In [9]:

```python
# Import the NumPy library as np
import numpy as np

# Example data (f) is an array containing 8 data points
f = np.array([6.000000000000000, 10.242640687119284, 2.000000000000000, -2.585786437626905,
              2.000000000000000, 1.757359312880716, -6.000000000000000, -5.414213562373098])

# Perform the FFT (Fast Fourier Transform) on the data 'f'
F = np.fft.fft(f)

# Get the number of samples (data points) in the array 'f'
N = len(f)

# Calculate the real coefficients (a_k) by scaling the real part of each Fourier coefficient by 2/N
a = F.real * 2 / N

# Calculate the imaginary coefficients (b_k) by scaling the imaginary part of each Fourier coefficient by -2/N
b = -F.imag * 2 / N

# Adjust a_0 and a_{N/2} (the constant and Nyquist components) since they are halved during the FFT process
a[0] /= 2
a[N//2] /= 2

# Print the real coefficients (a_k) and imaginary coefficients (-b_k)
print("Real coefficients (a_k):", a)
print("Imaginary coefficients (-b_k):", b)
```

```
Real coefficients (a_k): [1.0000000e+00 2.0000000e+00 3.0000000e+00 8.8817842e-16 4.4408921e-16
 8.8817842e-16 3.0000000e+00 2.0000000e+00]
Imaginary coefficients (-b_k): [-0.  4.  5. -0. -0. -0. -5. -4.]
```

In [10]:

```python
# Import the NumPy library as np and the Matplotlib.pyplot library as plt
import numpy as np
import matplotlib.pyplot as plt

# Input data (f) is an array containing 8 data points
f = np.array([6.000000000000000, 10.242640687119284, 2.000000000000000, -2.585786437626905,
              2.000000000000000, 1.757359312880716, -6.000000000000000, -5.414213562373098])

# Perform the FFT (Fast Fourier Transform) on the data 'f'
F = np.fft.fft(f)

# Get the number of samples (data points) in the array 'f'
N = len(f)

# Calculate the real coefficients (a_k) by scaling the real part of each Fourier coefficient by 2/N
a = F.real * 2 / N

# Calculate the imaginary coefficients (b_k) by scaling the imaginary part of each Fourier coefficient by -2/N
b = -F.imag * 2 / N

# Adjust a_0 and a_{N/2} (the constant and Nyquist components) since they are halved during the FFT process
a[0] /= 2
a[N//2] /= 2

# Define the x values for plotting the Fourier series reconstruction
x = np.linspace(0, 2*np.pi, 1000)

# Initialize s_4(x) with the constant term (a_0/2)
s = a[0] / 2

# Compute the Fourier series reconstruction s_4(x) using the Fourier coefficients
# Sum the terms for each frequency component (k), including cosine and sine terms
# The loop runs from k=1 to (N//2)-1, since a_N/2 is already accounted for
for k in range(1, N//2):
    s += a[k] * np.cos(k * x) + b[k] * np.sin(k * x)

# Add the term for a_N/2 * cos(N/2 * x)
s += a[N//2] / 2 * np.cos(N/2 * x)

# Plot the reconstructed Fourier series s_4(x)
plt.plot(x, s)
plt.grid(True)
plt.xlabel('x')
plt.ylabel('s_4(x)')
plt.show()
```