ADVANCED MATERIALS

Supporting Information

for *Adv. Mater.*, DOI 10.1002/adma.202305191

Patterning Complex Line Motifs in Thin Films Using Immersion-Controlled Reaction-Diffusion

Christiaan T. van Campenhout, Hinco Schoenmaker, Martin van Hecke and Willem L. Noorduin**

WILEY-VCH

Supporting Information

Tunable Thin Film Patterning Using Immersion Controlled Reaction-Diffusion

Christiaan T. van Campenhout, Hinco Schoenmaker, Martin van Hecke, Willem L. Noorduin**

This document contains the following supporting information:

Figure S1 – Patterning process
Movie S1 – Timelapse closeup patterning
Movie S2 – wafer scale timelapse
Movie S3 – diffraction pattern throughout entire thin film
Movie S4 – Moiré pattern upon compression
Python3 code for the immersion-controlled reaction-diffusion model

S1 – Patterning process

The patterning process described in this work uses only a few components, some of which are for inducing pattern formation and others are for live imaging (Figure S1).

Pattern formation:

- New Era NE-1000 Syringe pump.
- Custom sample holder: two glass windows separated by 6 mm thick, u-shaped Viton rubber, tightened with a stainless steel frame. This effectively makes a big cuvette, in which we can mount glass slides containing gel films.
- 20 mL syringe, connected to PTFE tubing.

The patterning process starts by mounting a gel film supported on a glass slide (75x50 mm) inside an empty sample holder, such that the gel film faces the empty chamber. Then, PTFE tubing connected to a 20 mL syringe introduced to the bottom of this chamber. This syringe is then placed on a syringe pump, with the appropriate rate and volume settings. Starting the syringe pump now starts the patterning process.

Live imaging

- Halogen light source
- Canon EOS 850D camera with Laowa 25mm F/2.8 2.5-5X Ultra-Macro lens
- xyz stage, with motorized y component.

The entire sample cell can be monitored by a camera equipped with an ultra-macro lens. In order to keep patterning in view, the entire sample holder is mounted on an xyz stage, of which the y component is motorized. As immersion reagent is pumped into the sample holder, the liquid-gel contact line starts moving upwards. By moving the entire sample holder down at the same speed we ensure that patterning occurs in view.

After patterning is complete, gel films are removed from the sample holder, rinsed with DI water and soaked twice in 200 mL DI water for 10 minutes to remove any unreacted reagent.

