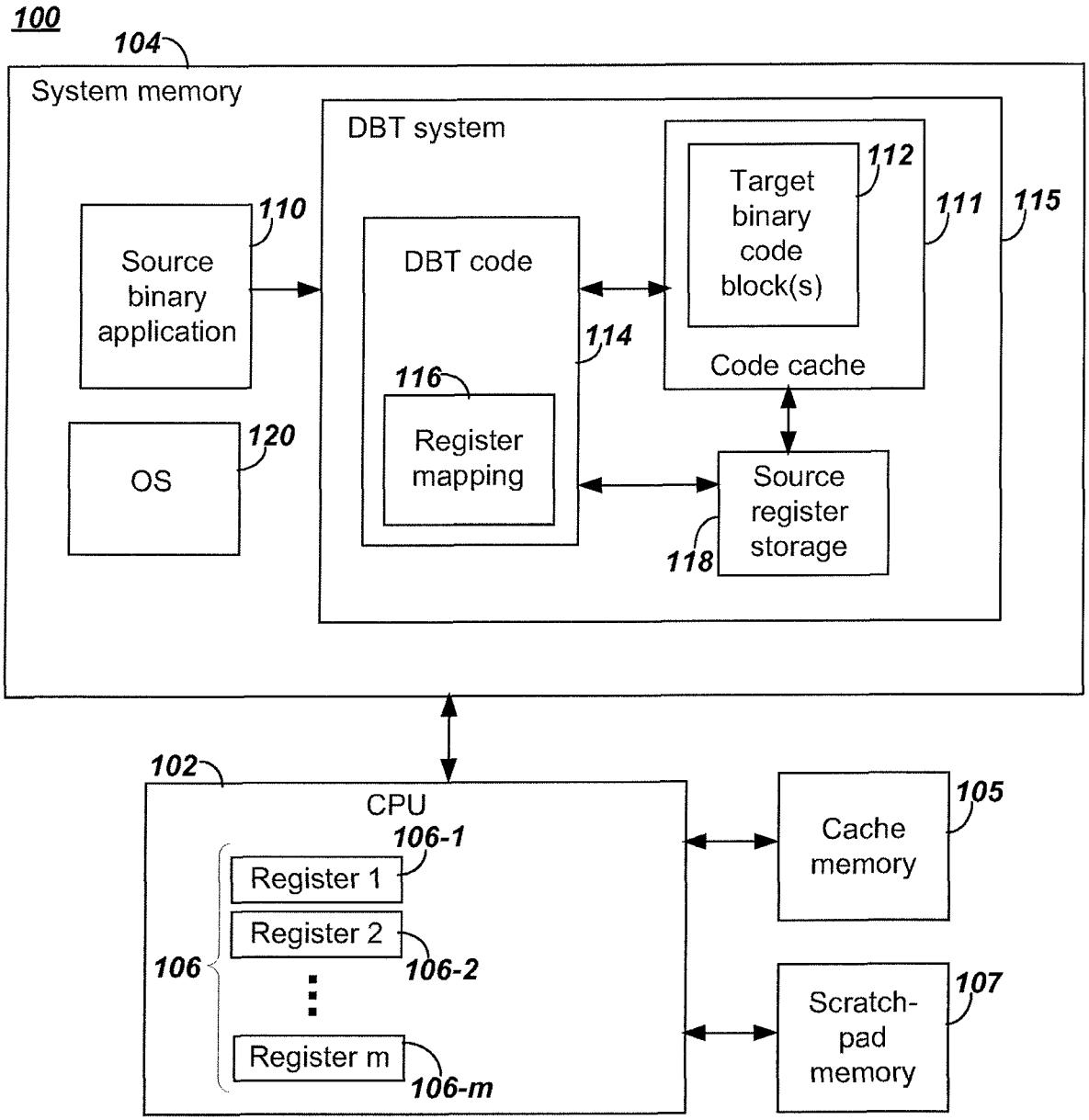
100
104
System memory
DBT system
Source binary application
110
DBT code
114
116
Register mapping
Target binary code block(s)
112
111
115
Code cache
OS
120
Source register storage
118
102
CPU
106-1
Register 1
Register 2
106-2
106
Register m
106-m
Cache memory
105
Scratch-pad memory
107

