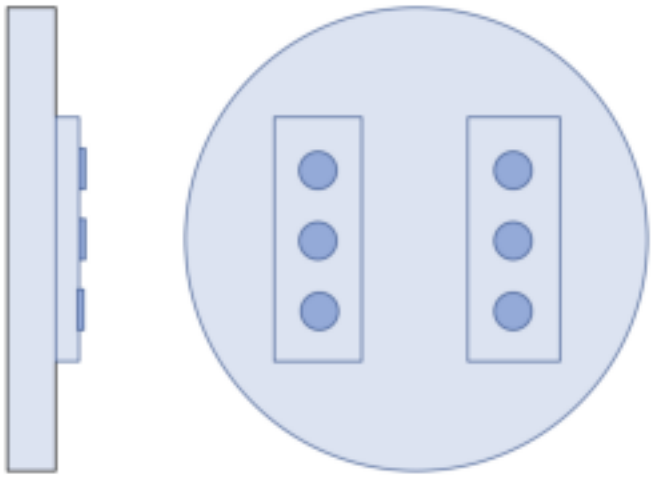
# baldr_phasemask_design_losses

November 17, 2023

[1]:
```
"""
Analysis of losses (Fresnel reflection) in a phase mask design for Baldr.
Also derive depth for optimal (90deg) phase shifts in H and J bands.

1/2" Si02 base
1mm thick Si02 wafer
Su-8 negative photoresist for dots
Noa61 UV curing adhesive for Si02 base / wafer interface.
"""
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import nbconvert
path = '/Users/bcourtne/Documents/ANU_PHD2/baldr'

os.chdir(path)
from functions import baldr_functions_2 as baldr

def fresnel_reflection(n1,n2,theta_i=0, polarisation='s'):
    """going material with refractive index n1 to n2"""
    theta_t = np.arcsin( n1 * np.sin(theta_i) / n2 )
    if polarisation == 's':
        R = abs( (n1*np.cos(theta_i)-n2*np.cos(theta_t)) / (n1*np.cos(theta_i) +␣
 ↪n2*np.cos(theta_t)) )**2
    else: # p
        R = abs( (n1*np.cos(theta_t)-n2*np.cos(theta_i)) / (n1*np.cos(theta_t) +␣
 ↪n2*np.cos(theta_i)) )**2
    return(R)

def fresnel_transmission(n1,n2,theta_i=0, polarisation='s'):
    """going material with refractive index n1 to n2"""
    T = 1 - fresnel_reflection(n1,n2,theta_i, polarisation)
    return(T)
```

[2]:
```
baldr_mask_figure = mpimg.imread( os.path.join( path,'figures/
 ↪baldr_mask_design_1.png') )
```

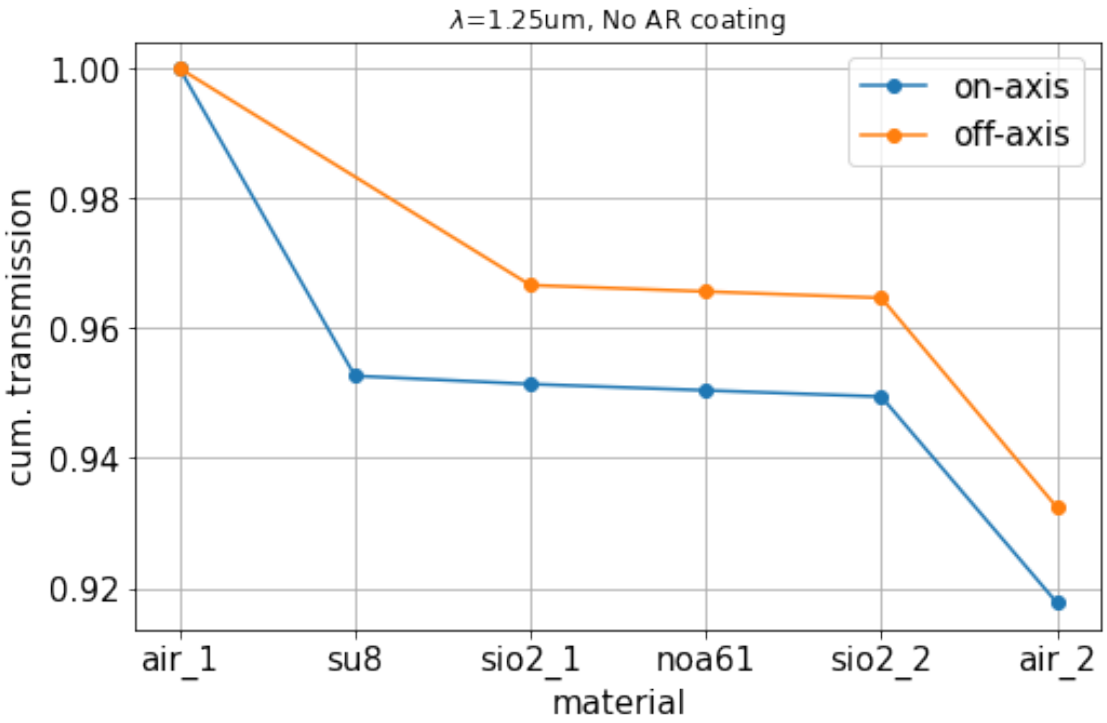
```
plt.imshow(baldr_mask_figure)
plt.axis('off')
plt.show()
```



