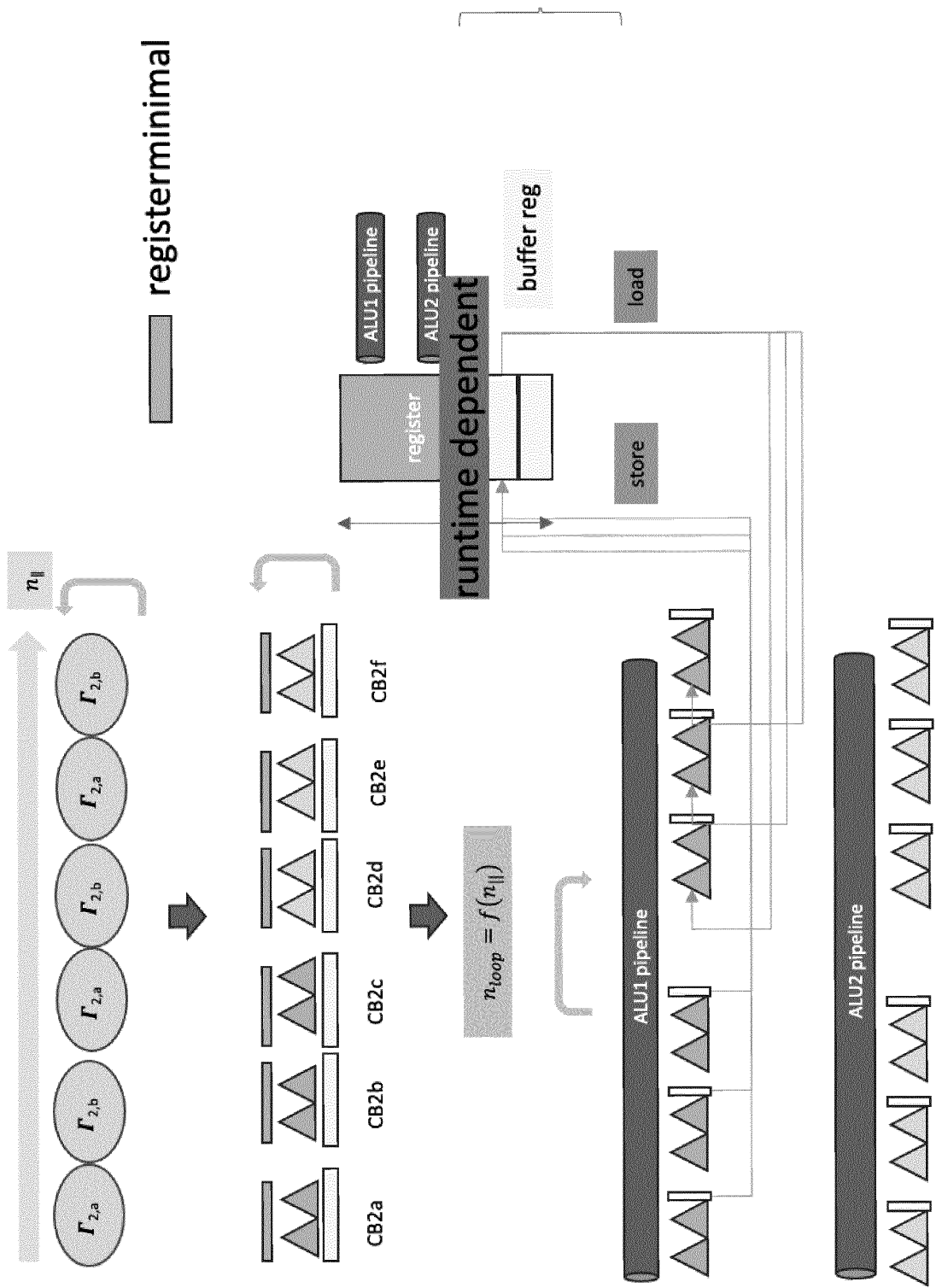


Fig. 149

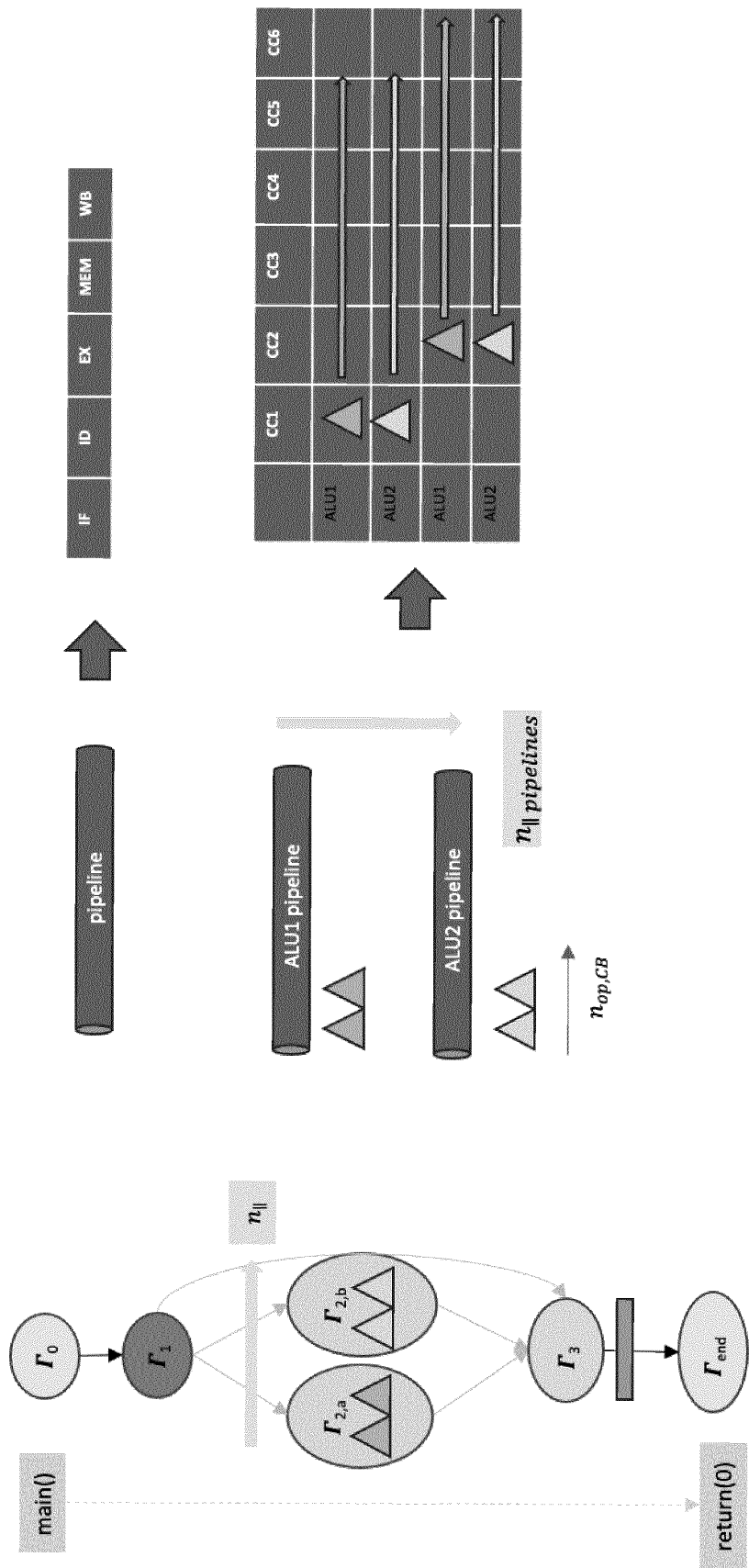


Fig. 150

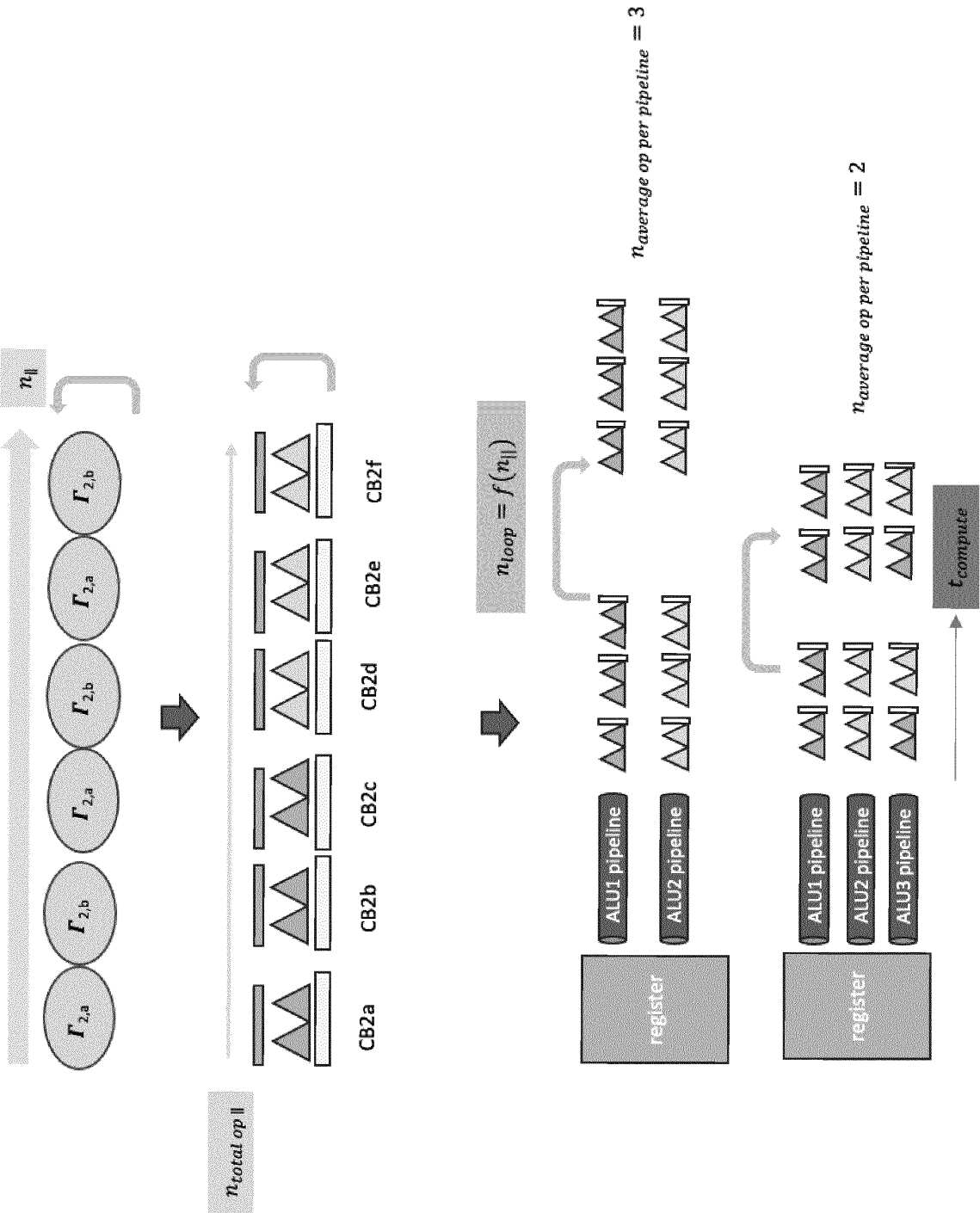


Fig. 151

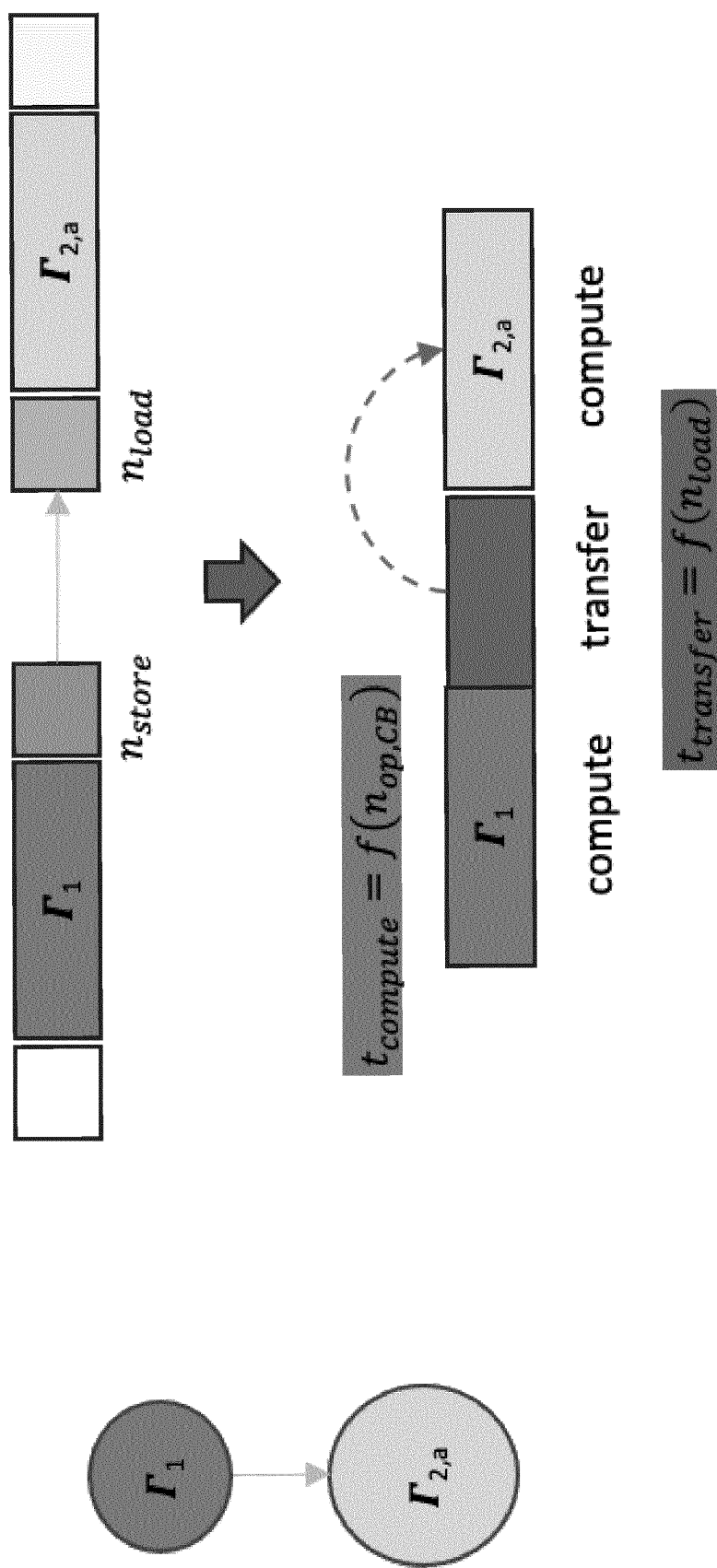