FIG. 1

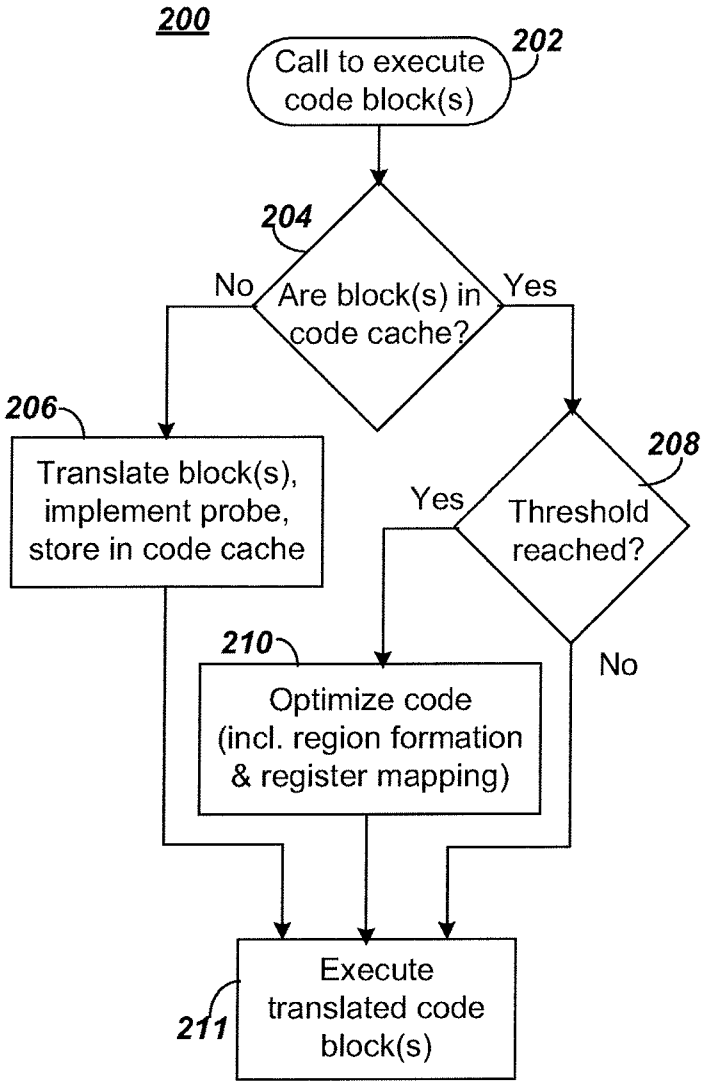
200
Call to execute code block(s)
202
204
Are block(s) in code cache?
No
Yes
206
Translate block(s), implement probe, store in code cache
208
Threshold reached?
Yes
No
210
Optimize code (incl. region formation & register mapping)
Execute translated code block(s)
211

