$$SnO_2 + 2Cl_2 \rightarrow SnCl_4 + O_2$$

-
-
-

$$r = k \cdot e^{-\frac{E_a}{RT}} \cdot [\mathrm{Cl_2}]^n$$

  - k
  - E_a
  - T
  - R
  - $[Cl_2]$
  - n

-
-

modelica

```
model SnO2_Cl2_Reaction
  import ThermoSysPro;
  import Modelica.Math;

  // Define Constants
  constant Real R = 8.314;  // Universal gas constant (J/mol·K)
  constant Real Ea = 100000; // Activation energy (J/mol) (example value)
  constant Real k = 1.0e5;  // Pre-exponential factor (arbitrary value)

  // Define Variables
  Real Cl2_conc(start=20e-6);  // Initial chlorine concentration in ppm
  Real SnCl4_conc(start=0);  // Tin chloride formation (mol/m³)
  Real rate;  // Reaction rate (mol/m³·s)
  Real T(start=300);  // Temperature in Kelvin (assumed 300 K)

equation
  // Arrhenius-based reaction rate
  rate = k * exp(-Ea / (R * T)) * Cl2_conc;

  // Mass balance equations
  der(Cl2_conc) = -2 * rate;  // Chlorine consumption
  der(SnCl4_conc) = rate;  // Tin chloride formation

  annotation (experiment(StartTime=0, StopTime=1000, Tolerance=1e-6));
end SnO2_Cl2_Reaction;
```

-
-
- k E_a
-
-
-

T

- $T = 300$
-

---

modelica

```
model SnO2_Cl2_Reaction
  import Modelica.Math;

  // Define Constants
  constant Real R = 8.314;    // Universal gas constant (J/mol·K)
  constant Real Ea = 100000;  // Activation energy (J/mol)
  constant Real k = 1.0e5;    // Pre-exponential factor (arbitrary value)

  // Define Variables
  Real Cl2_conc(start=20e-6);  // Chlorine concentration in ppm (converted to fraction)
  Real SnCl4_conc(start=0);    // Tin chloride formation (mol/m³)
  Real rate;  // Reaction rate (mol/m³·s)
  parameter Real T = 300;  // Constant temperature in Kelvin

equation
  // Arrhenius equation for reaction rate
  rate = k * exp(-Ea / (R * T)) * Cl2_conc;

  // Mass balance equations
  der(Cl2_conc) = -2 * rate;    // Chlorine consumption
  der(SnCl4_conc) = rate;       // Tin chloride formation

  annotation (experiment(StartTime=0, StopTime=1000, Tolerance=1e-6));
end SnO2_Cl2_Reaction;
```

$$\frac{dT}{dt} = \frac{-\Delta H \cdot r}{\rho C_p}$$

- ΔH
- ρ
- C_p
- r

modelica

```
model SnO2_Cl2_Reaction_DynamicT
  import Modelica.Math;

  // Define Constants
  constant Real R = 8.314;    // Universal gas constant (J/mol·K)
  constant Real Ea = 100000;  // Activation energy (J/mol)
  constant Real k = 1.0e5;    // Pre-exponential factor (arbitrary value)
```

```
  constant Real deltaH = -30000; // Reaction enthalpy change (J/mol) (example)
  constant Real rho = 5000;    // Density of solid phase (kg/m³)
  constant Real Cp = 500;      // Specific heat capacity (J/kg·K)

  // Define Variables
  Real Cl2_conc(start=20e-6);  // Chlorine concentration in ppm (converted to fraction)
  Real SnCl4_conc(start=0);    // Tin chloride formation (mol/m³)
  Real rate;  // Reaction rate (mol/m³·s)
  Real T(start=300);  // Temperature (K)

equation
  // Arrhenius equation for reaction rate
  rate = k * exp(-Ea / (R * T)) * Cl2_conc;

  // Mass balance equations
  der(Cl2_conc) = -2 * rate;   // Chlorine consumption
  der(SnCl4_conc) = rate;      // Tin chloride formation

  // Energy balance (Temperature change)
  der(T) = (-deltaH * rate) / (rho * Cp);

  annotation (experiment(StartTime=0, StopTime=1000, Tolerance=1e-6));
end SnO2_Cl2_Reaction_DynamicT;
```