[81]:
```
# calculate transmission without AR coating
wvls = np.linspace(1.25,1.65,2) #um

n_air = baldr.nglass( wvls, glass='air')
n_sio2 = baldr.nglass( wvls, glass='sio2')
n_su8 = baldr.nglass( wvls, glass='su8')
n_noa61 = baldr.nglass( wvls, glass='noa61')

#interfaces on-axis
interfaces_on_labels = ['air_1','su8', 'sio2_1', 'noa61', 'sio2_2', 'air_2']
interfaces_on = np.array( [n_air, n_su8, n_sio2, n_noa61, n_sio2, n_air ] )

#interfaces off-axis
interfaces_off_labels = ['air_1','sio2_1', 'noa61', 'sio2_2','air_2']
interfaces_off = np.array( [n_air, n_sio2, n_noa61, n_sio2, n_air ] )

T_on = [np.ones(len(wvls))] # init as 1
for i, _ in enumerate( interfaces_on[:-1] ):
    T_on.append(␣
 ↪fresnel_transmission(interfaces_on[i],interfaces_on[i+1],theta_i=0,␣
 ↪polarisation='s') )
```

```
T_off = [np.ones(len(wvls))] # init as 1
for i, _ in enumerate( interfaces_off[:-1] ):
    T_off.append(␣
 ↪fresnel_transmission(interfaces_off[i],interfaces_off[i+1],theta_i=0,␣
 ↪polarisation='s')  )

T_on_total = np.product(T_on,axis=0)
T_off_total = np.product(T_off,axis=0)

print(f'on axis mask transmission (without AR coating) at {wvls[0]}um =␣
 ↪{T_on_total[0]}\non axis mask transmission at {wvls[1]}um = {T_on_total[1]}')

print(f'\noff axis mask transmission (without AR coating) at {wvls[0]}um =␣
 ↪{T_off_total[0]}\noff axis mask transmission at {wvls[1]}um =␣
 ↪{T_off_total[1]}')

w_indx=0 # wvls index to plot
fig,ax = plt.subplots( 1,1, figsize=(8,5))
ax.plot( interfaces_on_labels, np.cumprod(T_on,axis=0)[:,w_indx]␣
 ↪,'-o',label='on-axis')
ax.plot( interfaces_off_labels, np.cumprod(T_off,axis=0)[:,w_indx]␣
 ↪,'-o',label='off-axis')
ax.set_ylabel('cum. transmission', fontsize=15)
ax.set_xlabel('material', fontsize=15)
ax.set_xlabel('material', fontsize=15)
ax.tick_params(labelsize=15)
ax.legend(fontsize=15)
ax.grid()
ax.set_title(r'$\lambda$'+f'={wvls[w_indx]}um, No AR coating')

#assuming SiO2 C broadband AR coating with at worst 99.5% transmission in J-H␣
 ↪(https://www.thorlabs.com/NewGroupPage9.cfm?ObjectGroup_ID=3983)
T_AR = 0.995
T_AR_total =  T_AR**2 * (1 - (np.cumprod(T_on,axis=0)[1,0] - np.
 ↪cumprod(T_on,axis=0)[-2,0] ) )
print(f'assuming SiO2 -C broadband AR coating with ~99.5% transmission, the␣
 ↪total transmission =  {T_ar_total} ')
```

```
on axis mask transmission (without AR coating) at 1.25um = 0.9177625349117127
on axis mask transmission at 1.65um = 0.9183622636007926

off axis mask transmission (without AR coating) at 1.25um = 0.9324563752349827
off axis mask transmission at 1.65um = 0.9334487260193335
assuming SiO2 AR coating with 99.5% transmission, the total transmission =
0.9868654099804911
```