FIG. 2A

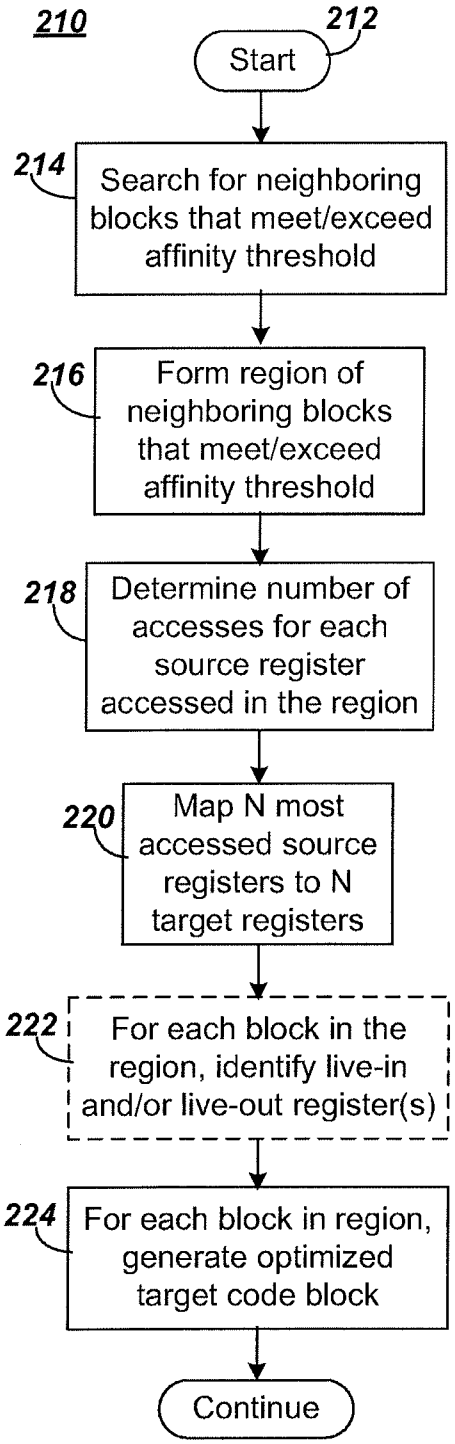
210
212
Start
214
Search for neighboring blocks that meet/exceed affinity threshold
216
Form region of neighboring blocks that meet/exceed affinity threshold
218
Determine number of accesses for each source register accessed in the region
220
Map N most accessed source registers to N target registers
222
For each block in the region, identify live-in and/or live-out register(s)
224
For each block in region, generate optimized target code block
Continue

FIG. 2B

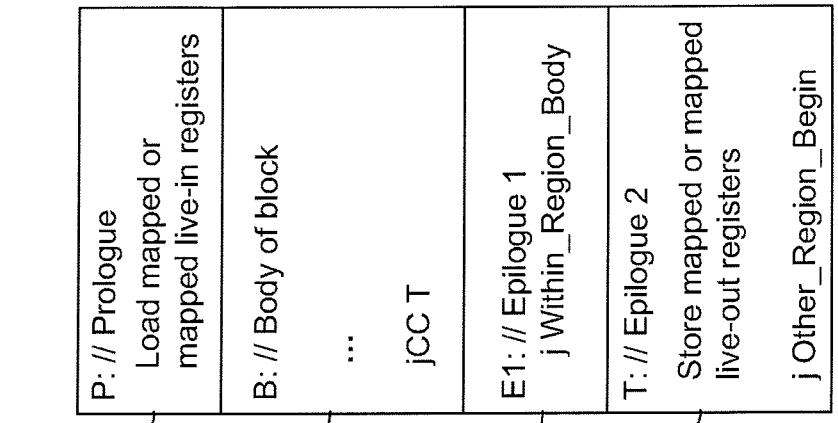
400
P: // Prologue
Load mapped or
mapped live-in registers
B: // Body of block
...
jCC T
E1: // Epilogue 1
j Within_Region_Body
T: // Epilogue 2
Store mapped or mapped
live-out registers
j Other_Region_Begin
410
420
430
440