-
- T

-
-
-

Property	SnO_2 (Solid)	Cl_2 (Gas)	$SnCl_4$ (Liquid)	O_2 (Gas)
(kg/m^3)	6900	3.21	2200	1.43
(J/kg·K)	643	775	1000	920
(W/m·K)	33	0.008	0.15	0.025
	–	239	487	90
	–	–	-30,000	–

modelica

```
model SnO2_Cl2_Reaction
 import Modelica.Math;

 // Constants
 constant Real R = 8.314;    // Universal gas constant (J/mol·K)
 constant Real Ea = 100000;  // Activation energy (J/mol)
 constant Real k = 1.0e5;    // Pre-exponential factor (arbitrary value)
 constant Real deltaH = -30000; // Reaction enthalpy change (J/mol)

 // Material Properties
 constant Real rho_SnO2 = 6900;    // Density of SnO2 (kg/m^3)
 constant Real Cp_SnO2 = 643;      // Specific heat of SnO2 (J/kg·K)

 constant Real rho_Cl2 = 3.21;     // Density of Cl2 (kg/m^3)
 constant Real Cp_Cl2 = 775;       // Specific heat of Cl2 (J/kg·K)

 constant Real rho_SnCl4 = 2200;   // Density of SnCl4 (kg/m^3)
 constant Real Cp_SnCl4 = 1000;    // Specific heat of SnCl4 (J/kg·K)
```

```
  // Variables
  Real Cl2_conc(start=20e-6);  // Chlorine concentration (ppm)
  Real SnCl4_conc(start=0);    // Tin chloride formation (mol/m³)
  Real rate;  // Reaction rate (mol/m³·s)
  Real T(start=300);  // Temperature in Kelvin

equation
  // Arrhenius equation for reaction rate
  rate = k * exp(-Ea / (R * T)) * Cl2_conc;

  // Mass balance equations
  der(Cl2_conc) = -2 * rate;   // Chlorine consumption
  der(SnCl4_conc) = rate;      // Tin chloride formation

  // Energy balance (Temperature change)
  der(T) = (-deltaH * rate) / (rho_SnO2 * Cp_SnO2);

  annotation (experiment(StartTime=0, StopTime=1000, Tolerance=1e-6));
end SnO2_Cl2_Reaction;
```

modelica

```
model SnO2_Cl2_Reaction_Library
  import ThermoSysPro.Properties.WaterSteam.IF97;
  import ThermoSysPro;
  import Modelica.Math;

  // Constants
  constant Real R = 8.314;
  constant Real Ea = 100000;
  constant Real k = 1.0e5;
  constant Real deltaH = -30000;

  // Thermodynamic Properties from Library
  Real rho_Cl2 = IF97.rho_pT(101325, 300);   // Density of Cl2 at 1 atm, 300K
  Real Cp_Cl2 = IF97.cp_pT(101325, 300);     // Specific heat of Cl2
  Real rho_SnCl4 = 2200; // Hardcoded, since it's not in IF97
  Real Cp_SnCl4 = 1000;

  // Variables
  Real Cl2_conc(start=20e-6);
  Real SnCl4_conc(start=0);
  Real rate;
  Real T(start=300);

equation
  // Arrhenius equation for reaction rate
  rate = k * exp(-Ea / (R * T)) * Cl2_conc;

  // Mass balance
  der(Cl2_conc) = -2 * rate;
  der(SnCl4_conc) = rate;

  // Energy balance
  der(T) = (-deltaH * rate) / (rho_Cl2 * Cp_Cl2);
```

```
  annotation (experiment(StartTime=0, StopTime=1000, Tolerance=1e-6));
end SnO2_Cl2_Reaction_Library;
```

-
-
-

-
-

$$O_2\,(gas) + e^- \rightarrow O_2^-\,(ads)$$

$$O_2^-\,(ads) + e^- \rightarrow 2O^-(ads)$$

-

$$Cl_2\,(gas) + O_2^-\,(ads) + 2e^- \rightarrow 2Cl^-(ads) + O_2\,(gas)$$

-
-

- $$2Cl^{-}\left(ads\right)+O_{2}\left(gas\right)\rightarrow Cl_{2}\left(gas\right)+O_{2}^{-}\left(ads\right)+2e^{-}$$
-