Fig. 152

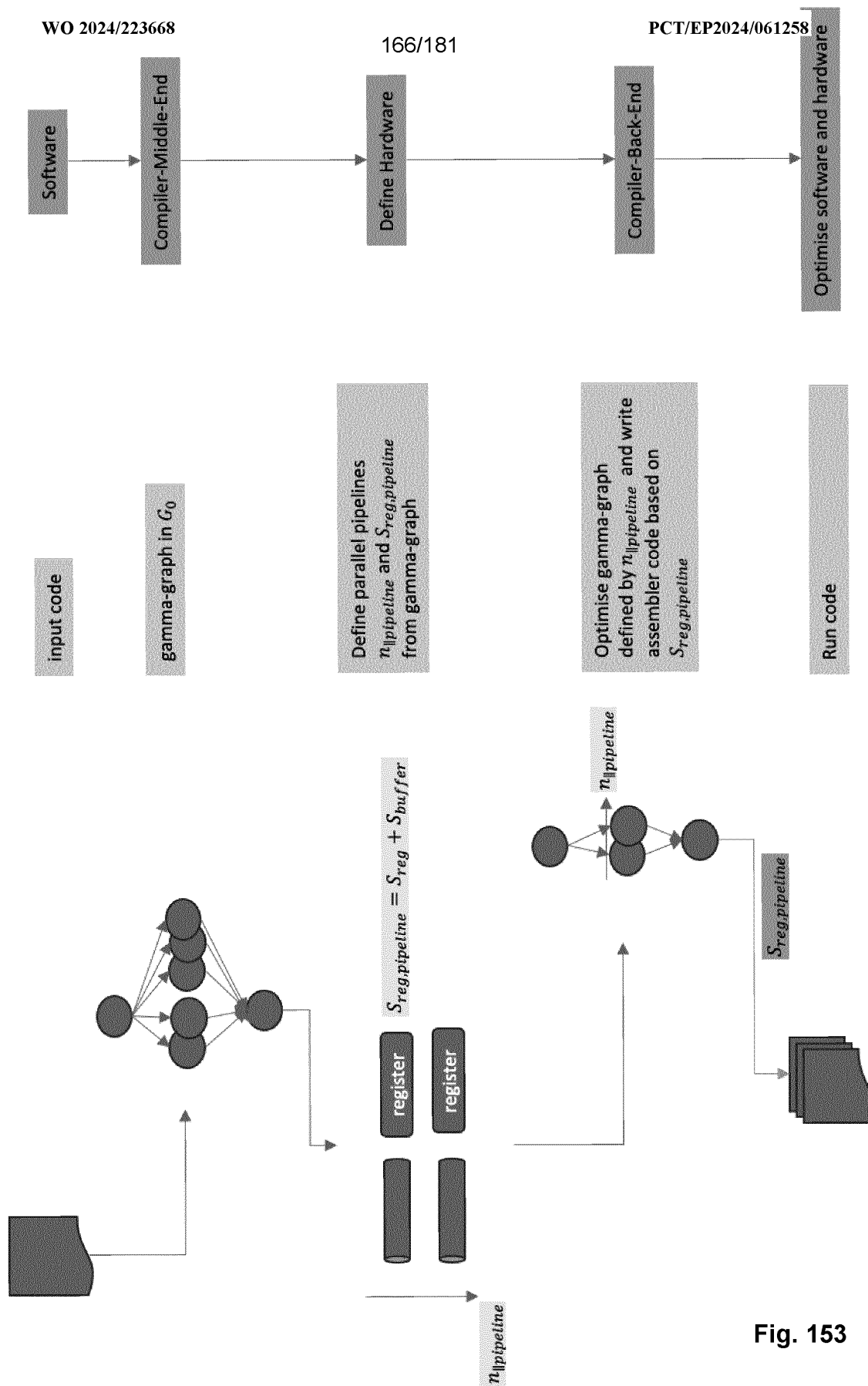


Fig. 153

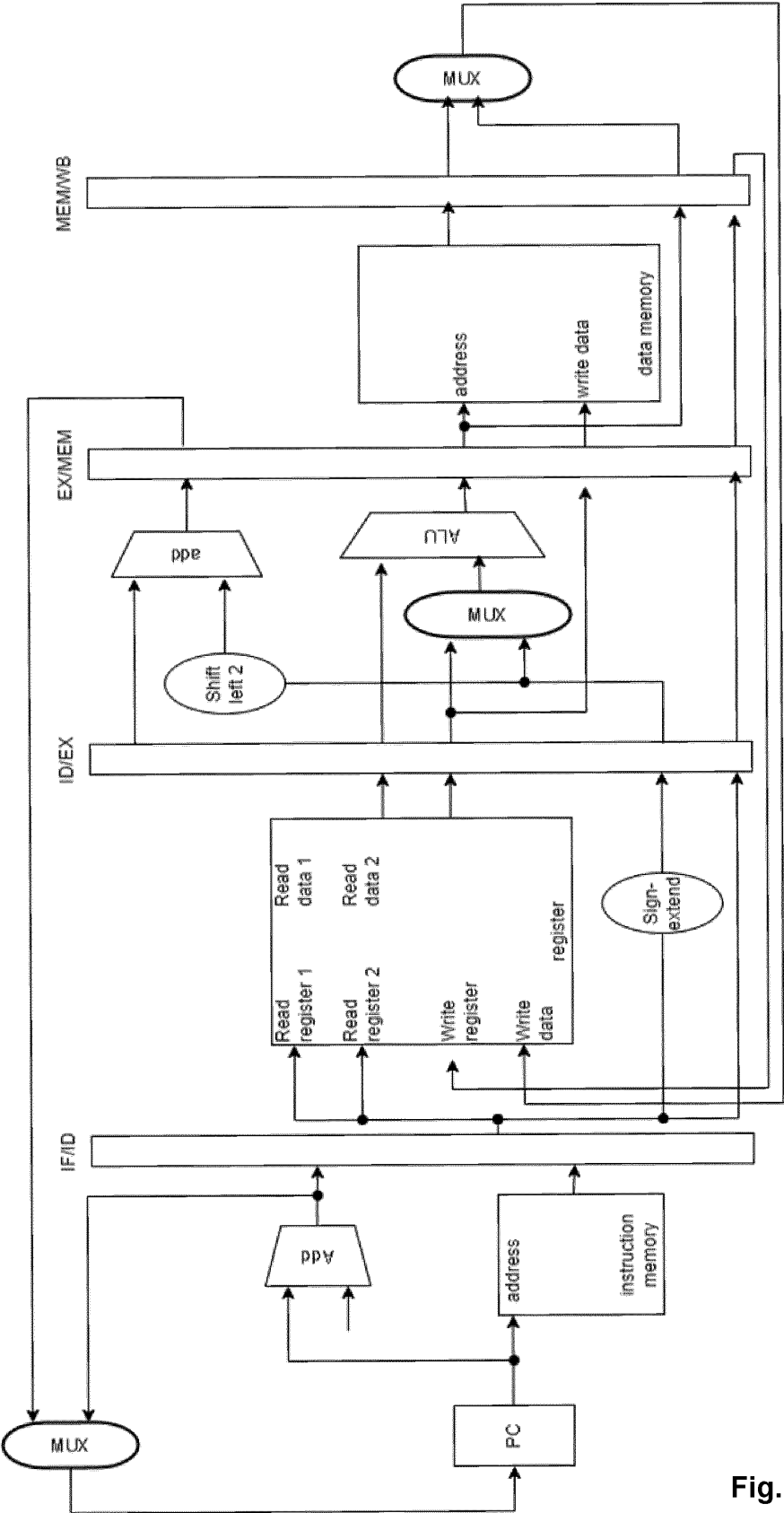


Fig. 155

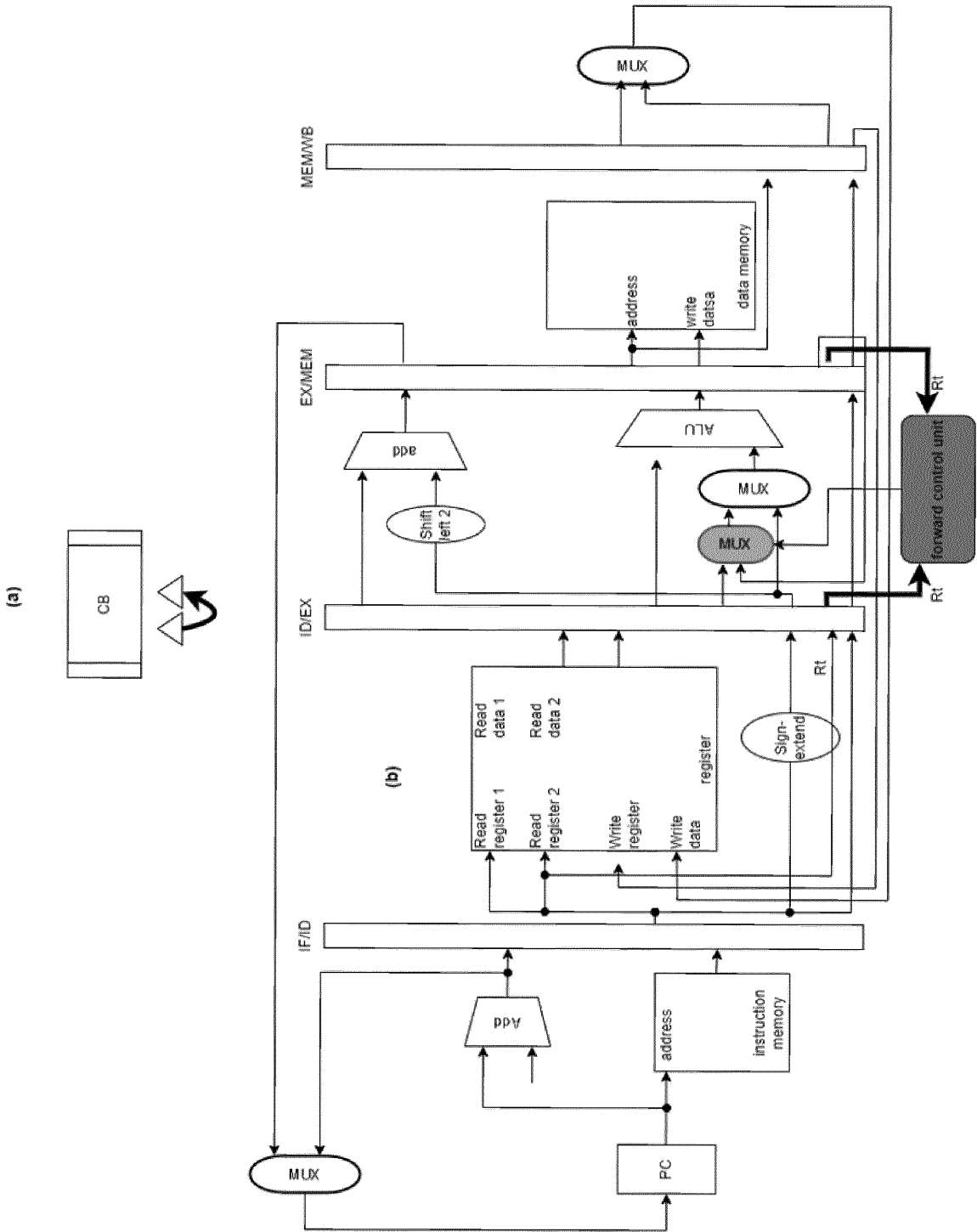


Fig. 156

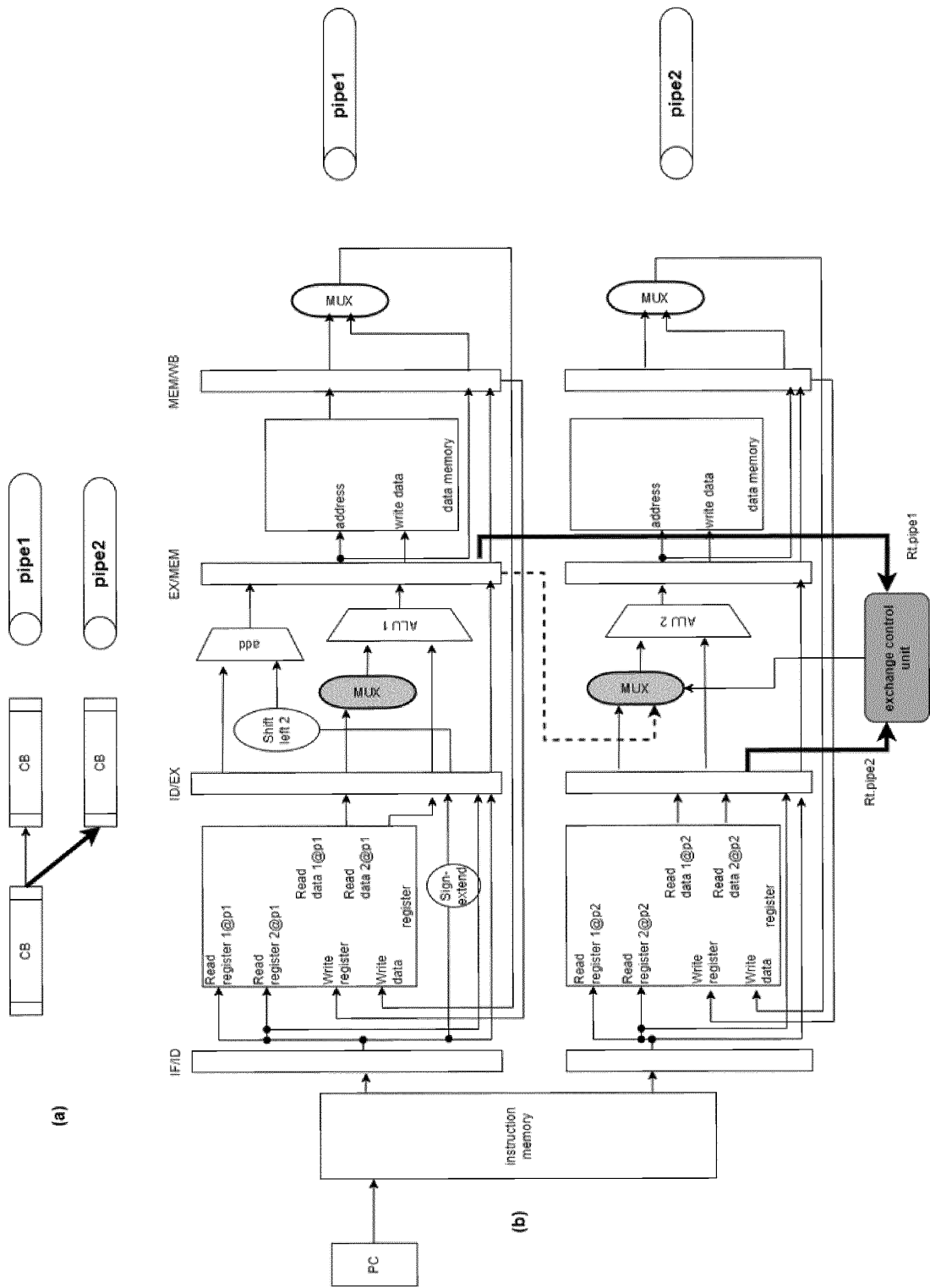