FIG. 4

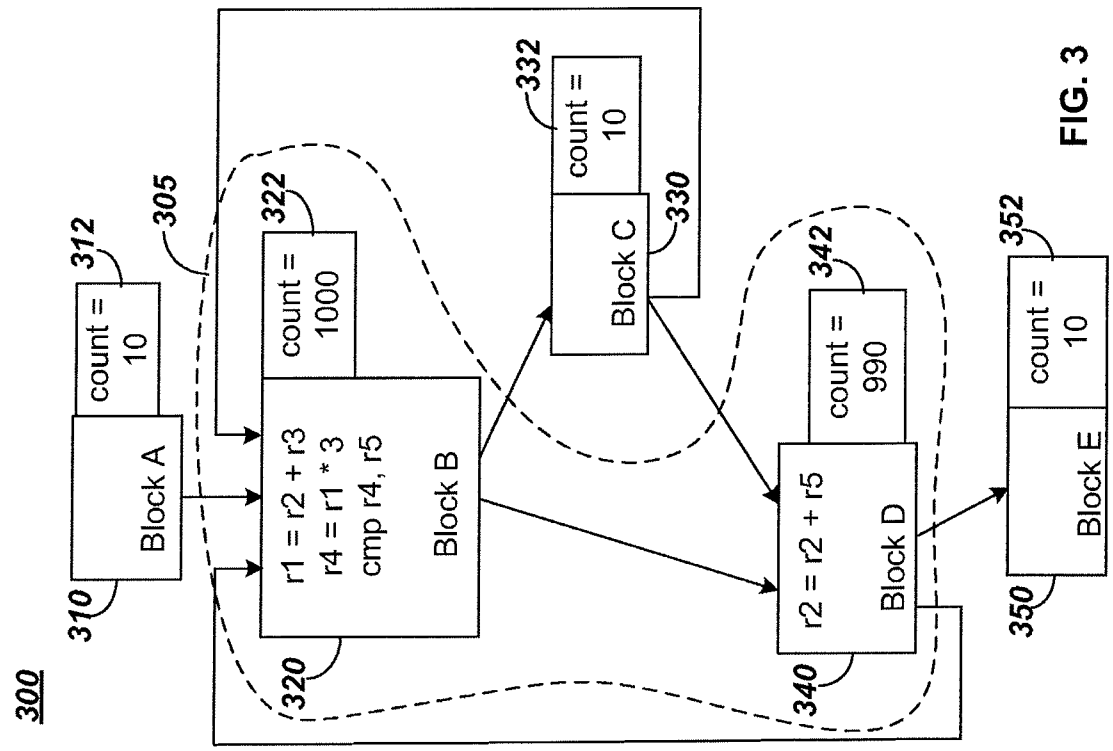
300
Block A
count = 10
310
312
305
r1 = r2 + r3
r4 = r1 * 3
cmp r4, r5
Block B
count = 1000
320
322
Block C
count = 10
330
332
r2 = r2 + r5
Block D
count = 990
340
342
Block E
count = 10
350
352

FIG. 3

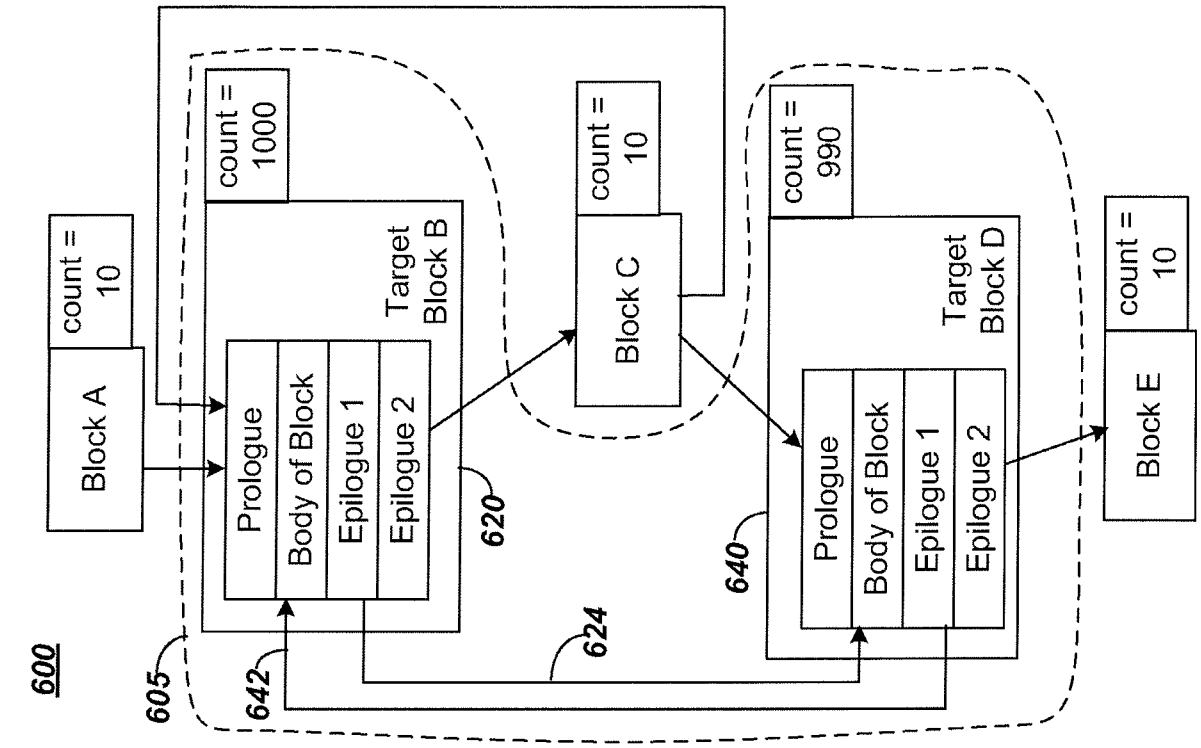
600
605
642
624
620
640
Block A
count = 10
count = 1000
Prologue
Body of Block
Epilogue 1
Epilogue 2
Target Block B
Block C
count = 10
count = 990
Prologue
Body of Block
Epilogue 1
Epilogue 2
Target Block D
Block E
count = 10