-
-
-

-
  -
  -
  -

R_{ads}

-
-

R_{des}

-
-

-
-

modelica

```
model SnO2_Cl2_Reaction_Adsorption
 import Modelica.Math;

 // Constants
 constant Real R = 8.314;    // Universal gas constant (J/mol·K)
 constant Real Ea_ads = 50000;  // Activation energy for adsorption (J/mol)
 constant Real Ea_des = 80000;  // Activation energy for desorption (J/mol)
 constant Real k_ads = 1.0e3;   // Adsorption rate constant (1/s)
 constant Real k_des = 5.0e4;   // Desorption rate constant (1/s)
 constant Real deltaH = -30000; // Reaction enthalpy change (J/mol)

 // Material Properties
 constant Real rho_SnO2 = 6900;  // Density of SnO2 (kg/m³)
 constant Real Cp_SnO2 = 643;    // Specific heat of SnO2 (J/kg·K)

 // Gas Properties
 constant Real rho_Cl2 = 3.21;  // Density of Cl2 (kg/m³)
 constant Real Cp_Cl2 = 775;    // Specific heat of Cl2 (J/kg·K)

 // Variables
 Real Cl2_conc(start=20e-6);   // Chlorine concentration in ppm
 Real SnCl4_conc(start=0);     // Tin chloride formation (mol/m³)
 Real O2_ads(start=1e-6);      // Adsorbed oxygen concentration (mol/m²)
 Real Cl_ads(start=0);         // Adsorbed chlorine concentration (mol/m²)
 Real rate_ads;                // Adsorption rate (mol/m²·s)
 Real rate_des;                // Desorption rate (mol/m²·s)
 Real rate_reaction;           // Surface reaction rate (mol/m²·s)
 Real T(start=300);            // Temperature in Kelvin
 Real resistance(start=1e6);   // Sensor resistance (ohm)
```

```
equation
  // Arrhenius adsorption rate
  rate_ads = k_ads * exp(-Ea_ads / (R * T)) * Cl2_conc * (1 - Cl_ads);

  // Arrhenius desorption rate
  rate_des = k_des * exp(-Ea_des / (R * T)) * Cl_ads;

  // Surface reaction: Cl2 reacts with O2 on SnO₂
  rate_reaction = rate_ads - rate_des;

  // Mass balance equations
  der(Cl2_conc) = -rate_ads + rate_des;  // Cl2 gas exchange
  der(O2_ads) = -rate_reaction;          // Oxygen loss due to reaction
  der(Cl_ads) = rate_reaction;           // Chlorine adsorption dynamics
  der(SnCl4_conc) = rate_reaction;       // SnCl4 formation

  // Energy balance (Temperature change due to reaction heat)
  der(T) = (-deltaH * rate_reaction) / (rho_SnO2 * Cp_SnO2);

  // Sensor resistance model: Resistance increases with adsorbed Cl
  resistance = 1e6 * (1 + Cl_ads);

  annotation (experiment(StartTime=0, StopTime=1000, Tolerance=1e-6));
end SnO2_Cl2_Reaction_Adsorption;
```

-
-
-
-
-
-
-
-

-
-
-
-
-
-
-
-
-
-

```python
Always show details  s np
 import matplotlib.pyplot as plt
 from scipy.integrate import solve_ivp

 # Constants
 R = 8.314       # Universal gas constant (J/mol·K)
 Ea_ads = 50000  # Activation energy for adsorption (J/mol)
 Ea_des = 80000  # Activation energy for desorption (J/mol)
 k_ads = 1.0e3   # Adsorption rate constant (1/s)
 k_des = 5.0e4   # Desorption rate constant (1/s)
 deltaH = -30000 # Reaction enthalpy change (J/mol)

 # Material properties
 rho_SnO2 = 6900  # Density of SnO2 (kg/m³)
 Cp_SnO2 = 643    # Specific heat of SnO2 (J/kg·
```

$$D_k = \frac{2r}{3}\sqrt{\frac{8RT}{\pi M}}$$

- D_k
- r
- M
- R
- T

$$D_{\text{eff}} = \epsilon D_k / \tau$$

- D_{eff}
- ϵ
- τ

$$\text{rate}_{\text{ads}} = k_{\text{ads}} \cdot D_{\text{eff}} \cdot C_{\text{Cl}_2} \cdot (1 - \theta_{\text{Cl}})$$

$$\text{rate}_{\text{des}} = k_{\text{des}} \cdot D_{\text{eff}} \cdot \theta_{\text{Cl}}$$