Fig. 157

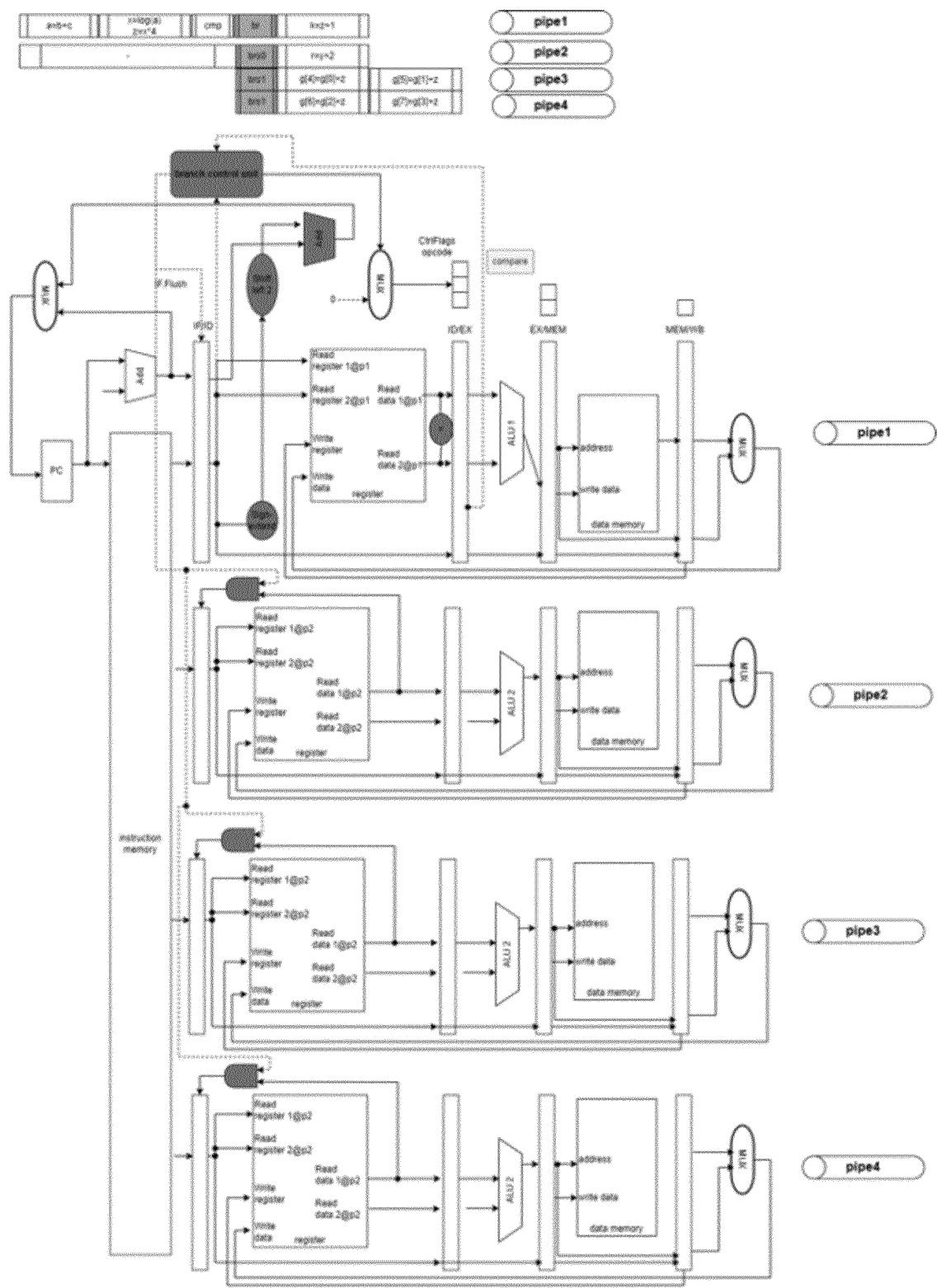


Fig. 158

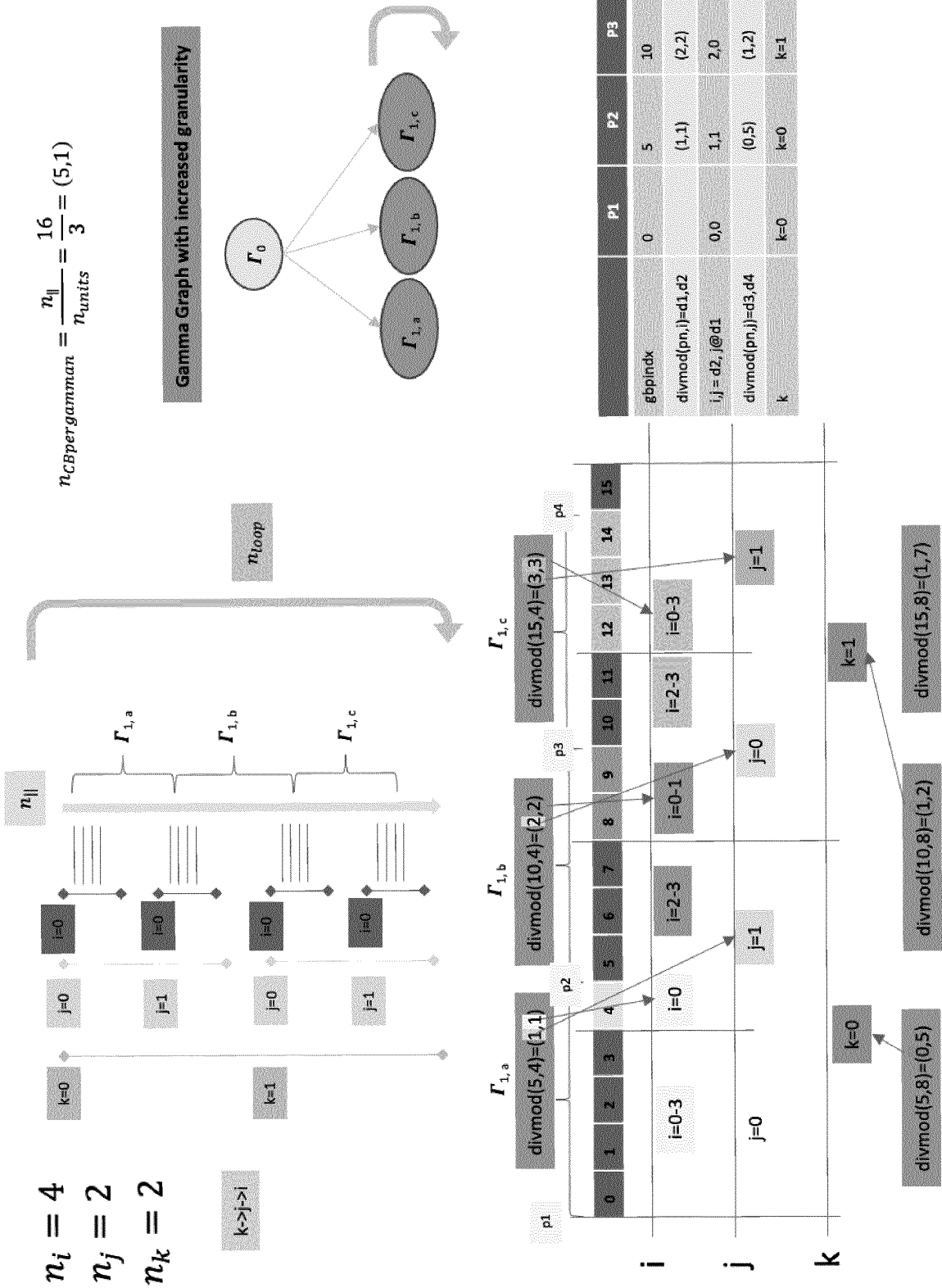


Fig. 159

```
#include "stdio.h"
#include "stdint.h"
#include "math.h"

#define N 8
int main(int argc, char *argv[]) {
    uint8_t a, b, c, x, y, z, r, k;
    int g[N];
    a = b + c;
    x = log(a);
    y = a*a;
    z = x*4;
    if (z < 5) {
        for (int i=0; i<N-4; i++) {
            g[i+4] = g[i]+z;
        }
        k = g[N-1]*g[N-2];
        r = y+4;
    } else {
        r = y+2;
        k = z+1;
    }
    printf("%d %d", r, k);
    return(1);
}
```