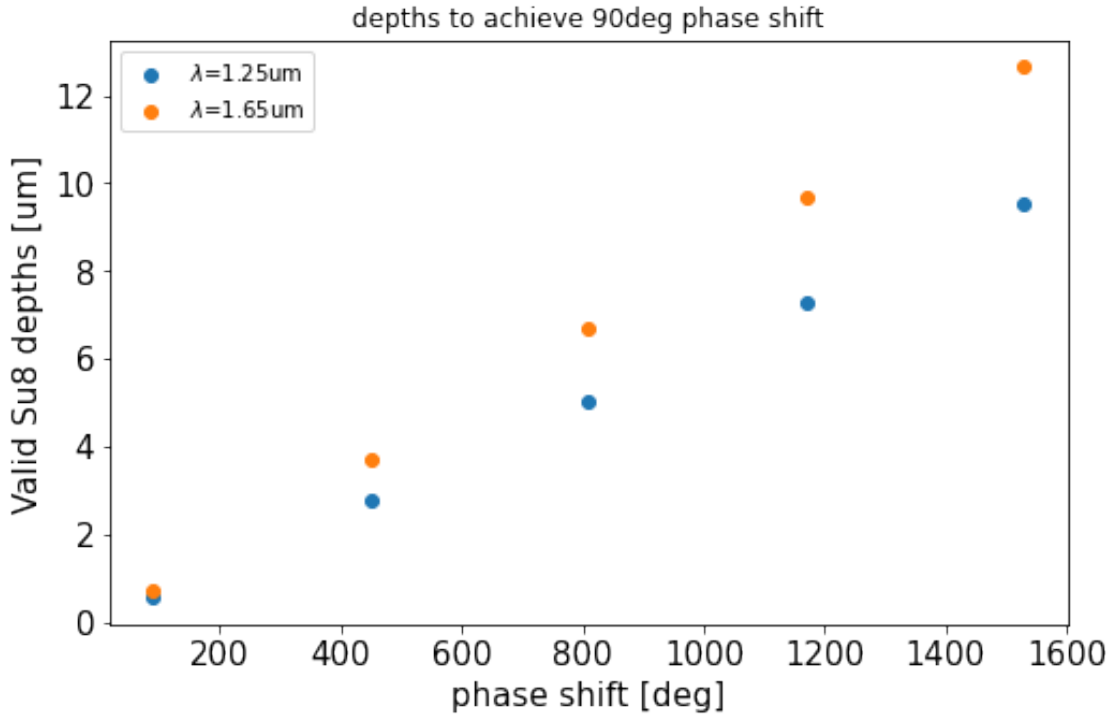

[ ]:
```
# Depth of Su8 for 90 degree phaseshift in J (1.25um) & H (1.65um)
```

[145]:
```
# neglecting glue interface with 1mm wafer + 1/2 SiO2 (https://www.thorlabs.com/
 ↪thorproduct.cfm?partnumber=WG40530-C)
desired_phase_shift = 90 #deg
z_dots = []
for d in 360 * np.arange(0,5):
    z_dots.append( ( np.deg2rad(90 + d) / (2 * np.pi) * (1e-6*wvls) ) /␣
 ↪(n_su8-n_air) )# calculate dot depth for each wavelength to get desired␣
 ↪phaseshift at that wavelength
    print( f'\nsu8 depth for {90 + d} degree phase shift at {wvls[0]}um is␣
 ↪{round(1e9 * z_dots[-1][0])}nm' )
    print( f'su8 depth for {90 + d} degree phase shift at {wvls[1]}um is␣
 ↪{round(1e9 * z_dots[-1][1])}nm' )

plt.figure(figsize=(8,5))
plt.plot( 360 * np.arange(0,5)+90 , 1e6 * np.array(z_dots)[:,0]␣
 ↪,'o',label=r'$\lambda$'+f'={wvls[0]}um')
plt.plot( 360 * np.arange(0,5)+90 , 1e6 * np.array(z_dots)[:,1]␣
 ↪,'o',label=r'$\lambda$'+f'={wvls[1]}um')
plt.xlabel('phase shift [deg]',fontsize=15)
plt.ylabel('Valid Su8 depths [um]',fontsize=15)
plt.title('depths to achieve 90deg phase shift')
```

```
plt.gca().tick_params(labelsize=15)
plt.legend()
```

```
su8 depth for 90 degree phase shift at 1.25um is 562nm
su8 depth for 90 degree phase shift at 1.65um is 744nm

su8 depth for 450 degree phase shift at 1.25um is 2808nm
su8 depth for 450 degree phase shift at 1.65um is 3719nm

su8 depth for 810 degree phase shift at 1.25um is 5055nm
su8 depth for 810 degree phase shift at 1.65um is 6694nm

su8 depth for 1170 degree phase shift at 1.25um is 7301nm
su8 depth for 1170 degree phase shift at 1.65um is 9669nm

su8 depth for 1530 degree phase shift at 1.25um is 9548nm
su8 depth for 1530 degree phase shift at 1.65um is 12644nm
```

[145]: `<matplotlib.legend.Legend at 0x7fbc6a168d30>`




[ ]:
```
"""z_sio2 = 2e-3 #m
z_dot = 1e-6
opl_off = n_air * z_dot + n_sio2 * z_sio2
```

```
opl_on = n_su8 * z_dot + n_sio2 * z_sio2

opd = opl_on - opl_off

phase_shift = (2 * np.pi / (1e-6*wvls)) * opd
 """
```