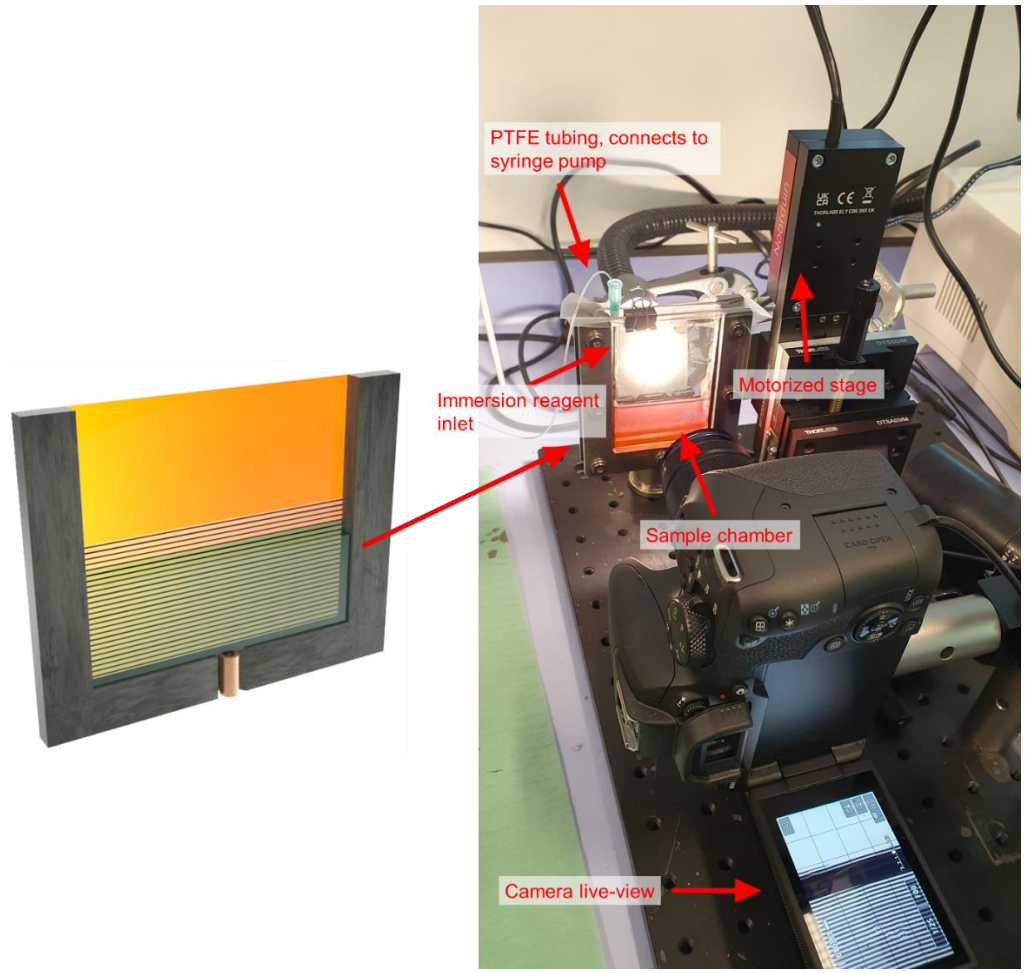

Figure S1: Overview image showing the experimental setup, with the sample chamber containing a gel sheet in the middle connected to a motorized stage that moves the sample down as the liquid-gel contact line moves up. Note that a syringe pump is just out of view on the left, which pumps the immersion reagent through PTFE tubing into the bottom of the sample chamber. The stage, syringe pump and camera are all connected to a PC, which allows for remote monitoring and tuning.

Movie S1 – Silver Dichromate Patterning with R-DIP.mp4

This movie shows a timelapse of the R-DIP process, where 1 second of movie corresponds to 240 seconds in real time. Out of view, immersion reagent is injected into the empty chamber, which causes the contact line to move upwards. Simultaneously, the chamber is moved downwards at the same speed to keep the contact line and pattern formation in view.

Movie S2 – Wafer-scale patterning of silver nanoparticles.mp4

This movie shows a timelapse of the wafer-scale R-DIP process, where the total movie time corresponds to 60 hours real-time. Because the patterns produced in this example are too small to see by naked-eye, we include three zoom-ins of the microscopic pattern taken at the location corresponding to the contact line position at that time in the video.

Movie S3 – A4-scale diffraction grating

In this movie we shine a green (532 nm) laser light at the dried A4-sized diffraction grating, to demonstrate the uniformity of the pattern throughout.

Movie S4 – Moiré patterns emerge under compression

In this movie we show an opto-mechanical sensor based on Moiré patterns, made by superimposing two identical line patterns of silver dichromate parallel, separated by a flexible spacer. We then slowly compress and decompress the sensor and observe the emerging Moiré pattern.

