Motivating Example

Motivating Example

```c
void quantum_cond_phase(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(quantum_objcode_put(COND_PHASE, control, target))
    return;
  z = quantum_cexp(pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```c
void quantum_cond_phase_inv(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;


  z = quantum_cexp(-pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

Motivating Example

```
void quantum_cond_phase(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(quantum_objcode_put(COND_PHASE, control, target))
    return;
  z = quantum_cexp(pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void quantum_cond_phase_inv(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;


  z = quantum_cexp(-pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void merged(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;




  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

Motivating Example

```
void quantum_cond_phase(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(quantum_objcode_put(COND_PHASE, control, target))
    return;
  z = quantum_cexp(pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void quantum_cond_phase_inv(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;


  z = quantum_cexp(-pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void merged(bool func_id,
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(func_id)
    if(quantum_objcode_put(COND_PHASE, control, target))
      return;


  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

Motivating Example

```
void quantum_cond_phase(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(quantum_objcode_put(COND_PHASE, control, target))
    return;
  z = quantum_cexp(pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void quantum_cond_phase_inv(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;


  z = quantum_cexp(-pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void merged(bool func_id,
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(func_id)
    if(quantum_objcode_put(COND_PHASE, control, target))
      return;
  float var = (func_id)?pi:(-pi);
  z = quantum_cexp(var / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

Motivating Example

```
void quantum_cond_phase(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(quantum_objcode_put(COND_PHASE, control, target))
    return;
  z = quantum_cexp(pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void quantum_cond_phase_inv(
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;


  z = quantum_cexp(-pi / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```

```
void merged(bool func_id,
int control, int target, quantum_reg *reg){
  int i;
  COMPLEX_FLOAT z;
  if(func_id)
    if(quantum_objcode_put(COND_PHASE, control, target))
      return;
  float var = (func_id)?pi:(-pi);
  z = quantum_cexp(var / (1 << (control - target)));
  for(i=0; i<reg->size; i++) {
    if(reg->node[i].state & (1 << control)) {
      if(reg->node[i].state & (1 << target))
        reg->node[i].amplitude *= z;
    }
  }
  quantum_decohere(reg);
}
```