Fig. 160

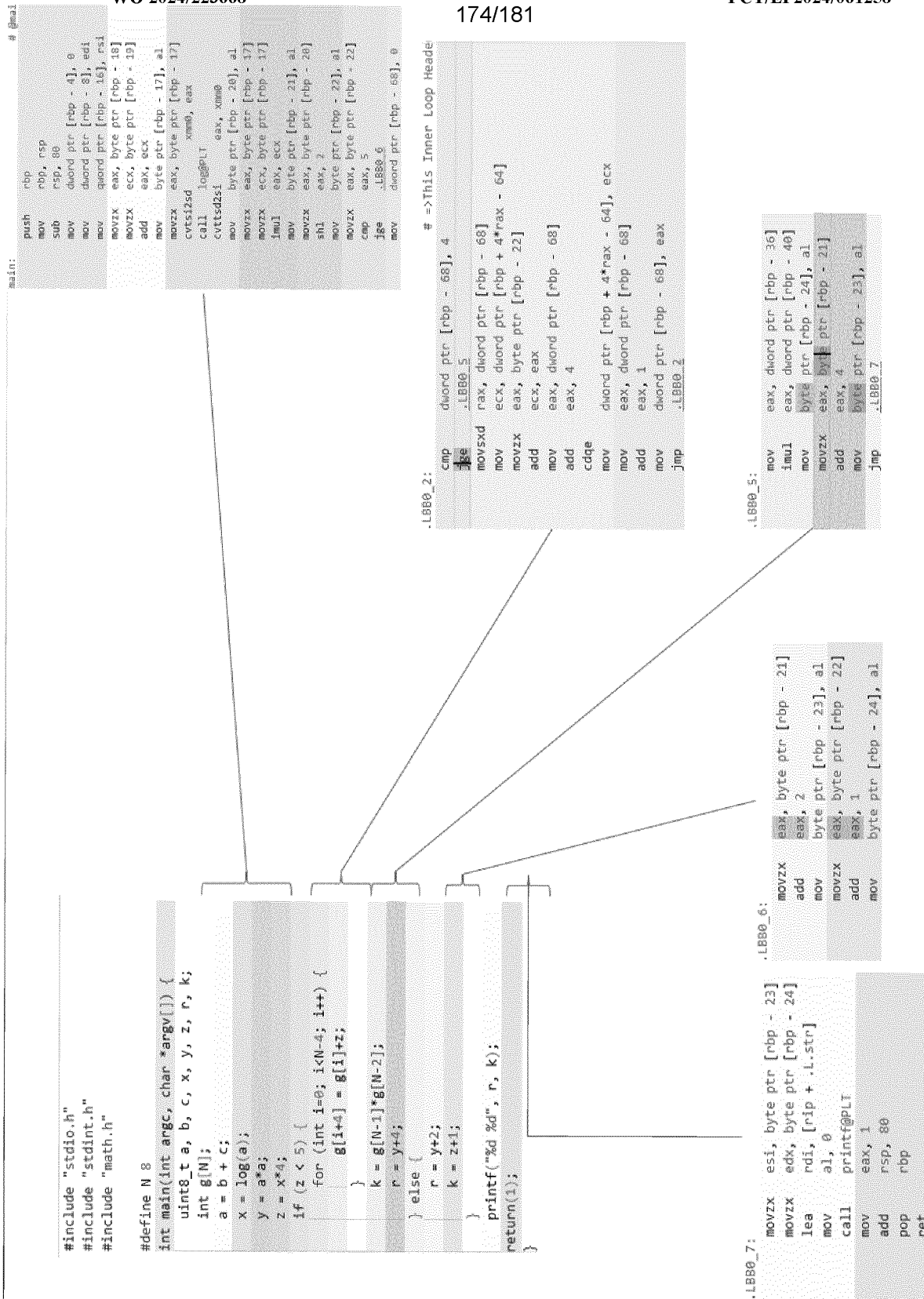


Fig. 161

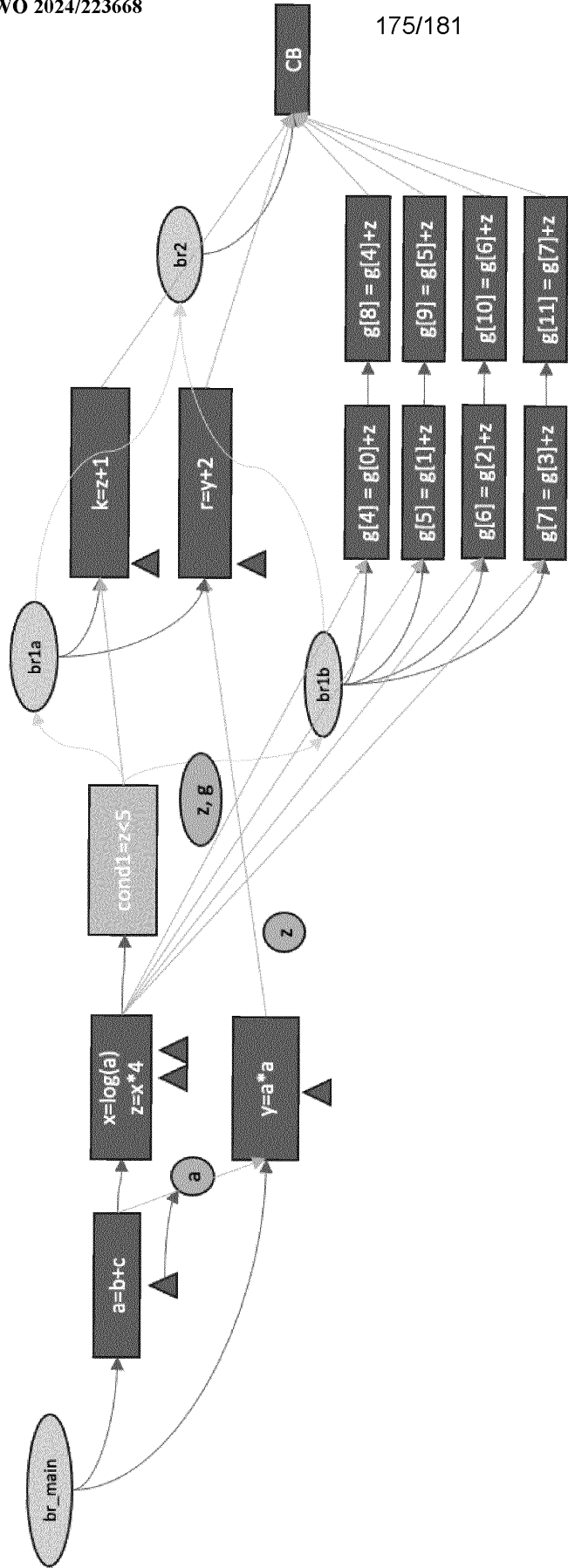


Fig. 162

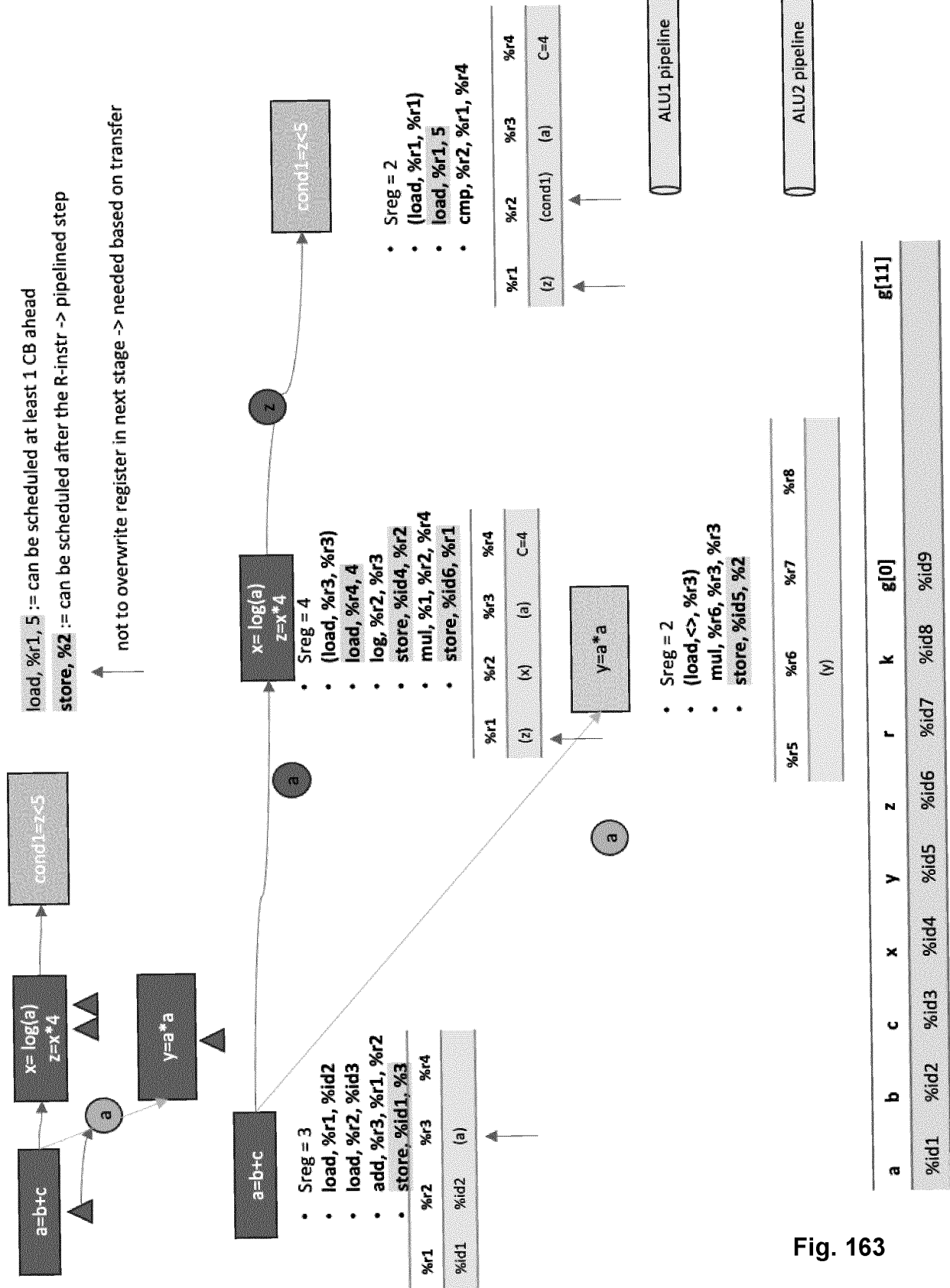


Fig. 163

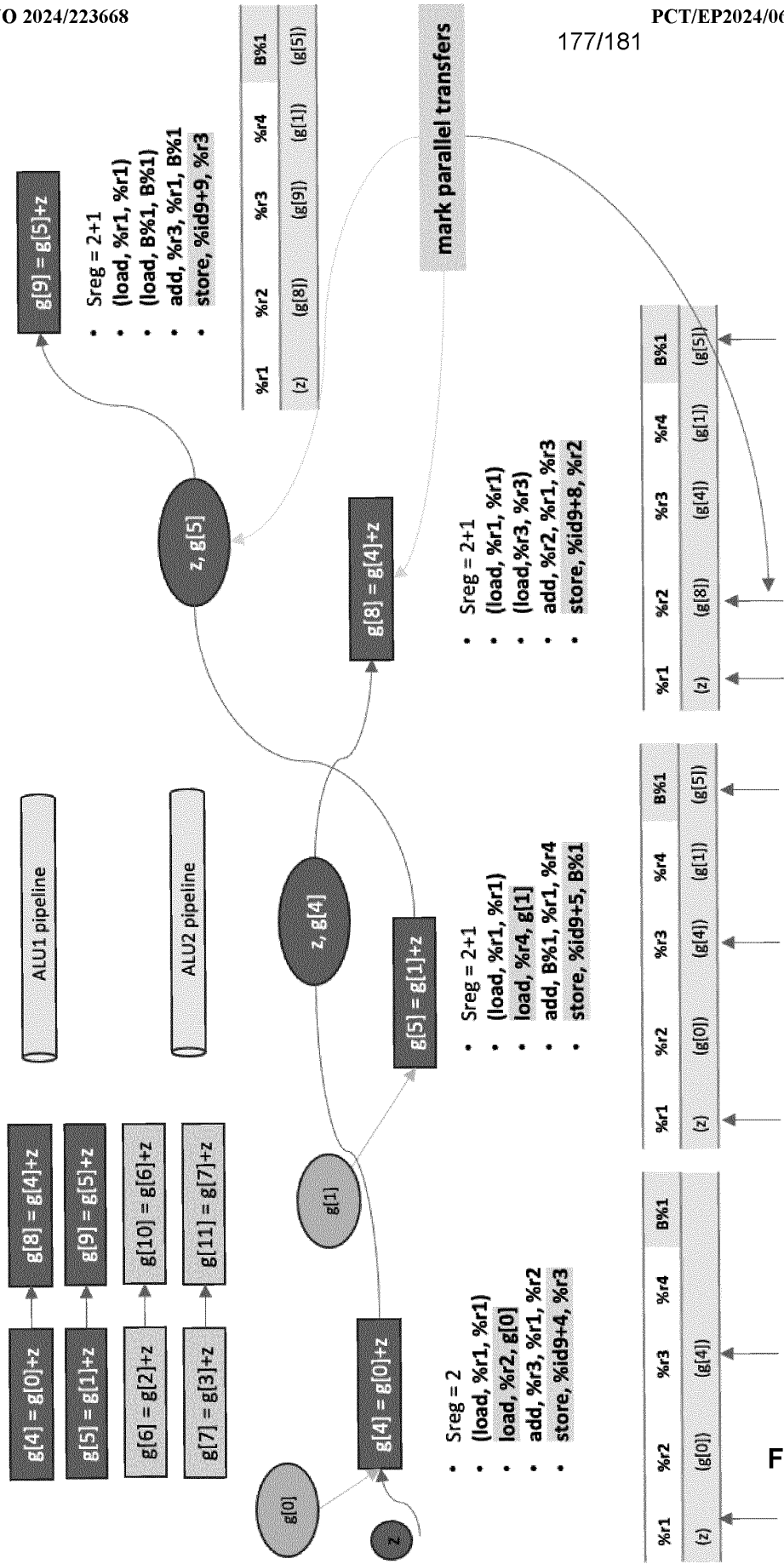


Fig. 164

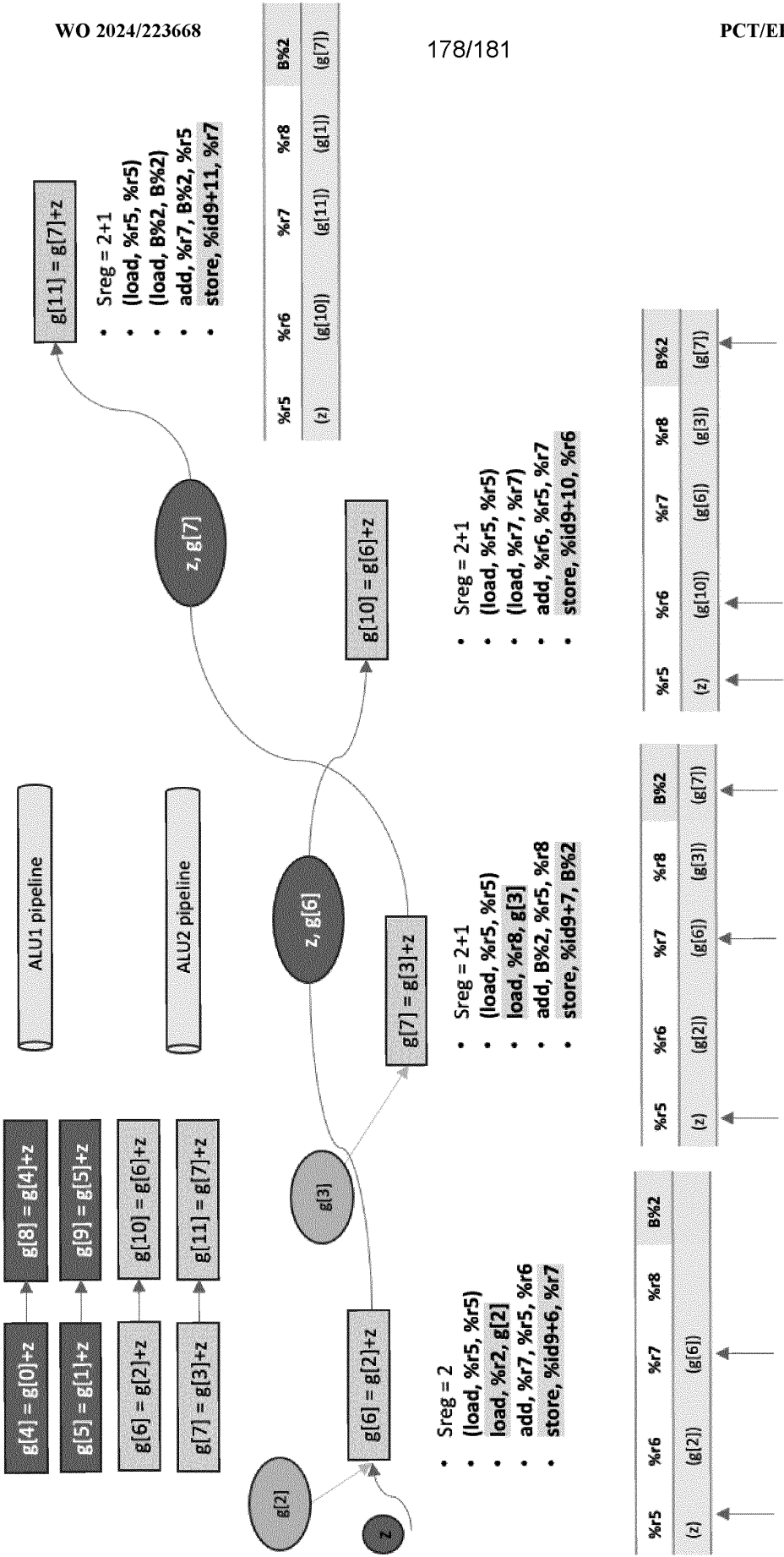


Fig. 165

pipeline 1		pipeline 2
load, %r1, %id2		
load, %r2, %id3		
load, %r4, 4		
add, %r3, %r1, %r2		
store, %id1, %3		
load, %r1, 5		
log, %r2, %r3		mul, %r6, %r3, %r3
store, %id4, %r2		store, %id5, %2
mul, %1, %r2, %r4		
store, %id6, %r1		
cmp, %r2, %r1, %r4		
nop		
nop		
load, %r2, g[0]		load, %r2, g[2]
load, %r4, g[1]		load, %r8, g[3]
add, %r3, %r1, %r2		add, %r7, %r5, %r6
store, %id9+4, %r3		store, %id9+6, %r7
add, B%1, %r1, %r4		add, B%2, %r5, %r8
store, %id9+5, B%1		store, %id9+7, B%2
add, %r2, %r1, %r3		add, %r6, %r5, %r7
store, %id9+8, %r2		store, %id9+10, %r6
add, %r3, %r1, B%1		add, %r7, B%2, %r5
store, %id9+9, %r3		store, %id9+11, %r7