FIG. 6

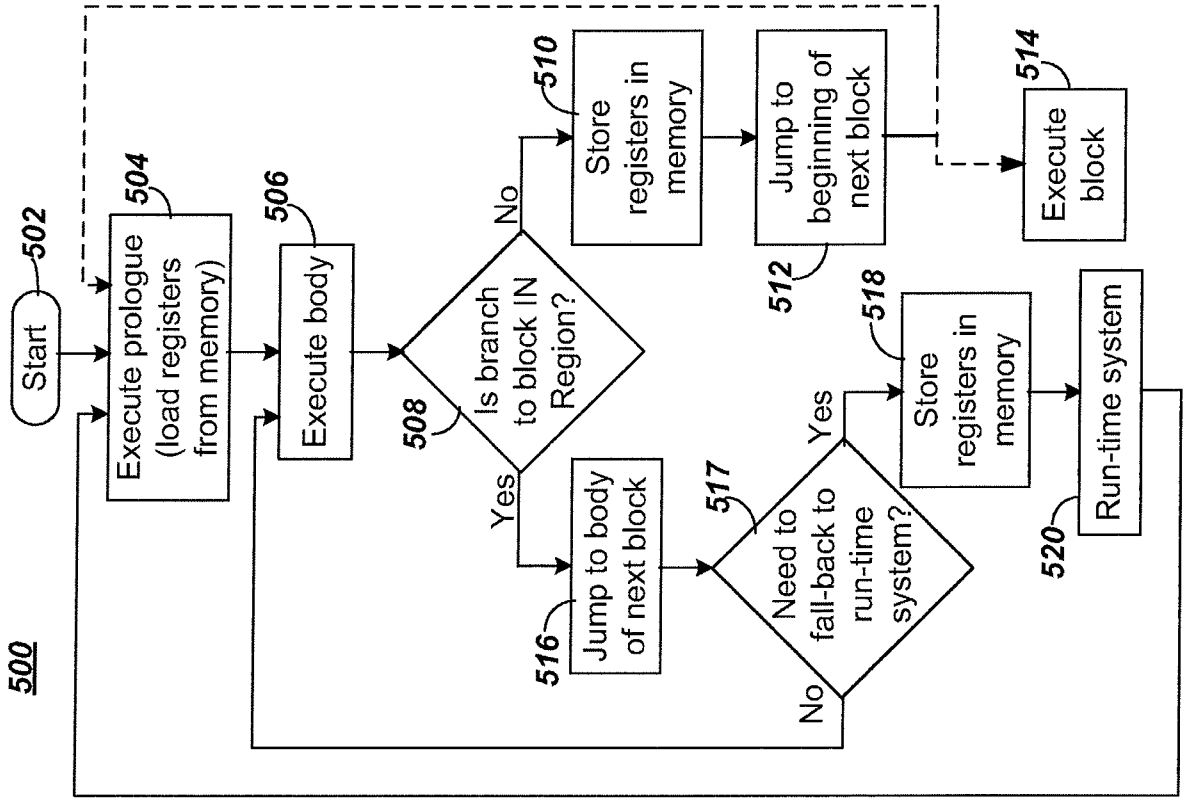
500
Start
502
Execute prologue (load registers from memory)
504
Execute body
506
Is branch to block IN Region?
508
No
Store registers in memory
510
Jump to beginning of next block
512
Execute block
514
Yes
Jump to body of next block
516
Need to fall-back to run-time system?
517
Yes
Store registers in memory
518
Run-time system
520
No