---

python

Always show details

```
ımeters for Knudsen Diffusion
pore_radius = 5e-9  # Pore radius (5 nm in meters)
M_Cl2 = 70.906      # Molar mass of Cl2 (g/mol)
epsilon = 0.4       # Porosity
tau = 2.5           # Tortuosity
```

```
# Simulation time
t_span = (0, 1000)  # Time from 0 to 1000 seconds
t_eval = np.linspace(*t_span, 1000)

# Initial conditions
Cl2_conc_0 = 20e-6   # Chlorine concentration (ppm)
O2_ads_0 = 1e-6      # Adsorbed oxygen concentration (mol/m²)
Cl_ads_0 = 0         # Adsorbed chlorine concentration (mol/m²)
SnCl4_conc_0 = 0     # SnCl4 formation (mol/m³)
T_0 = 300            # Temperature (K)

# Function to compute Knudsen Diffusivity
def knudsen_diffusivity(T):
    return (2 * pore_radius / 3) * np.sqrt((8 * R * T) / (np.pi * M_Cl2))

# Function to compute effective diffusivity
def effective_diffusivity(T):
    Dk = knudsen_diffusivity(T)
    return epsilon * Dk / tau

# Differential equations
def reaction_dynamics(t, y):
    Cl2_conc, O2_ads, Cl_ads, SnCl4_conc, T = y

    # Effective diffusivity
    Deff = effective_diffusivity(T)

    # Adsorption rate (with Knudsen diffusion effect)
    rate_ads = k_ads * Deff * Cl2_conc * (1 - Cl_ads)

    # Desorption rate
    rate_des = k_des * Deff * Cl_ads * np.exp(-Ea_des / (R * T))

    # Surface reaction rate
    rate_reaction = rate_ads - rate_des

    # Energy balance (reaction heat)
    dT_dt = (-deltaH * rate_reaction) / (rho_SnO2 * Cp_SnO2)

    # Mass balance
    dCl2_conc_dt = -rate_ads + rate_des
    dO2_ads_dt = -rate_reaction
    dCl_ads_dt = rate_reaction
    dSnCl4_conc_dt = rate_reaction

    return [dCl2_conc_dt, dO2_ads_dt, dCl_ads_dt, dSnCl4_conc_dt, dT_dt]

# Solve ODEs
y0 = [Cl2_conc_0, O2_ads_0, Cl_ads_0, SnCl4_conc_0, T_0]
sol = solve_ivp(reaction_dynamics, t_span, y0, t_eval=t_eval, method='RK45')

# Extract results
Cl2_conc = sol.y[0]
O2_ads = sol.y[1]
Cl_ads = sol.y[2]
SnCl4_conc = sol.y[3]
T = sol.y[4]

# Compute sensor resistance (increases with Cl_ads)
resistance = 1e6 * (1 + Cl_ads)

# Plot results
plt.figure(figsize=(12,
```

```
model SnO2_Cl2_Reaction_Adsorption_Oxygen
  import Modelica.Math;

  // Constants
  constant Real R = 8.314;     // Universal gas constant (J/mol·K)
  constant Real Ea_ads = 50000;  // Activation energy for adsorption (J/mol)
  constant Real Ea_des = 80000;  // Activation energy for desorption (J/mol)
  constant Real k_ads = 1.0e3;   // Adsorption rate constant (1/s)
  constant Real k_des = 5.0e4;   // Desorption rate constant (1/s)
  constant Real deltaH = -30000; // Reaction enthalpy change (J/mol)

  // Pore Properties (Knudsen Diffusion)
  constant Real pore_radius = 5e-9;   // Pore radius (m)
  constant Real porosity = 0.4;       // Porosity (dimensionless)
  constant Real tortuosity = 2.5;     // Tortuosity (dimensionless)
```