Python3 code for the immersion-controlled reaction-diffusion model

```
import numpy as np
import time
from scipy import special
from numba import njit
import numba as nb
from joblib import Parallel, delayed
import multiprocessing
#Measuring run time
start_time = time.time()

#Defining the box
x = 0.005
y = 0.05
dx = 0.00005
dy = 0.00005
dx2 = dx**2
dy2 = dy**2
nx = int(x/dx)
ny = int(y/dy)
xgrid = np.linspace(0, x, nx)
ygrid = np.linspace(0, y, ny)
dt = .01
dtdx2 = dt/dx2
#reaction/diffusion constants
DAg = 3*(10**-9)*dt
Dsol = 3*(10**-9)*dt
ksalt = 0.001*dt
knuc = (0.001/1000)*dt
kgrowth = 0.01*10000*dt
cthresh0 = 0.002
randomgrid = np.random.uniform(0.9,1.1, size=((ny,nx)))
cthresh = np.ones((ny,nx))*cthresh0*randomgrid
#starting concentrations
cAg0 = 0.80
cdich0 = 0.1

volinner = 1
volouter = 10

@njit
def roller_left(a):
    b = np.zeros((ny,nx))
    for i in nb.prange(ny):
        b[i, 1:] = a[i, :(nx - 1)]
        b[i, :1] = a[i, (nx - 1):]
    return b
@njit
def roller_right(a):
```

```
    b = np.zeros((ny,nx))
    for i in nb.prange(ny):
        b[i, :(nx - 1)] = a[i, 1:]
        b[i, (nx - 1):] = a[i, :1]
    return b
@njit
def roller_bot(a):
    b = np.zeros((ny,nx))
    for i in nb.prange(nx):
        b[:(ny - 1), i] = a[1:, i]
        b[(ny-1):, i] = a[:1, i]
    return b
@njit
def roller_top(a):
    b = np.zeros((ny,nx))
    for i in nb.prange(nx):
        b[1:, i] = a[:(ny - 1), i]
        b[:1, i] = a[(ny-1):, i]
    return b

@njit
def diff(x, Dx):
    xd = np.zeros((ny,nx))
    xT = roller_top(x)
    xB = roller_bot(x)
    xL = roller_left(x)
    xR = roller_right(x)
    xd[1:-1] = Dx*((xL[1:-1]+xR[1:-1]-2*x[1:-1])/dx2 + (xT[1:-1]+xB[1:-1]-
2*x[1:-1])/dy2)
    xd[0] = Dx*((x[1]-x[0])/dy2)
    xd[-1] = Dx*((x[-2]-x[-1]))/dy2
    return xd

@njit
def growth_check(x):
    xd = np.zeros((ny,nx))
    xT = roller_top(x)
    xB = roller_bot(x)
    xL = roller_left(x)
    xR = roller_right(x)

    return (xT+xB+xL+xR)>0.001



@njit
def simulator(cdich, cAg, csol, cNP, act_time, cthresh, rate_mid, tgrid):
    for i in tgrid:
        act_time = i
```

```
        cAg = cAg + diff(cAg, DAg) - 2*cAg*cAg*cdich*ksalt
        cdich = cdich - cAg*cAg*cdich*ksalt
        csol = csol + diff(csol, Dsol) + cAg*cAg*cdich*ksalt -
knuc*csol*(csol>cthresh) - csol*cNP*kgrowth  - csol*growth_check(cNP)*kgrowth
        cNP = cNP + knuc*csol*(csol>cthresh) + csol*cNP*kgrowth  +
csol*growth_check(cNP)*knuc
        rate = rate_mid
        channel_open = rate*act_time
        channel_front = int(channel_open/y) + int(ny/10)
        cAg[:channel_front] = cAg0*
(volouter/(((channel_front/ny)*volinner)+volouter))

    return cNP, cAg

def init_run(time_total):
    act_time = 0
    t = time_total #t in seconds
    nt = int(t/dt)
    tgrid = np.linspace(0,t,nt)
    cAg = np.zeros((ny,nx))
    cdich = np.zeros((ny,nx))
    csol = np.zeros((ny,nx))
    cNP = np.zeros((ny,nx))
    rate_mid = 0.005
    #Filling cAg and cdich np arrays with starting concentrations (erf to
provide less of a sharp interface)
    for i in range(ny):
        cAg[i] = cAg0*(special.erf(ny-(9*i))+1)/2
        cdich[i] = cdich0*(special.erf((9*i)-ny)+1)/2

    name = f"24012023_long2Dpump_rate={rate_mid}_t={t}"

    pattern_final, silver_conc = simulator(cdich, cAg, csol, cNP, act_time,
cthresh, rate_mid, tgrid)
    np.savetxt(f'{name}_pattern.csv', pattern_final, delimiter=',')
    np.savetxt(f'{name}_silver.csv', silver_conc, delimiter=',')
    #Printing the runtime
    print(f"Run took: {time.time() - start_time:.2f} seconds")


time_list = [2500, 5000, 7500]

num_cores = multiprocessing.cpu_count()
results = Parallel(n_jobs=num_cores)(delayed(init_run)(i) for i in time_list)
```