```
import random
import numpy as np
from concurrent.futures import ThreadPoolExecutor
rows, cols = 5, 5
num_iterations = 5
grid = np.random.randint(2, size=(rows, cols))
def count_neighbors(grid, x, y):
    neighbor_coords = [(-1, 0), (1, 0), (0, -1), (0, 1)]

    count = 0
    for dx, dy in neighbor_coords:
        nx, ny = x + dx, y + dy
        if 0 <= nx < rows and 0 <= ny < cols:
            count += grid[nx, ny]
    return count
def next_state(x, y, grid):
    alive_neighbors = count_neighbors(grid, x, y)
    if grid[x, y] == 1:
        if alive_neighbors < 2 or alive_neighbors > 3:
            return 0
        else:
            return 1
    else:
        if alive_neighbors == 3:
            return 1
        else:
            return 0
def update_grid(grid):
    new_grid = np.zeros_like(grid)
    with ThreadPoolExecutor() as executor:
        futures = []
        for x in range(rows):
            for y in range(cols):
                futures.append(executor.submit(next_state, x, y, grid))
        idx = 0
        for future in futures:
            x, y = idx // cols, idx % cols
            new_grid[x, y] = future.result()
            idx += 1

    return new_grid
def print_grid(grid):
    for row in grid:
        print(' '.join(str(cell) for cell in row))
    print()
print("Initial State:")
print_grid(grid)
```

```
for _ in range(num_iterations):
    grid = update_grid(grid)
    print("Next Generation:")
    print_grid(grid)
```

Output:

```
Initial State:

1 1 1 1 1
1 0 0 1 1
1 0 0 1 1
0 0 0 1 0
1 1 0 0 0

Next Generation:
1 1 1 1 1
1 0 0 1 1
0 0 0 1 1
0 0 0 0 0
0 0 0 0 0

Next Generation:
1 1 1 1 1
0 0 0 1 1
0 0 0 1 1
0 0 0 0 0
0 0 0 0 0

Next Generation:
0 1 1 1 1
0 0 0 1 1
0 0 0 1 1
0 0 0 0 0
0 0 0 0 0

Next Generation:
0 0 1 1 1
0 0 0 1 1
0 0 0 1 1
0 0 0 0 0
0 0 0 0 0

Next Generation:
0 0 0 1 1
0 0 0 1 1
0 0 0 1 1
0 0 0 0 0
0 0 0 0 0
```