Fig. 166

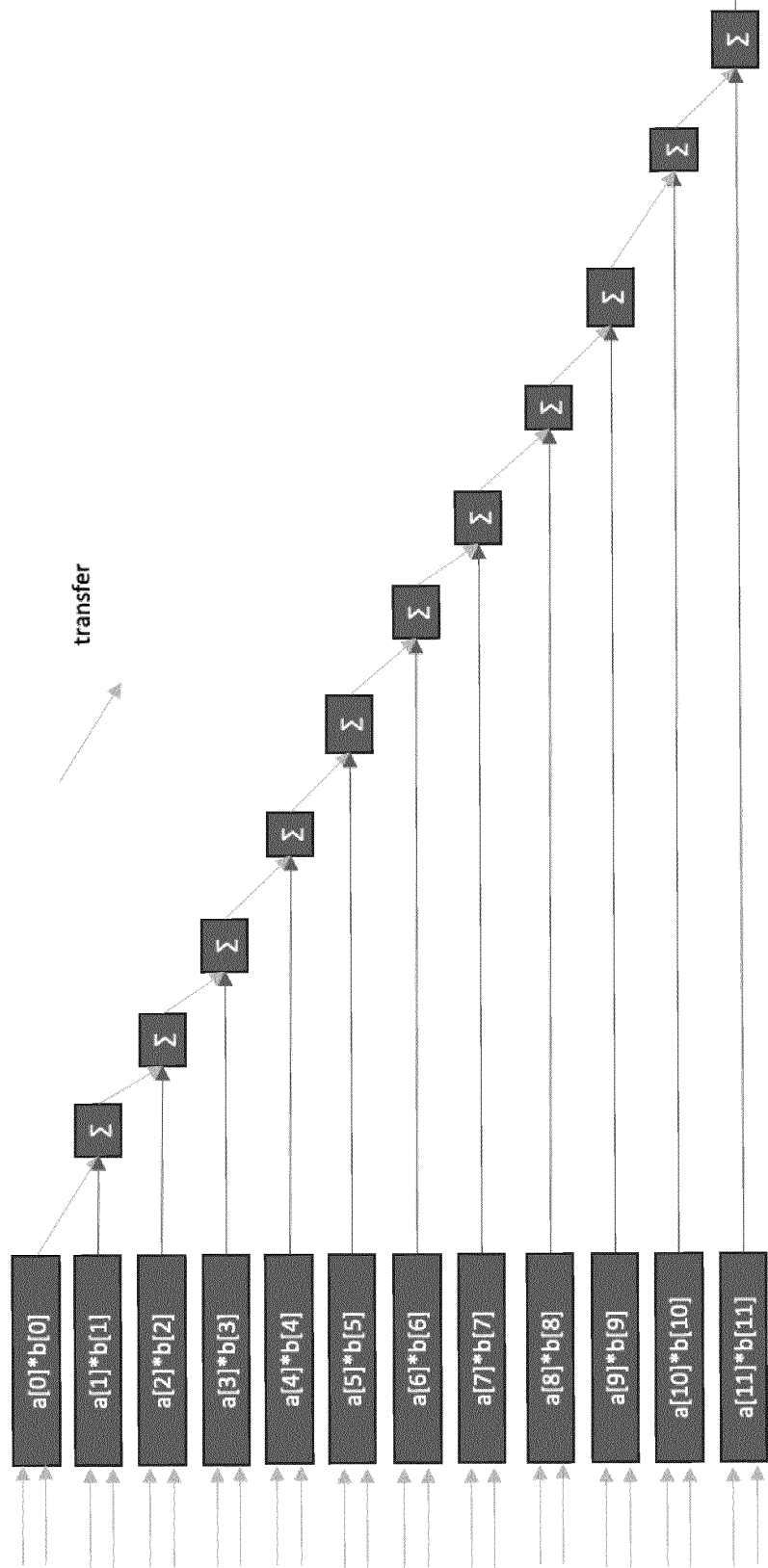


Fig. 167

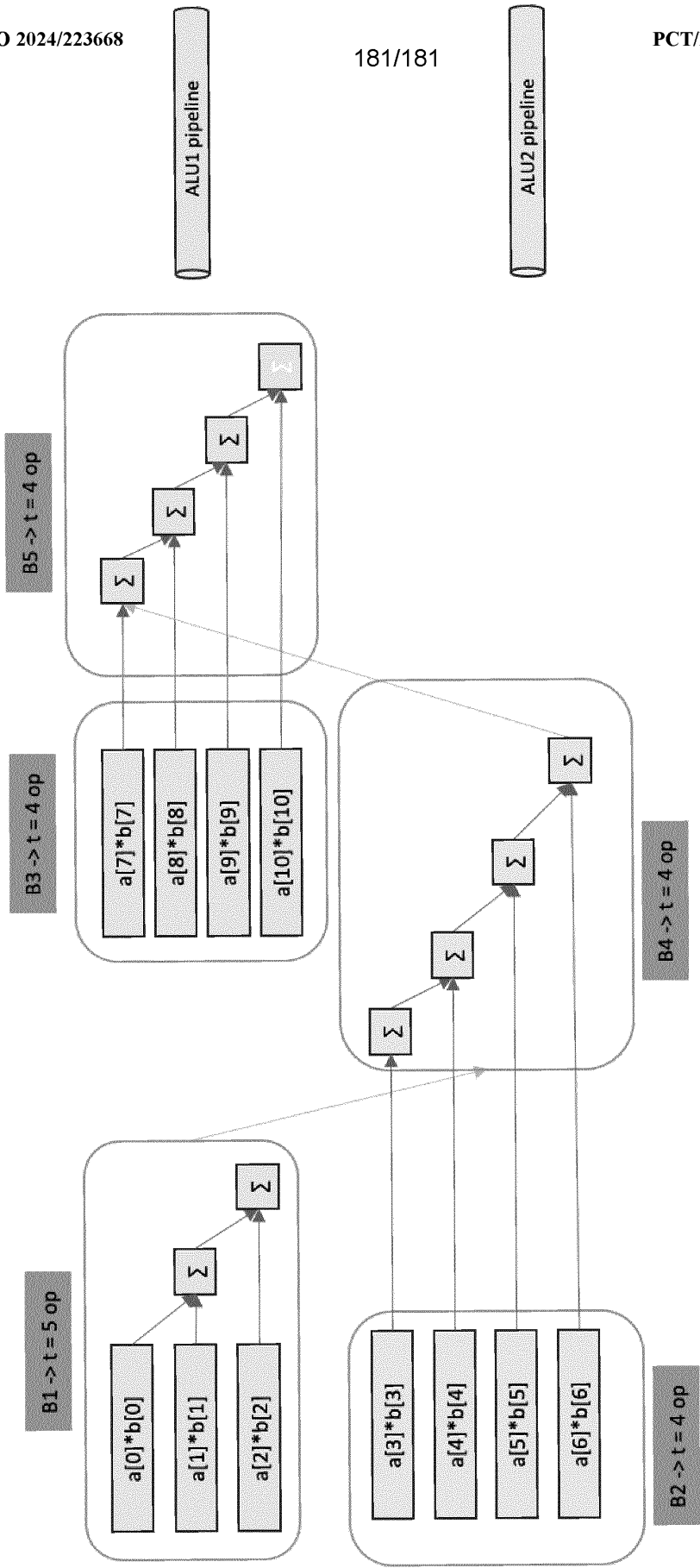


Fig. 168