FIG. 5

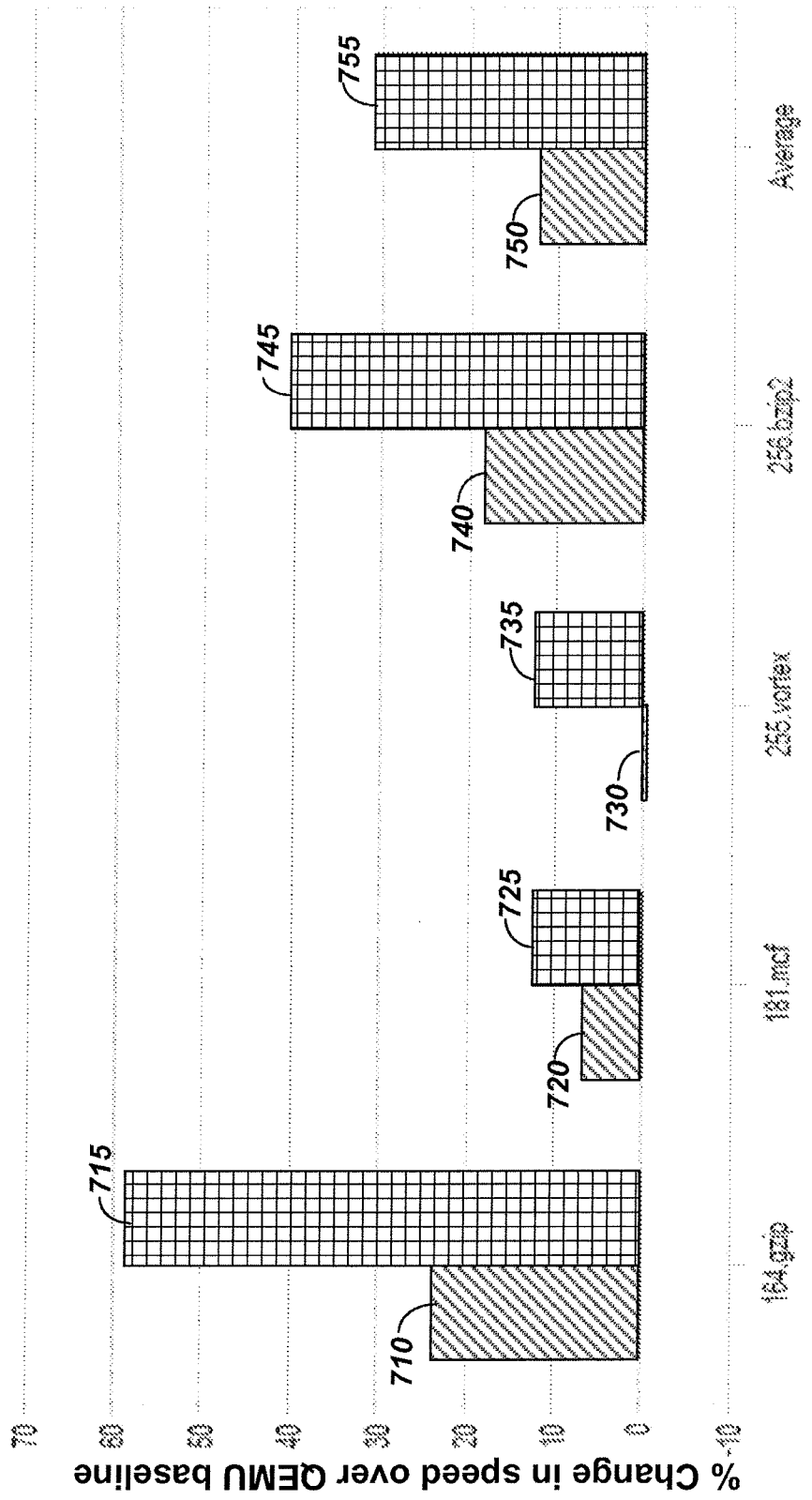
% Change in speed over QEMU baseline
70
60
50
40
30
20
10
0
-10
164.gzip
181.mcf
255.vortex
256.bzip2
Average
Benchmarks
710
715
720
725
730
735
740
745
750
755

FIG. 7